Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна технологія прогнозування побічних ефектів поліпрагмазії за допомогою графової нейронної мережі»

Виконав: студент 2-го курсу, групи 1КН-22м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
Кузняк В.П.
(прізвище та ініціали)

Керівник: к.т.н., проф. каф. КН
Колесницький О.К.
(прізвище та ініціали)
«07» 12 2023 р.

Опонент: д.т.н., проф. каф. КСУ
Биков М.М.
(прізвище та ініціали)
«07» 12 2023 р.

Допущено до захисту
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
(прізвище та ініціали)
«03» 12 2023 р.

Вінниця ВНТУ - 2023 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо-професійна програма – «Системи штучного інтелекту»

ЗАТВЕРДЖУЮ
Завідувач кафедри
КН
Д.т.н., проф. Яровий А.А.
29.08 2023
року.

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Кузняку Владиславу Павловичу

1. Тема роботи: «Інформаційна технологія прогнозування побічних ефектів поліпрагмазії за допомогою графової нейронної мережі»
керівник роботи: Колесницький Олег Костянтинович, к.т.н., проф.
затверджені наказом вищого навчального закладу "18" 09 2023 року №247
2. Строк подання студентом роботи 13. 11. 2023.
3. Вихідні дані до роботи:
Вхідні дані – формат вхідних даних – json, розмірність вхідних даних – 2 препарати, кількість класів розпізнавання – 1071 побічна дія, обсяг навчальної вибірки – не менше 300000, обсяг тестової вибірки – 10000, мова програмування – об'єктно-орієнтована.
4. Зміст текстової частини:
Вступ, аналіз предметної області прогнозування побічних ефектів поліпрагмазії, розробка інформаційної технології прогнозування побічних ефектів поліпрагмазії за допомогою графової нейронної мережі, програмна реалізація інформаційної технології прогнозування побічних ефектів поліпрагмазії, тестування та аналіз результатів роботи програми прогнозування побічних ефектів поліпрагмазії, економічна частина, висновки, перелік використаних джерел, додатки.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов`язкових креслень):
Архітектура гетерогенної графової мережі з увагою; схема алгоритму графової мережі; приклад згенерованого вектору, який описує побічні ефекти; створений гетерогенний граф та збережений у зручній формі для роботи з ним, вигляд інтерфейсу програми, результати тестування програми прогнозування побічних ефектів поліпрагмазії та програми-аналога.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Колесницький О. К., к.т.н., проф. каф. КН	29.08.23	14.11.23
15	Адлер О. О., к.т.н, доц. каф. ЕПВМ	2.09.23	21.11.23

7. Дата видачі завдання 29.08.2023р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Обґрунтування доцільності розробки інформаційної технології для прогнозування побічних ефектів поліпрагмазії	01.09.23	05.09.23	
2	Математичне моделювання інформаційної технології для прогнозування побічних ефектів поліпрагмазії	06.09.23	16.09.23	
3	Структурна організація та особливості програмної реалізації інформаційної технології для прогнозування побічних ефектів поліпрагмазії	17.09.23	07.10.23	
4	Підготовка економічної частини	08.09.23	23.10.23	
5	Апробація та/або впровадження результатів дослідження	24.10.23	01.11.23	
6	Оформлення пояснювальної записки, графічного матеріалу та презентації	02.11.23	10.11.23	

Студент _____ Кузняк В. П.

Керівник роботи _____ Колесницький О. К.

АНОТАЦІЯ

УДК 004.8

Кузняк В. П. Інформаційна технологія прогнозування побічних ефектів поліпрагмазії за допомогою графової нейронної мережі. Магістерська кваліфікаційна робота зі спеціальності 122 – комп'ютерні науки, освітня програма – системи штучного інтелекту.

Вінниця: ВНТУ, 2023. 102 с.

На укр. мові. Бібліогр.: 47 назв; рис.: 22; табл. 6.

Дана магістерська кваліфікаційна робота присвячена розробці програмного забезпечення для прогнозування побічних ефектів поліпрагмазії за допомогою графової нейронної мережі. Під час роботи розглянуті та проаналізовані існуючі методи прогнозування побічних ефектів поліпрагмазії. Як найбільш сучасний, було обрано архітектуру графової нейронної мережі. Було проаналізовано архітектури різних графових нейронних мереж та обґрунтовано вибір для даної задачі гетерогенної графової нейронної мережі з блоком уваги. Архітектура обраного типу мережі налічує 128 входів, два модуля - кодувальник та декодувальник, та вихідний шар із 1317 нейронів. Було спроектовано програмну реалізацію прогнозування побічних ефектів поліпрагмазії та написана за допомогою мови програмування Python з використанням фреймворку PyTorch Geometry. Достовірність прогнозування побічних ефектів поліпрагмазії на 0.021 AP@50 краще аналогу.

Графічна частина складається з 7 плакатів.

У економічному розділі розраховано суму витрат на розробку та виготовлення нового технічного рішення, яка складає 2847755,5 гривень, спрогнозовано орієнтовану величину витрат по кожній з статей витрат, розраховано чистий прибуток, термін окупності витрат для виробника 0,2 роки та економічний ефект для споживача при використанні даної розробки.

Ключові слова: поліпрагмазія, архітектура, медичний граф знань, графова нейронна мережа

ABSTRACT

Kuzniak V.P. Information technology for predicting side effects polypharmacy using a graph neural network. Master's thesis in the specialty 122 - computer sciences, educational program – artificial intelligence systems.

Vinnytsia: VNTU, 2023. 101 p.

In Ukrainian language. Bibliographer: 47 titles; fig .: 22; table 6

This master's thesis is devoted to the development of software for predicting the side effects of polypharmacy using a graph neural network. During the work, the existing methods of predicting the side effects of polypharmacy were considered and analyzed. The graph neural network architecture was chosen as the most modern. The architectures of various graph neural networks were analyzed and the choice of a heterogeneous graph neural network with a block of attention was justified for this task. The architecture of the selected type of network has 128 inputs, two modules - an encoder and a decoder, and an output layer of 1317 neurons. A software implementation for predicting side effects of polypharmacy was designed and written using the Python programming language using the PyTorch Geometry framework. The reliability of predicting the side effects of polypharmacy is 0.021 AP@50 better than the analogue.

The graphic part consists of 7 posters.

In the economic section, the amount of costs for the development and production of a new technical solution is calculated, which is 2847755.5 hryvnias, the estimated amount of costs for each of the cost items is predicted, the net profit is calculated, the payback period for the manufacturer is 0.2 years, and the economic effect for the consumer at using this development.

Keywords: polypharmacy, architecture, medical knowledge graph, graph neural network

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Більшість захворювань людини спричинені складними біологічними процесами, стійкими до дії будь-якого окремого препарату. Багатообіцяючою стратегією боротьби з хворобами є поліфармакотерапія, інша назва поліпрагмазія, тип комбінаторної терапії, який передбачає одночасне використання кількох лікарських засобів під час лікувального процесу. Оскільки комбінація препаратів можуть стимулювати активність окремих білків, вони покращують терапевтичну ефективність шляхом дії на різні шляхи патогенезу та надають комплексний ефект терапії. Наприклад, було продемонстровано, що комбінація препаратів Venetoclax та Idasanutlin призводить до чудової антилейкемічної ефективності при лікуванні гострого мієлоїдного лейкозу. Під час лікування два препарати діють взаємно на два різних шляхи регуляції апоптозу (регулюючий процес клітинної загибелі): Venetoclax пригнічує антиапоптичні білки сімейства Bcl-2, тоді як Idasanutlin активує шлях p53, який навпаки стимулює апоптоз. У результаті комбінація цих двох препаратів покращує виживаність пацієнтів шляхом одночасного впливу на комплементарні механізми.

Проте через складність людського організму складно передбачити появу нових ефектів непритаманних лікарським засобам роздільно. Поліпрагмазія стає все більш поширеним у сучасній медичній практиці. Це обумовлено зростанням кількості хронічних захворювань, старінням населення та появою нових лікарських засобів. Серед побічних дій можна виділити такі, як взаємодія препаратів, збільшення токсичності, зниження ефективності лікування та модуляція побічних ефектів. Це призвело до виникнення потреби розробки ефективного методу передбачення несприятливих ефектів під час лікування для виявлення потенційних побічних явищ та коригування терапії.

Не доцільне призначення препаратів може призвести до незадовільних результатів лікування, хронізації наявного захворювання, збільшення фінансових витрат на лікування та надмірного використання ресурсів медичної системи.

Графові нейронні мережі є відносно новим напрямом у глибокому навчанні, які ефективно моделюють взаємозв'язки між об'єктами в структуризованих даних. Даний тип мереж почав проникати в кожну галузь промисловості, включаючи медицину. Серед основних областей використання можна виділити фармакологію, хемоінформатику, нейрофізіологію тощо. Комбінація графових нейронних мереж із задачею прогнозування побічних ефектів поліпрагмазії може сприяти створенню нових методів для покращення точності та ефективності прогнозів виникнення побічних ефектів.

Таким чином, дослідження прогнозування побічних ефектів поліпрагмазії за допомогою графових нейронних мереж є актуальним та інноваційним напрямком, який має велике значення для покращення безпеки, ефективності та персоналізації лікування пацієнтів.

Мета та завдання дослідження. Метою роботи є підвищення точності передбачення взаємодії лікарських засобів під час терапії та виникнення побічних ефектів.

Об’єкт дослідження – процес комп'ютеризованого прогнозування виникнення побічних ефектів поліпрагмазії.

Предмет дослідження – інформаційна технологія та програмні засоби прогнозування побічних ефектів комбінації ліків на основі графової згорткової нейронної мережі та їх точність прогнозування.

Для досягнення наведеної мети були поставлені та вирішені наступні задачі:

- проаналізувати наявні методи для передбачення виникнення побічних ефектів поліпрагмазії;
- розробити інформаційну технологію прогнозування виникнення побічних впливу поліпрагмазії;

- вдосконалити наявну архітектуру інформаційної технології для прогнозування побічних ефектів поліпрагмазії;
- здійснити програмну реалізацію інформаційної технології для прогнозування побічних ефектів поліпрагмазії;
- провести тестування розробленого програмного забезпечення для передбачення побічних ефектів поліпрагмазії на бенчмарк наборі даних.

Методи дослідження. У роботі використано такі методи наукових досліджень: нейромережеві методи; статистичний аналіз; методи об'єктно-орієнтованого програмування для автоматизації розрахунків та створення програмного забезпечення.

Наукова новизна одержаних результатів полягає в наступному: удосконалено інформаційну технологію прогнозування побічних ефектів поліпрагмазії у пацієнтів, яка відрізняється застосуванням удосконаленої архітектури графової мережі, у яку додано блоки уваги та аугментація, що дозволило підвищити точність прогнозуванням.

Практичне значення одержаних результатів полягає у такому:

1. Схема алгоритму програми на основі згорткової графової нейронної мережі.
2. Схема процесу навчання згорткової графової нейронної мережі.
3. Програмна реалізація інформаційної технології для прогнозування побічних ефектів поліпрагмазії.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується строгістю постановки задач, коректним застосуванням математичних методів під час доведення наукових положень, строгим виведенням аналітичних співвідношень, порівнянням результатів з відомими та збіжністю результатів математичного моделювання з результатами, що отримані під час впровадження розроблених програмних засобів.

Особистий внесок магістрантів. Усі результати, наведені в магістерській роботі, було отримано самостійно. У працях здобувачу належать: аналіз процесу

прогнозування побічних ефектів поліпрагмазії та методи підвищення точності прогнозування побічних ефектів поліпрагмазії.

Апробація результатів роботи. Результати роботи були апробовані на конференції «Молодь в науці: дослідження, проблеми, перспективи-2024», Вінниця, жовтень 2023 – травень 2024 року [1].

Публікації. За результатами досліджень опубліковано одні тези доповіді на науково-технічній конференції [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ПРОГНОЗУВАННЯ ПОБІЧНИХ ЕФЕКТІВ ПОЛІПРАГМАЗІЇ

1.1 Постановка задачі прогнозування побічних ефектів поліпрагмазії

Перш за все, для розробки модуля гетерогенної графової згорткової нейронної мережі необхідно визначити процес створення блоків та оцінювання на наборі даних.

Графові нейронні мережі нещодавно викликали великий інтерес завдяки численним методам розробленим для різних галузей промисловості [2, 3, 4]. Розробка потужних і виразних архітектур GNN є ключовою проблемою для практичного застосування та реального впровадження графового машинного навчання у промисловість. Однак відстеження прогресу часто є складним завданням за відсутності визнаного науковою спільнотою стандарту, оскільки моделі, які оцінюються на традиційно використовуваних наборах даних із непослідовними експериментальними порівняннями, ускладнюють розрізнення складних, простих і незалежних від графів архітектур [5]. Тому для даної роботи підібраний спеціалізований набір даних, який містить складну структуру взаємодії між компонентами та складений з багатьох інших оцінювальних наборів окремих компонентів [6].

Вхідними даними в межах даної роботи є перелік використаних препаратів під час лікування. Ціль роботи полягає в тому, щоб використати можливості графової нейронної мережі використовувати структурні властивості даних знайти закономірність та взаємодію між різними діючими речовинами.

Метою роботи є огляд існуючих підходів та алгоритмів з метою створення нової архітектури для поліпшення попередніх результатів. Аналіз взаємодії діючих речовин включає аналіз великої графової бази знань про відомі взаємодії та побічні дії на цільові органи.

У цій роботі планується вирішити такі завдання:

- Підготовка набору даних для тренування мережі. Оптимізація форми подання даних для прискорення тренування та реалізації сходження процесу навчання.
- Визначення впливу додаткових блоків графової нейронної мережі на результат передбачення.
- Порівняння результатів розробленої програми з існуючою альтернативою.

Завдання можна сформулювати як отримання вихідних векторів, які залежать від взаємодії лікарських засобів у графовій базі знань (гетерогенний граф набору даних). Для вирішення цієї проблеми використовуватиметься гетерогенна графова згорткова нейронна мережа з блоком уваги для передбачення зв’язків у графовій медичній базі знань.

Для цього я спробую змінити наявну архітектуру алгоритма, яка здатна краще охопити власне інформацію в атрибутах, які описують власне лікарський засіб, та структурну взаємодію препаратів між собою й молекулами-мішенями для передбачення наявності зв’язку між лікарськими засобами, що призводить до виникнення побічного явища.

Отже, планується розроблення інтелектуальної технології для виконання аналізу графового набору даних. Вона вивчає структурні властивості зв’язків між різними сутностями та враховує індивідуальні властивості кожного компонента (препарату, протеїна). Результати будуть представлені у вигляді вектора, де кожне значення відповідатиме ймовірності появи певного класу побічних ефектів.

1.2 Огляд відомих методів

Методи обчислювальної фармакології та хемоінформатики спрямовані на пошук зв’язків між ліками та молекулярними мішенями, прогнозування потенційних побічних реакцій на ліки та пошук нових способів використання існуючих лікарських засобів. На відміну від окремих препаратів і однопрепаратної

терапії (тобто монотерапії), які переважно розглядаються наявними методами, мало програм враховують комбінацію ліків (тобто поліпрагмазію). Це важливо, оскільки поліпрагмазія є корисною стратегією для боротьби зі складними захворюваннями із важливими наслідками для системи охорони здоров'я.

Традиційно ефективні комбінації ліків виявляються шляхом експериментального скринінгу всіх можливих комбінацій попередньо визначеного набору препаратів. Враховуючи велику кількість ліків, експериментальні скринінги по парних комбінацій препаратів становлять величезну проблему з точки зору вартості та часу. Наприклад, для заданих n препаратів, існує $n(n-1)/2$ по парних комбінацій і багато інших комбінацій вищого порядку. Щоб усунути комбінаторний вибух, розроблено обчислювальні методи для визначення пар лікарських засобів, які потенційно проявляють активність до взаємодії, тобто пар лікарських засобів, які викликають понад нормову адитивну реакцію, очікувану за відсутності взаємодії [9]. Попередні дослідження в цій сфері були зосереджені на визначенні взаємодії між ліками за допомогою концепцій синергії та антагонізму [10], кількісного вимірювання кривих доза-ефект [11] і визначення того, чи взаємодіє дана пара ліків, відповідно до експерименту з вимірювання життєздатності клітин. Усі ці підходи передбачають взаємодію між ліками як скалярні значення, що представляють загальну ймовірність/силу взаємодії для даної пари ліків.

З розвитком комп'ютерних технологій та обчислювальних засобів дослідники випробували різні методи для збору інформації про взаємодію між ліками через медичні записи та наукову літературу [12]. Вони намагалися виявити їх за допомогою нейромережевих методів та напів контрольованого навчання (semi-supervised learning). Перші моделі використовували двоетапний конвеєр обробки даних, який зазвичай складається з моделі виділення ознак графа та моделі прогнозування зв'язку (ребра графа), обидві з яких навчаються окремо. Серед прикладів можна навести роботу [13], де використовується тристороння мережа під

назвою Linked Tripartite Network (LTN), перший блок якої називається Similarity Learning та відповідає за отримання корисних ознак із даних за допомогою алгоритму DeepWalk [14], а другий – Association discovering, відповідає за передбачення зв'язку між вершинами графа (сутностями, такими як препарат–препарат, препарат–білок) за допомогою методів висновку на основі правил [15].

Одним із відомих підходів являється Decagon на рисунку 1.1, метод прогнозування побічних ефектів пар лікарських засобів [16]. Їхня модель — це згорткова графова нейронна мережа, яка працює в мультиреляційній системі (система з великою кількістю зв'язків між компонентами). Дана модель започаткувала підхід end-to-end до задачі передбачення побічних ефектів поліпрагмазії, де одна модель виявляє та створює корисні ознаки з даних та робить на основі них передбачення.

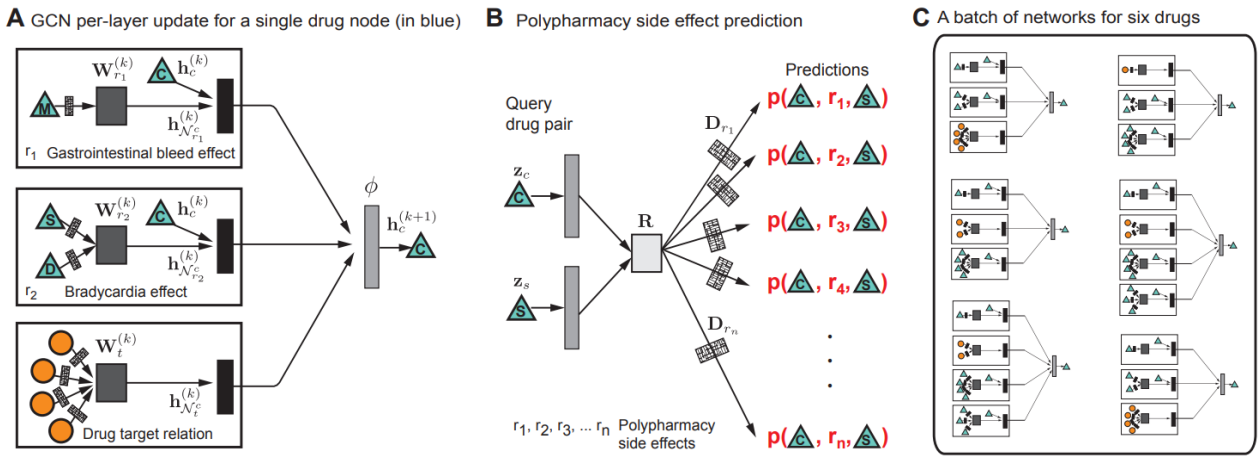


Рисунок 1.1 - Огляд архітектури моделі Decagon

Також хочеться згадати про розвиток підходів на основі графів, які використовують концепції роботи доповнення графу знань.

Прикладом роботи, яка посилила дослідження у цьому напрямку для виконання цієї задачі, де використовується створення вкладень (embeddings) графу

знань для задачі доповнення графу знань побічних дій від комбінації ліків являється робота [17]. Модель TriVec — це модель вбудовування на основі тензорної факторизації, яка вирішує проблему передбачення зв'язку як завершення 3D-тензора, де розміри тензора представляють сутності та зв'язки. У задачі прогнозування побічних ефектів поліпрагмазії комбінації ліків моделюються як суб'єкти та об'єкти трійок, тоді як відповідні побічні ефекти поліпрагмазії моделюються як відносини. У процесі навчання модель обробляє різні типи тверджень, такі як білок-білкова взаємодія, лікарсько-білкова взаємодія, лікарсько-лікарська взаємодія, побічні ефекти окремих ліків і побічні ефекти поліфармація. Це дозволяє моделі вивчати ефективні вбудовування для компонентів, що відповідають різним сутностям і зв'язкам у графі знань. На етапі прогнозування модель TriVec потім вивчає ймовірність асоціацій побічних ефектів поліпрагмазії з комбінаціями ліків, заповнюючи тривимірний тензор ліків і побічних ефектів поліпрагмазії, як показано на рисунку 1.3.

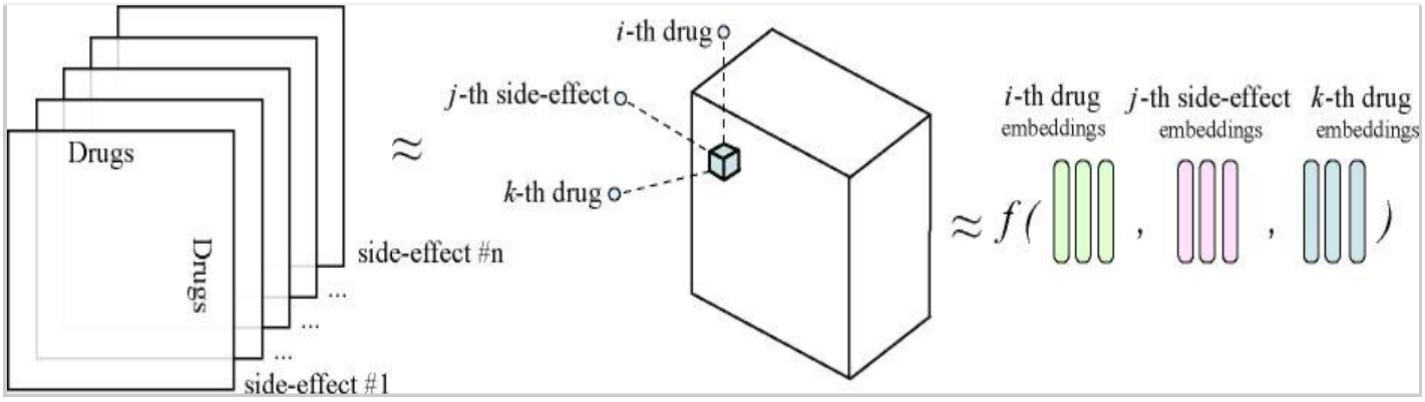


Рисунок 1.2 - Схема обробки даних у моделі TriVec

Метою всіх цих методів є моделювання можливих взаємодій між вбудованими графами та надання оцінки можливості існування зв'язків між сутностями у графі.

1.3 Обгрунтування вибору аналогу до програми прогнозування побічних

За програму-аналог було взято вищезгадану мережу Decagon, оскільки дана мережа має подібний підхід до виконаного в даній роботі, проте відрізняється методами закодовування отриманої інформації в латентні вектори та подальше декодування у вектор побічних ефектів.

1.4 Висновки до розділу 1

У розділі було розглянуто постановку задачі прогнозування побічних ефектів поліпрагмазії, проведено огляд відомих методів прогнозування побічних ефектів поліпрагмазії, які можна використовувати для поставленої задачі. Обґрунтовано доцільність використання графових нейронних мереж для прогнозування побічних ефектів поліпрагмазії. Було проаналізовано різні методи прогнозування побічних ефектів поліпрагмазії та обрано аналоги, головними недоліками яких є відсутність можливості аналізу декількох препаратів, що ставить мету дослідження – підвищення результативності інтелектуального аналізу.

2 РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ПРОГНОЗУВАННЯ ПОБІЧНИХ ЕФЕКТІВ ПОЛІПРАГМАЗІЇ ЗА ДОПОМОГОЮ ГРАФОВОЇ НЕЙРОННОЇ МЕРЕЖІ

2.1 Обгрунтування вибору графової нейронної мережі для прогнозування побічних ефектів поліпрагмазії

Графові нейронні мережі спрямовані на розширення існуючих нейронних мереж за допомогою теорії графів, що дозволяє їм працювати з даними, які мають графову структуру. Після успіху згорткових нейронних мереж, її хотіли імплементувати в графові нейронні мережі за допомогою спектра графових лапласівських операцій, які транслюють згорткові властивості в область Фур'є, що з'являється в більш прямому представленні даних графа.

Після реалізації архітектури оптимальної згортки до графової мережі почали використовувати додаткові аугментації, які поліпшили результат. Додаткові вдосконалення стосувалися розширення спектру роботи мережі на ребра графа, оскільки попередньо мережа працювала тільки з атрибувами вузлів. Далі значним прогресом було створення блоків уваги, які оцінювали важливість кожного сусіднього вузла в кінцевому передбачені.

Як зазначалося в основі інтелектуального модуля інформаційної системи використовується гетерогенна графова згорткова мережа з увагою (HAN) [18] над якою здійснилися додаткові аугментації для поліпшення отримання ознак, які сприятимуть отримання надійних та точніших результатів. Необхідність в гетерогенній структурі мережі пояснюється наявністю декількох типів ребер та вузлів, що неаналізуватимуться просто, гомогенною реалізацією.

Основним фактором, який сприяв вибору як основної мережі HAN є гетерогенність структури графу знань та доцільність використання анізотропності ознак всередині графа. Під анізотропністю мається на увазі, що оскільки всі

компоненти графа неоднаково впливають на остаточний висновок щодо наявності побічного ефекту при комбінації двох препаратів під час лікування, то нам потрібно визначати коефіцієнти важливості кожного препарату (сутності) та взаємозв'язку під час висновку.

Серед коефіцієнтів важливості виділяються два основних:

1. коефіцієнт важливості індивідуального вузла під час прийняття висновку;
2. коефіцієнт важливості металяху під час остаточного висновку.

Нижче перераховано основні аугментації здійснені над основною мережею:

1. Для об'єднання висновків з усіх метаграфів в один для подальшого аналізу з використанням додаткової мережі використовуватиметься техніка з'єднання (fusion) відносно важливості, щоб зробити граф більш *розрідженим* для прискорення розрахунків та *гомогенним* для акумуляції інформації в одному графі.
2. Для зменшення внутрішньої дисперсії ознак під час розрахунків використовуватиметься Multi-Head Attention механізм.
3. Для поліпшення отримання коефіцієнтів важливості під час проміжних етапів розрахунків використовуватиметься Gated Attention Aggregator. [19] У той час як агрегатор уваги з Multi-Heads має можливість, щоб дослідити кілька підпросторів представлення ознак між центральним вузлом і його сусідами, не всі ці підпростори однаково важливі; деякі підпростори можуть навіть не існувати для певних вузлів. Подача виходу з такої голови уваги, яка захоплює хибне уявлення, може ввести в оману остаточний прогноз моделі. Тому я обчислюватиму додатковий м'який вентиль (фільтр для голів уваги) між 0 (низька важливість) і 1 (висока важливість), щоб призначити різну важливість кожній голові.

Отже створена архітектура гетерогенної графової згорткової нейронної мережі використовуватиме потенціал однієї з найкращих архітектур для роботи з гетерогенними графами та додаткових блоків, які поліпшують точність результатів, розширюють можливості поточної мережі та дають напрямок для подальших досліджень.

2.2 Структура та моделювання графової нейронної мережі

Основним компонентом для опису структури досліджуваних даних у роботі являється граф. Граф (геометричний граф) — це фігура на площині, яка складається з непорожньої скінченної множини **V** точок (вершин) і скінченної множини **E** орієнтованих чи не орієнтованих ліній (ребер), що з'єднують деякі пари вершин. Як правило, граф визначається як **G = (V, E)**, де **V** — набір вузлів, а **E** — ребра між ними [20]. Приклад графа з аннотацією наведений на рисунку 2.1.

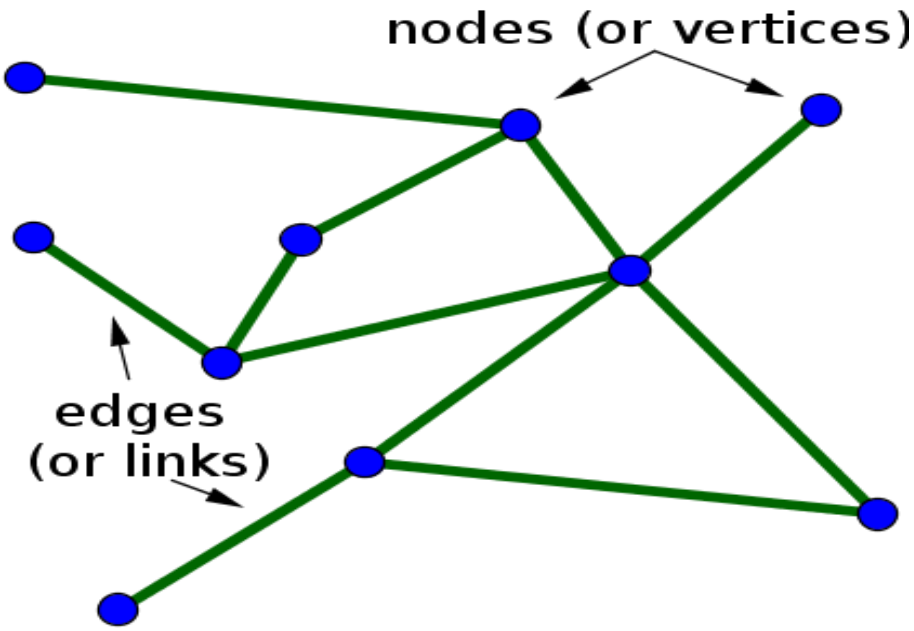
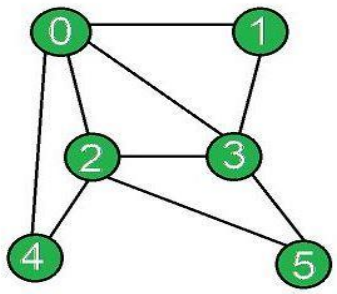


Рисунок 2.1 - Приклад графа з анотацією компонентів

Граф часто представляється матрицею суміжності **A,** приклад наведений нижче на рисунку 2.2.



	0	1	2	3	4	5
0	0	1	1	1	1	0
1	1	0	0	1	0	0
2	1	0	0	1	1	1
3	1	1	1	0	0	1
4	1	0	1	0	0	0
5	0	0	1	1	0	0

Рисунок 2.2 - Матриця суміжності для наведеного графу

Матриця суміжності графа - це квадратна матриця **A**, стовпці та рядки якої відповідають вершинам графа. Для неорієнтованого графа матриця суміжності дорівнює кількості ребер, інцидентних **i**-й і **j**-й вершинам, а для орієнтованого графа цей елемент матриці суміжності відповідає кількості ребер з початком у **i**-й вершині й кінцем у **j**-й. Таким чином, матриця суміжності неорієнтованого графа симетрична, а орієнтованого – необов'язково [20]. Компоненти матриці суміжності визначаються формулою 2.1

$$a_{i,j} = \begin{cases} 1 & \text{if } \{i,j\} \in E \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

Через свою структуру, графи складно аналізувати за допомогою наявних методів:

- Граф не існує в евклідовому просторі [21], а це означає, що він не може бути представлений будь-якою системою координат, з якою ми знайомі. Це значно ускладнює інтерпретацію даних графіка порівняно з іншими типами даних,

такими як зображення або сигнали часових рядів («текст» також можна розглядати як часові ряди), які можна легко відобразити на 2-D або 3-D евклідів простір.

- Граф не має фіксованої форми.
- Графи, як правило, важко візуалізувати для людської інтерпретації. У даному контексті мається на увазі величезні графи [22], які містять сотні чи тисячі вузлів. Через великий розмір та густо згруповані вузли ускладнюється розуміння графа людиною. Тому навчити комп’ютер виконувати це завдання також складно.

Тому для цього завдання створили новий тип архітектури нейронних мереж, який назвали графовими.

Графова нейронна мережа, як її називають, — це нейронна мережа, яку можна безпосередньо застосовувати до графів. Це зручний спосіб для завдання прогнозування на рівні вузла, ребер та цілого графу [23].

Оператори згортання, які реалізуються у графовій структурі, являются операторами розповсюдження для моделей GNN [24]. Основна ідея згорткових операторів полягає в тому, щоб узагальнити згортання від інших математичних областей до графового. У цьому напрямку дані методи часто класифікуються як спектральні та просторові.

Спектральні методи працюють із спектральним представленням графів. Ці методи теоретично засновані на обробці графічних сигналів [25] та визначають оператора згортання у спектральній області.

У спектральних методах графічний сигнал **x** спочатку перетворюється у спектральний домен графовим перетворенням Фур'є $\mathscr{F}$, а потім відбувається операція згортання. Після згортання сигнал перетворюється назад за допомогою зворотного графового перетворення Фур'є $\mathscr{F}^{-1}$. Ці перетворення визначаються як:

$$\mathscr{F}(\mathbf{x}) = \mathbf{U}^{\mathbf{T}}\mathbf{x}, \tag{2.2}$$

$$\mathscr{F}^{-1}(\mathbf{x}) = \mathbf{U}\mathbf{x}. \quad (2.3)$$

Тут **U** - матриця власних векторів нормалізованого Лапласіану графа:

$$\mathbf{L} = \mathbf{I}_N - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}, \quad (2.4)$$

(**D** - матриця ступенів вершин, а **A** - матриця суміжності графа).

Нормалізований Лапласіан графа - це справжня симетрична позитивна напівизначена матриця, тому він може бути факторизований як:

$$\mathbf{L} = \mathbf{U}\lambda\mathbf{U}^{\mathbf{T}}, \quad (2.5)$$

де λ - діагональна матриця власних значень.

На основі теореми згортання [26] операція згортання визначається як:

$$\mathbf{g} \bigstar \mathbf{x} = \mathscr{F}^{-1}(\mathscr{F}(\mathbf{g}) \odot \mathscr{F}(\mathbf{x})) = \mathbf{U}(\mathbf{U}^{\mathbf{T}}\mathbf{g} \odot \mathbf{U}^{\mathbf{T}}\mathbf{x}), \quad (2.6)$$

де $\mathbf{U}^{\mathbf{T}}\mathbf{g}$ є фільтром у спектральній області. Якщо ми спростимо заданий вираз, то фільтрація за допомогою навчаючої діагональної матриці $\mathbf{g}_w$ може виражатися наступним рівнянням:

$$\mathbf{g} \bigstar \mathbf{x} = \mathbf{U}\mathbf{g}_w\mathbf{U}^{\mathbf{T}}\mathbf{x}. \quad (2.7)$$

Кожна архітектура даного сімейства моделей відрізняється визначенням діагональної матриці як фільтра. Однак ця операція є обчислювально неефективною, а фільтр не є просторово локалізованим. [27] спробували зробити спектральні фільтри просторово локалізованими, вводячи параметр з плавними коефіцієнтами.

У нашому випадку згорткова графова нейронна мережа (GCN) спрощує операцію згортання у рівнянні (2.7) з $\mathbf{K} = 1$, щоб полегшити проблему перенавчання. Далі припускається, що $\boldsymbol{\lambda_{max}} \approx 2$ і спрощує рівняння (2.7) до:

$$\mathbf{g} \bigstar \mathbf{x} = w_0\mathbf{x} + w_1(\mathbf{L} - \mathbf{I_N})\mathbf{x} = w_0\mathbf{x} - w_1\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\,\mathbf{x}, \tag{2.8}$$

з двома вільними параметрами w_0 та w_1. З обмеженням параметра $w = w_0 = -w_1$ ми можемо отримати наступне рівняння:

$$\mathbf{g} \bigstar \mathbf{x} = w(\mathbf{I_N} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})\mathbf{x}. \tag{2.9}$$

Розв’язок рівняння під час тренування являється нестійким, що погіршує остаточну якість результатів, тому GCN у подальшому вводиться ренормалізаційний трюк для вирішення проблеми вибухаючого/затухаючого градієнта:

$$(\mathbf{I_N} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}) \rightarrow \check{\mathbf{D}}^{-1/2}\check{\mathbf{A}}\check{\mathbf{D}}^{1/2}, \tag{2.10}$$

де $\check{\mathbf{A}} = \mathbf{A} + \mathbf{I_N}$, а $\check{\mathbf{D}}_{\mathbf{ii}} = \sum_j^K \check{\mathrm{A}}_{\mathbf{ij}}$.

У підсумку, компактна форма GCN визначається через формулу

$$\mathbf{H} = \check{\mathbf{D}}^{-1/2}\check{\mathbf{A}}\check{\mathbf{D}}^{1/2}\mathbf{X}\mathbf{W}, \tag{2.11}$$

де $\mathbf{X} \in \mathbb{R}^{\mathbf{NxF}}$ є вхідною матрицею, $\mathbf{W} \in \mathbb{R}^{\mathbf{FxF`}}$ є матрицею параметрів і $\mathbf{H} \in \mathbb{R}^{\mathbf{NxF`}}$ є згортковою матрицею. $\mathbf{F}$ і $\mathbf{F`}$ є вхідною та вихідною розмірностями матриць відповідно. Необхідно звернути увагу, що GCN може розглядатися як просторова мережа за деяких умов.

Просторові підходи визначають згортання безпосередньо на графі на основі топології графів. Основним завданням просторових підходів є визначення операції згортання з сусідніми областями різного розміру та підтримкою локальної інваріантності CNN.

Однією з відомих архітектур даного підходу є GraphSAGE [28], яка започаткувала індуктивний підхід до вирішення задач. Вона генерує вбудовування шляхом вибірки та агрегації функцій з сусідніх областей, які прилягають до вибраного вузла.

$$\boldsymbol{h_{Nv}^{t+1}} = \mathbf{AGG_{t+1}}(\{\boldsymbol{h_u^t}, \forall u \in \boldsymbol{Nv}\}), \tag{2.12}$$

$$\boldsymbol{h_v^{t+1}} = \boldsymbol{\sigma}(\mathbf{W^{t+1}} \cdot [\boldsymbol{h_v^{t+1}} || \boldsymbol{h_{Nv}^{t+1}}]). \tag{2.13}$$

Замість того, щоб використовувати повний набір сусідніх вузлів, GraphSAGE рівномірно відбирає набір фіксованого розміру для агрегування інформації.

$\mathbf{AGG_{t+1}}$ - функція агрегації, яка пропонується в класичній інтерпретації у трьох видах: агрегатор середнього значення, агрегатор LSTM та агрегатор об'єднання. GraphSAGE із агрегатором середнього значення може розглядатися як індуктивна версія GCN, тоді як агрегатор LSTM не є інваріантним до перестановки, що вимагає визначеного порядку вузлів.

Оскільки задача вимагатиме передбачення на лікарських засобах, що можуть незустрічатися у тренувальному наборі (індуктивне навчання), ми вибираємо GraphSAGE архітектуру щодо отримання інформації (передачі повідомлення) тільки від сусідніх вузлів. Проте не кожен вузол може бути важливим для прийняття рішення про наявність зв’язку (побічної дії). Тому нам потрібно ввести додатковий блок, а саме поняття уваги.

Блок уваги на графах вперше був введений у роботі [29], де була створена мережа GAT. Механізм уваги успішно використовується у багатьох завданнях на основі послідовностей, таких як машинний переклад [30], машинне читання [31] і

так далі. Порівняно з операторами, які згадувалися раніше, оператори на основі уваги призначають різні ваги для сусідів, щоб вони могли полегшити шуми та досягти кращих результатів. Графічна мережа уваги (GAT) включає механізм уваги в етап поширення (передачі повідомлення). Вона обчислює приховані стани кожного вузла, відвідуючи своїх сусідів, слідуючи за стратегією самооцінки. Прихований стан вузла v може бути отриманий рівнянням:

$$\boldsymbol{h_v^{t+1}} = \boldsymbol{\rho}\left(\sum_{\boldsymbol{u}\in N_v} \boldsymbol{\alpha}_{vu}\mathbf{W}h_u^t\right), \tag{2.14}$$

$$\boldsymbol{\alpha}_{vu} = \frac{\exp\left(\text{LeakyReLU}(\mathbf{a}^{\mathbf{T}}[\mathbf{W}\mathbf{h}_v||\mathbf{W}\mathbf{h}_u])\right)}{\sum_{k\in N_v}\exp\left(\text{LeakyReLU}(\mathbf{a}^{\mathbf{T}}[\mathbf{W}\mathbf{h}_v||\mathbf{W}\mathbf{h}_k])\right)}. \tag{2.15}$$

Тут **W** є матрицею ваг, що пов'язані з лінійною трансформацією, яка застосовується до кожного вузла, а **a** є вектором ваг одношарового MLP.

Більше того, GAT використовує багато голів уваги, концепцію яку використовують [32] для стабілізації процесу навчання. Він застосовує незалежні матриці голів уваги для обчислення прихованих станів, а потім об'єднує їхні ознаки (або обчислює середнє значення), в результаті чого отримуємо наступні два представлення:

$$h_v^{t+1} = ||_{k=1}^{K}\ \sigma\left(\sum_{\boldsymbol{u}\in N_v}\boldsymbol{\alpha}_{vu}^k\mathbf{W}h_u^t\right), \tag{2.16}$$

$$h_v^{t+1} = \ \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{\boldsymbol{u}\in N_v}\boldsymbol{\alpha}_{vu}^k\mathbf{W}h_u^t\right). \tag{2.17}$$

Тут $\boldsymbol{\alpha}^k_{ij}$ - це нормалізований коефіцієнт уваги, обчислений *k*-тою головою уваги.

Архітектура уваги має декілька властивостей:

(1) Обчислення взаємодії сусідніх пар вузлів паралелізується, таким чином, виконання математичних операцій є ефективним;

(2) Це можна використовувати до графових вузлів з різним ступенем, вказуючи довільні ваги сусідам;

(3) Це можна легко застосувати до індуктивних проблем навчання.

Для додаткового поліпшення показників буде використовуватися ще один блок, пов'язаний з воротами уваги. Подібний принцип використовується в GaAN мережі [33]. Інтуїтивно, замість того, щоб кожну голову уваги розглядати рівноцінною під час висновку, ми вводимо додаткові коефіцієнти, які їх оцінюють. Математично це виражається наступними формулами:

$$\boldsymbol{y}_i = \mathbf{FC}_{\theta_o}(\boldsymbol{x_i} \oplus \,||_{k=1}^{K}(g_i^{(k)} \sum_{j \in N_v} w_{ij}^{(k)}\, \mathbf{FC}_{\theta_v^{(k)}}^{h}(\boldsymbol{z}_j))), \tag{2.18}$$

$$\boldsymbol{g_i} = \left[g_i^{(1)}, g_i^{(2)}, \ldots, g_i^{(K)}\right] = \psi_g(\mathbf{x}_i, \mathbf{z}_{N_i}), \tag{2.19}$$

де $g_i^{(k)}$ є скаляром, значення воріт k-ої головки при вузлі **i**. Щоб переконатися, що додавання воріт не вводить занадто багато додаткових параметрів, ми використовуємо згорткову мережу ψ_g, яка приймає центральний вузол і ознаки сусідніх вузлів для генерації значень воріт. Всі інші параметри мають ті ж значення, що і в рівнянні (2.15).

Хоча перераховані вище графові нейронні мережі продемонстрували відмінний результат на бенчмарках, вони були створені для даних шомогенного графа, який містить один тип вузлів та ребер. Дані, які виникають в практичних ситуаціях, часто містять комплекс складніших взаємозв'язків, які поділяються на декілька типів та вимагають більш складніших методів аналізу для використання допоміжних ознак.

Неоднорідність є внутрішньою властивістю неоднорідного графа, тобто різноманітних типів вузлів і ребер. Наприклад, різні типи вузлів мають різні ознаки, і їхні функції можуть потрапляти в інший простір ознак. HG визначається як граф **G = {V, E}**, у якому **V** і **E** представляють набір вузлів і набір ребер відповідно.

Кожен вузол $v \in \mathbf{V}$ і кожне ребро $e \in \mathbf{E}$ асоційовані зі своєю функцією відображення $\mathbf{f}(v)\mathbf{: V \rightarrow A}$ і $\mathbf{j}(e)\mathbf{: E \rightarrow R}$. **A** і **R** позначають набір типів вузлів і набір типів ребер відповідно, де $|\mathbf{A}| + |\mathbf{R}| > 2$.

Схема мережі для **G** визначається як **S = (A, R)**, який можна розглядати як меташаблон неоднорідного графа **G = {V, E}** із функцією відображення типу вузла $\mathbf{f}(v)\mathbf{: V \rightarrow A}$ та функцією відображення типу краю $\mathbf{j}(e)\mathbf{: E \rightarrow R}$. Схема мережі — це граф, визначений над типами вузлів A, з ребрами як типами відношень із R.

HG не лише надає графову структуру асоціації даних, але й відображає семантику вищого рівня. Приклад HG проілюстровано на рисунку 2.3, який складається з чотирьох типів вузлів (автор, стаття, місце проведення та тематикастатті) і трьох типів країв (автор - стаття, стаття – тематика статті та стаття – конференція). Щоб сформулювати семантику зв’язків вищого порядку між сутностями, додатково пропонується поняття мета-шляху [34].

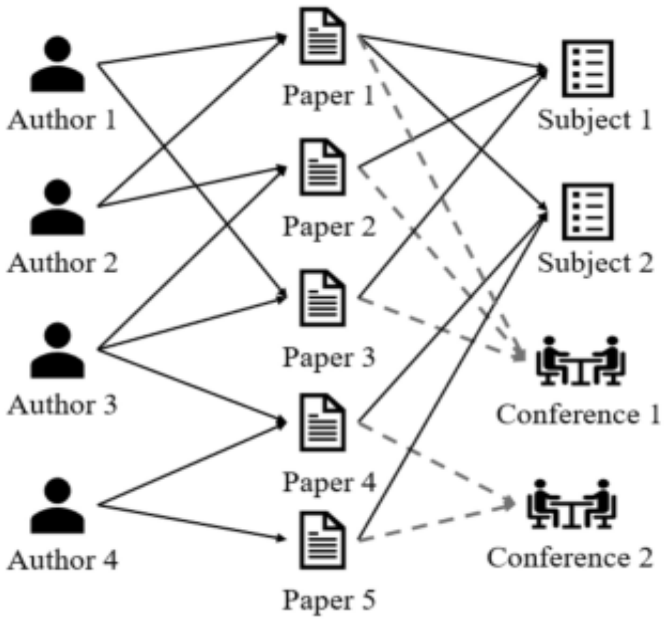


Рисунок 2.3 – Приклад гетерогенного графа

Для аналізу гетерогенного графу виділяють додаткові структури вищого семантичного рівня. Одна із них являється меташлях зображений на рисунку 2.4.

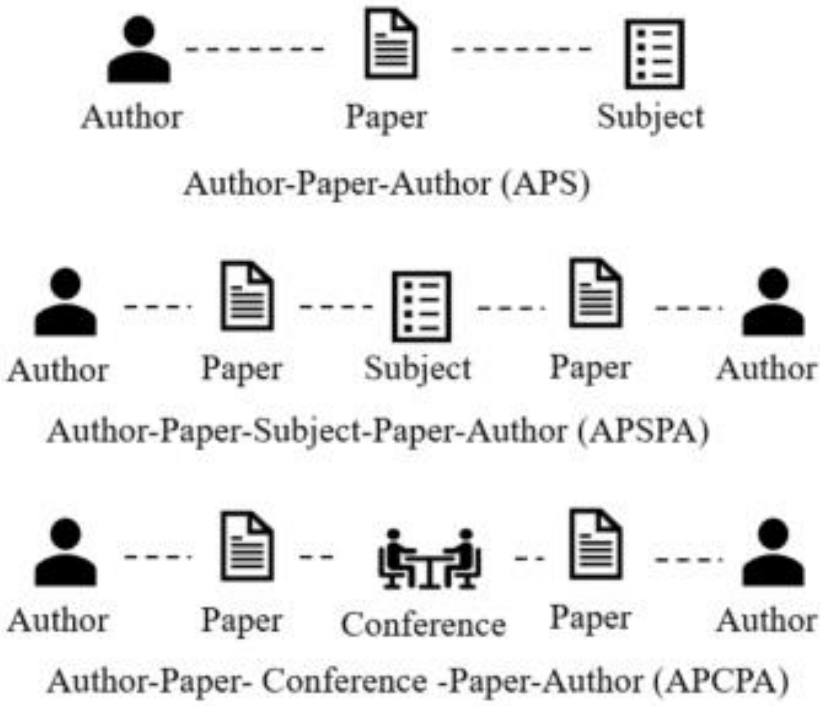


Рисунок 2.4 - Приклад металяхів у гетерогенному графі

Зауважте, що різні мета шляхи описують семантичні зв'язки в різних представленнях. Наприклад, металяхів APA вказує на зв'язок співавторства, а APCPA представляє зв'язок спільної конференції. Обидва вони можуть бути використані для формулювання спорідненості за авторами. Хоча мета-шлях можна використовувати для зображення зв'язку між сутностями, він не в змозі охопити більш складний зв'язок, такий як мотиви [35]. Щоб вирішити цю проблему, мета-граф [36] пропонує використовувати спрямований ациклічний граф типів сутностей і відношень для охоплення складніших зв'язків між сутностями.

Метаграф $\boldsymbol{T}$ можна розглядати як спрямований ациклічний граф (DAG), що складається з кількох металяхів із спільними вузлами. Формально метаграф визначається як $\boldsymbol{T} = (V_T, E_T)$, де V_T — набір вузлів, а E_T — набір ребер. Для будь-якого вузла $v \in V_T$, $\mathbf{f}(v) \in A$. Приклад метаграфа показано на рисунку 2.5, який

можна розглядати як комбінацію металяху APA та APCPA, що відображає подібність високого порядку двох вузлів.

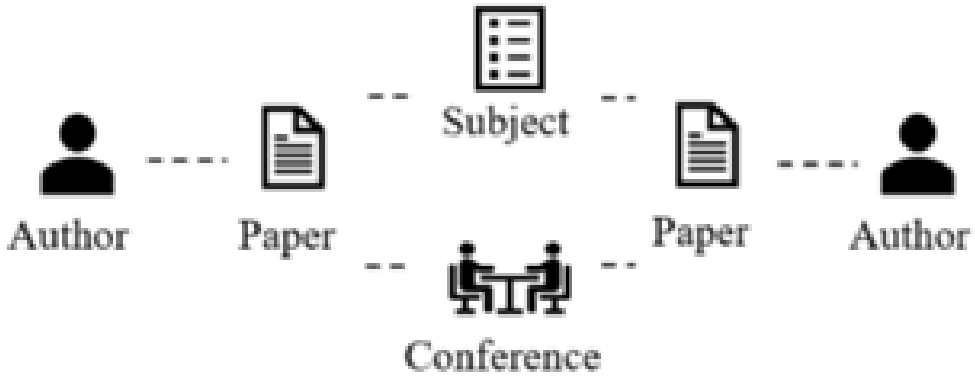


Рисунок 2.5 - Приклад метаграфу

Серед основних складнощів під час роботи з гетерогенними графовими мережами виділяють наступні:

- Складність структури. В однорідному графі основними структурами можуть розглядатися структури першого порядку, другого порядку і навіть вищого порядку. Всі ці структури добре виражені і мають гарну інтуїцію. Проте структура в HG кардинально змінюватиметься залежно від обраних співвідношень. Для прикладу візьмемо академічний граф на Рисунку 6, сусідами однієї статті будуть автори з відношенням «запис»; у той час як із відношенням «містить» сусіди стають «предмет». Ще більше ускладнюючи ситуацію, поєднання цих зв'язків, які можна розглядати як структури вищого порядку в HG, призведе до різних і більш складних структур:

- Гетерогенні атрибути (проблема злиття, викликана неоднорідністю атрибутів). Оскільки вузли та ребра в однорідному графі мають однаковий тип, кожен вимір атрибутів вузла чи ребра має однакове значення. У цій ситуації вузол може безпосередньо зливати атрибути своїх сусідів. Однак у HG атрибути різних

типів вузлів і країв можуть мати різне значення [37]. Наприклад, атрибути автора можуть бути галузями дослідження, тоді як стаття може використовувати ключові слова як атрибути. Тому те, як подолати різнорідність атрибутів і ефективно об’єднати атрибути сусідів, становить ще одну проблему для вбудовування HG

- Залежить від застосування. HG тісно пов'язана з реальними ситуаціями, тоді як багато практичних проблем залишаються невирішеними. Наприклад, побудова відповідної HG може вимагати достатніх знань предметної області в реальному застосуванні. Крім того, мета-шлях і/або мета-граф широко використовуються для фіксації структури HG. Однак, на відміну від однорідного графа, де структура (наприклад, структура першого та другого порядку) чітко визначена, вибір металяху також може потребувати попередніх знань. Крім того, для кращого сприяння додаткам у реальному світі нам зазвичай потрібно детально кодувати додаткову інформацію (наприклад, атрибути вузла) [37] або більш просунуте знання домену до процесу вбудовування HG.

Тому для вирішення задачі була запропонована нова архітектура, яка врахувала усі особливості типу даних та була направлена на вирішення подібних проблем. Модель має ієрархічну структуру уваги: увага на рівні вузла → увага на семантичному рівні. На рисунку 9 представлено всю структуру HAN. По-перше, у ній запропонована увага на рівні вузла, щоб дізнатися вагу сусідів на основі мета шляху та агрегувати їх, щоб отримати семантичне вбудовування вузла.

Після цього HAN може визначити різницю між металяхами за допомогою уваги семантичного рівня та отримати оптимальну зважену комбінацію вбудовування семантичного вузла для конкретного завдання як на рисунку 2.6.

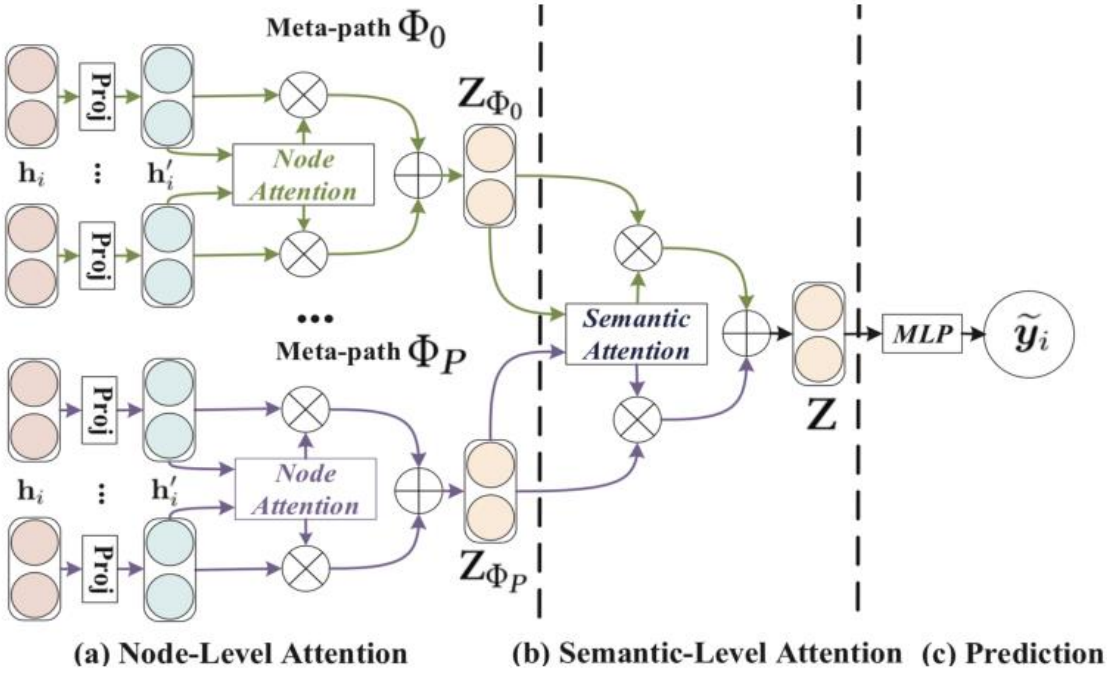


Рисунок 2.6 - Архітектура гетерогенної графової мережі з увагою

Перед агрегацією інформації від сусідів металяху для кожного вузла, можна помітити, що сусіди на основі металяху кожного вузла відіграють різну роль і демонструють різну важливість у вбудовуванні вузла навчання для конкретного завдання. Тут представляється, що увага на рівні вузла може дізнатися про важливість сусідів на основі металяху для кожного вузла в неоднорідному графі та агрегувати представлення цих значущих сусідів, щоб сформувати вбудовування вузла. Через неоднорідність вузлів різні типи вузлів мають різні простори ознак. Тому для кожного типу вузлів, розробилася специфічна для типу матрицю перетворення M_{φ_i} для проектування функцій різних типів вузлів у той самий простір ознак. На відміну від [14], специфічна для типу матриця перетворення базується на типі вузла, а не на типі ребра. Процес проектування можна показати таким чином:

$$\boldsymbol{h_i'} = M_{\varphi_i} \cdot h_i, \tag{2.20}$$

де h_i, та $\boldsymbol{h_i'}$ є оригінальною та прогнозованою ознаками вузла i. За допомогою операції проектування, що залежить від типу, увага на рівні вузла може обробляти довільні типи вузлів.

Після цього можна добавити блок самоуваги [32], щоб дізнатися про вагу різних типів вузлів. Маючи пару вузлів (i, j), які з'єднані через металях **Φ**, увага на рівні вузла $\boldsymbol{e_{ij}^{\Phi}}$ може дізнатися важливість $\boldsymbol{e_{ij}^{\Phi}}$, що означає, наскільки важливим буде вузол j для вузла i. Важливість пари вузлів на основі металяху (i, j) можна сформулювати наступним чином:

$$\boldsymbol{e_{ij}^{\Phi}} = \text{att}_{\text{node}}(\boldsymbol{h_i'}, \boldsymbol{h_j'}; \Phi). \tag{2.21}$$

У даному рівнянні att_{node} означає глибоку нейронну мережу, яка виконує увагу на рівні вузла. Враховуючи мета-шлях Ф, att_{node} є спільним для всіх пар вузлів на основі мета-шляху. Це тому, що в одному металяху є кілька схожих шаблонів з'єднання. Наведене вище рівняння (2) показує, що заданий металях Ф, вага пари вузлів на основі металяху (i, j) залежить від їх характеристик. Також, відповідно до рівняння $\boldsymbol{e_{ij}^{\Phi}}$ є асиметричним, тобто важливість вузла i для вузла j та важливість вузла j для вузла i можуть відрізнятися. Це показує, що увага на рівні вузла може зберігати асиметрію, яка є критичною властивістю гетерогенного графа.

Додатково в модель вводиться структурна інформація через масковану увагу, що означає, що ми обчислюємо лише $\boldsymbol{e_{ij}^{\Phi}}$ для вузлів $j \in \boldsymbol{N_i^{\Phi}}$, де $\boldsymbol{N_i^{\Phi}}$ позначає сусідів на основі металяху вузла i (включає себе). Після отримання важливості між парами вузлів на основі металяху дану величину потрібно нормалізувати, щоб отримати ваговий коефіцієнт α_{ij}^{Φ} за допомогою функції softmax:

$$\alpha_{ij}^{\Phi} = \text{softmax}(\boldsymbol{e}_{ij}^{\Phi}) = \frac{\exp\left(\sigma(\mathbf{a}_{\Phi}^{T} \cdot [h_i' \| h_j'])\right)}{\sum_{k \in N_i^{\Phi}} \exp\left(\sigma(\mathbf{a}_{\Phi}^{T} \cdot [h_i' \| h_k'])\right)}, \quad (2.22)$$

де σ позначається функція активації, $\|$ позначається операція конкатенації, а $\mathbf{a}_{\Phi}$ — вектор уваги на рівні вузла для меташляху Ф. Як видно з рівняння (3) ваговий коефіцієнт (i, j) залежить від їх особливостей. Також ваговий коефіцієнт α_{ij}^{Φ} зберігає властивість асиметричності, що означає, що вони роблять різний внесок один в одного. Не лише тому, що порядок об'єднання в чисельнику, але й тому, що вони мають різних сусідів, тому нормалізований член (знаменник) буде досить різним.

Тоді вбудовування вузла i на основі меташляху може бути агреговано спроектованими ознаками сусідів з відповідними коефіцієнтами наступним чином:

$$z_i^{\Phi} = \sigma(\boldsymbol{j} \in N_i^{\Phi} \alpha_{ij}^{\Phi} \cdot \boldsymbol{hj}'), \quad (2.23)$$

де z_i^{Φ} це вивчене вбудовування вузла i для меташляху Ф.

Щоб краще зрозуміти процес агрегування на рівні вузла, додано ілюстративне пояснення на рисунку 2.7.

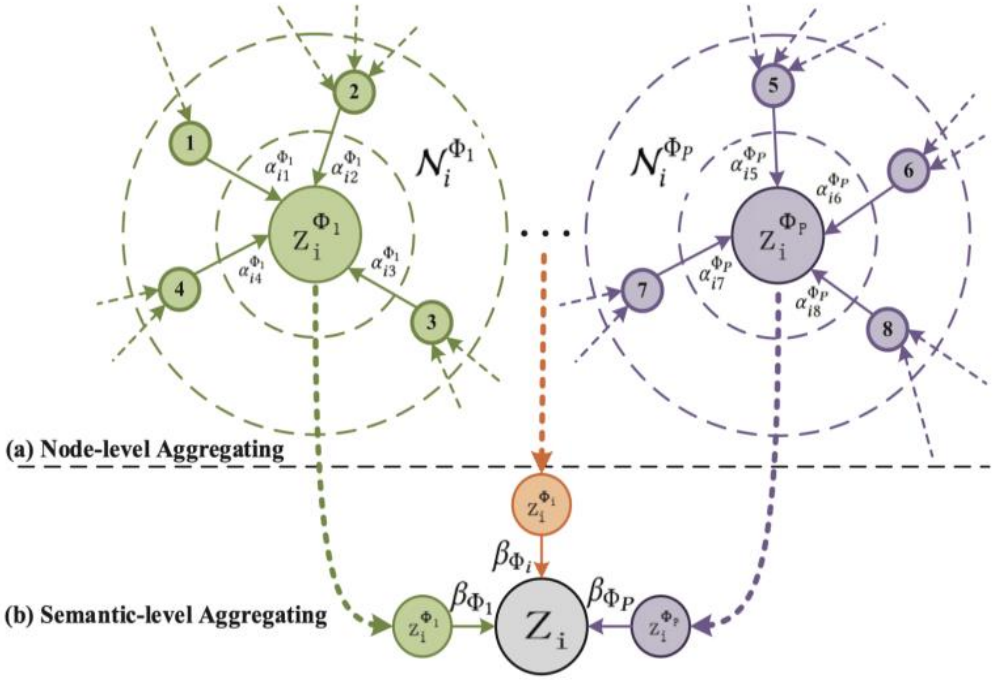


Рисунок 2.7 - Пояснення архітектури HAN на рівні вузла

Оскільки вага уваги α_{ij}^{Φ} генерується для одного металяху, він є семантично-специфічним і може отримувати один вид семантичної інформації.

Різнорідний граф має властивість вільного масштабу, дисперсія даних графа є досить високою. Щоб вирішити цю проблему, ми розширюється увага на рівні вузла до уваги з кількома головами, щоб процес навчання був більш стабільним.

Враховуючи набір металяхів $\{\Phi_1, \ldots, \Phi_P\}$, після подачі характеристик вузла в увагу на рівні вузла, ми можемо отримати P групи семантичних специфічних вбудовувань вузла, позначених як $Z_{\Phi 1}, \ldots, Z_{\Phi p}$.

Кожен вузол гетерогенного графа містить декілька типів семантичної інформації та семантично-специфічне вбудовування вузла можуть відображати вузол лише з одного аспекту. Для більш комплексного створення вбудовування вузлів у вигляді векторів, у мережі відбувається об'єднання декількох семантик, які розкриваються за допомогою металяхів. Щоб вирішити проблему вибору

металяхів і семантичного злиття в гетерогенному графі, вводиться додаткова увага на семантичному рівні, щоб автоматично вивчати важливість різних металяхів і об'єднувати їх для конкретного завдання. Взявши P групи семантичних специфічних вбудованих вузлів, отриманих від уваги на рівні вузла, як вхідні дані, вивчені ваги кожного металяху ($\beta_{\Phi 1}, \ldots, \beta_{\Phi p}$) можна показати таким чином:

$$\left(\beta_{\Phi 1}, \ldots, \beta_{\Phi p}\right) = att_{sem}(Z_{\Phi 1}, \ldots, Z_{\Phi p}) \quad (2.24)$$

У даному випадку att_{sem} позначає глибоку нейронну мережу, яка виконує операцію уваги на семантичному рівні. Дане рівняння демонструє, що увага на семантичному рівні може захоплювати різні типи семантичної інформації у гетерогенному графі.

Щоб зрозуміти важливість кожного металяху, відбувається трансформація семантичного вбудовування через нелінійне перетворення (наприклад, одношаровий MLP). Далі після цього вимірюється важливість семантичного вбудовування як подібність трансформованого вкладення з вектором уваги семантичного рівня q. Окрім того, відбувається усереднення важливістей усіх семантично-специфічних вбудовувань вузлів, що можна пояснити важливістю кожного металяху. Важливість кожного металяху, позначеного як $w_{\Phi i}$, показано наступним чином

$$w_{\Phi p} = \frac{1}{|V|} \sum_{i \in V} \boldsymbol{q}^T \cdot \tanh\left(\mathbf{W} \cdot \mathbf{z}_i^{\Phi p} + \mathbf{b}\right), \quad (2.25)$$

де W — вагова матриця, b — вектор зміщення, q — вектор уваги семантичного рівня. Зауважте, що для значущого порівняння всі вищезазначені параметри є спільними для всіх металяхів і семантичного вбудовування.

Після закінчення операції отримання важливості кожного мета-шляху у гетерогенному графі, проводиться операція нормалізації за допомогою функції softmax. Вага меташляху Φi, позначеного як $\beta_{\Phi i}$, може бути отримана шляхом нормалізації зазначеної вище важливості всіх меташляхів за допомогою функції softmax,

$$\beta_{\Phi p} = \frac{\exp(w_{\Phi p})}{\sum_{p=1}^{P} \exp(w_{\Phi p})}, \tag{2.26}$$

який можна інтерпретувати як внесок меташляху Φ_p для конкретного завдання. Потрібно звернути увагу, що чим вище $\beta_{\Phi p}$, тим важливішим є меташлях Φ_p. Зауважте, що для різних завдань мета-шлях Φ_p може мати різні ваги. Далі вивчені ваги як коефіцієнти можна об'єднати зі специфічними семантичними вбудовування, щоб отримати остаточне вбудовування $\mathbf{Z}$ на графі наступним чином:

$$\mathbf{Z} = \sum_{p=1}^{P} \beta_{\Phi p} \cdot \mathbf{Z}_{\Phi p} . \tag{2.27}$$

Остаточне вбудовування агрегується всіма семантично-специфічними вбудовуваннями. У кінці використовується остаточне вбудовування до конкретних завдань і розробляється функція втрат, яка використовуватиметься під час тренування.

2.3 Розробка алгоритму роботи програмного забезпеченя прогнозування побічних ефектів поліпрагмазії

Для реалізації основної функції алгоритму було створено наступну діаграму для зображення основної архітектури гетерогенної графової згорткової нейронної мережі з механізмом уваги.

На рисунку 2.8 нижче можна ознайомитися з детальнішою архітектурою.

Технологія складається з двох основних блоків обробки інформації, а саме з кодувальника та декодувальника. Кодувальник приймає на вхід інформацію про препарати, яка містить перелік побічних ефектів індивідуального використання та взаємозв'язок з білками. Через розрідженість вектора, який створиться з переліку побічних ефектів, всередині зідйснюватиметься додаткове нілінійне перетворення, яке зменшить розмірність вектора та узагальнить інформацію по всім індивідуальним ефектам. Також це зменшить в подальшому кількість розрахунків та пришвидшить тренування. Окрім того, оскільки наявний граф знань, на якому трнеувалася модель, не містив додаткової інформації про протеїни, я добавив додатковий блок, який по унікальним id присвоював кожному протеїну вектор, який змінювався під час навчання, щоб вивчити властивості кожного протеїна в графі. Ці вектори можуть приховувати інформацію про шляхи фізіологічної активації при взаємодії з окремими препаратами, фізико-хімічні та біохімічні властивості тощо. Далі отримані вектори для препаратів та білків оброблятимуться графовим модулем. Ці перетворення включатимуть аналіз атрибутів від сусідніх вузлів (для ліків це протеїни, а для протеїнів це інші протеїни та ліки); передача інформації від сусідніх вузлів до основного, атрибути якого змінюватимуться під час тренування; агрегація інформації та оновлення. У кінці мережа даватиме два латентних вектори, які описуватимуть вхідні препарати.

Усередині декодувального модуля отримані латентні вектори піддаватимуться додатковій передобробці та використовуватимуться шаром обробки для отримання остаточного результату. Як шар обробки в найпростішому випадку може використовуватися внутрішній добуток між векторами, а в найскладнішому методи роботи з графами знань.

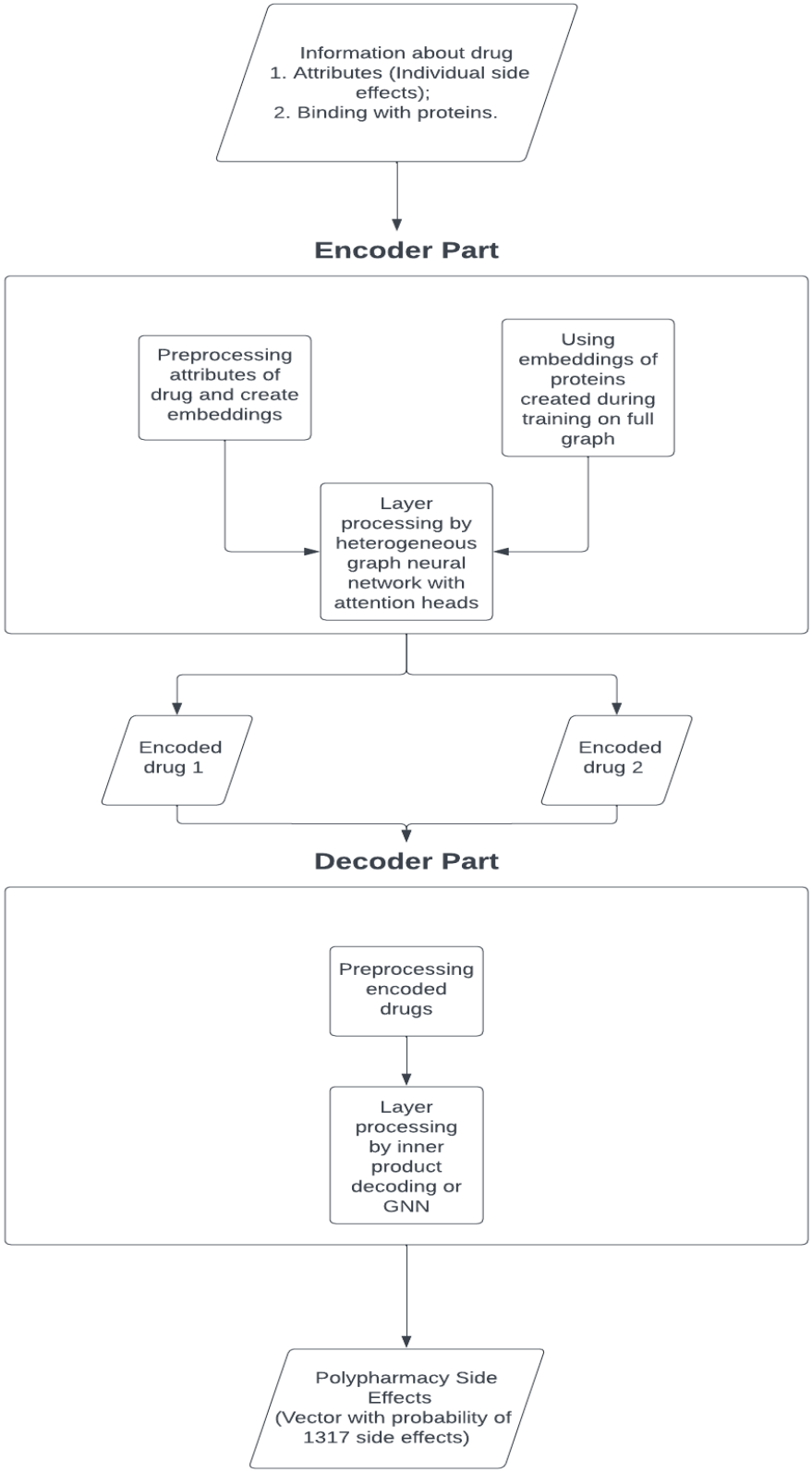


Рисунок 2.8 - Схема алгоритму графової мережі

У результаті ми матимемо вектор високої розмірності, який міститиме інформацію про можливі побічні ефекти. Описаний вектор зображений на рисунку 2.9.

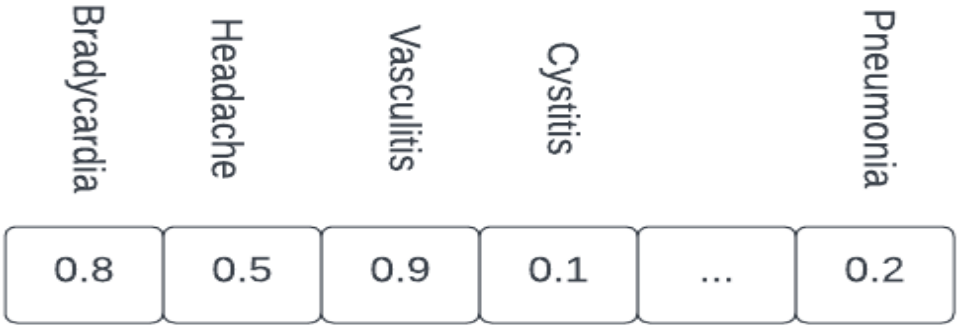


Рисунок 2.9 - Приклад згенерованого вектору, який описує побічні ефекти

Оскільки мережа вимагатиме великої кількості розрахунків, то для її створення, тренування та тестування використовуватимуться графічні процесори (GPU). Блоки обробки графіки, або графічні процесори, є спеціалізованими процесорами, спочатку призначеними для відтворення графіки та зображень. Однак завдяки своїй високопаралельній архітектурі вони стали інструментом для прискорення різноманітних розпаралелюваних обчислювальних завдань. Навчання нейронних мереж передбачає виконання численних матричних операцій, які за своєю суттю розпаралелювані, як зображено на рис. 2.10. На відміну від центральних процесорів (ЦП), які оптимізовані для послідовної обробки, графічні процесори перевершують паралельну обробку. Під час навчання параметри нейронної мережі коригуються за допомогою алгоритмів оптимізації. Графічні процесори прискорюють цей процес, як на рисунку, одночасно

обробляючи кілька точок даних і оновлюючи ваги, що значно скорочує час навчання.

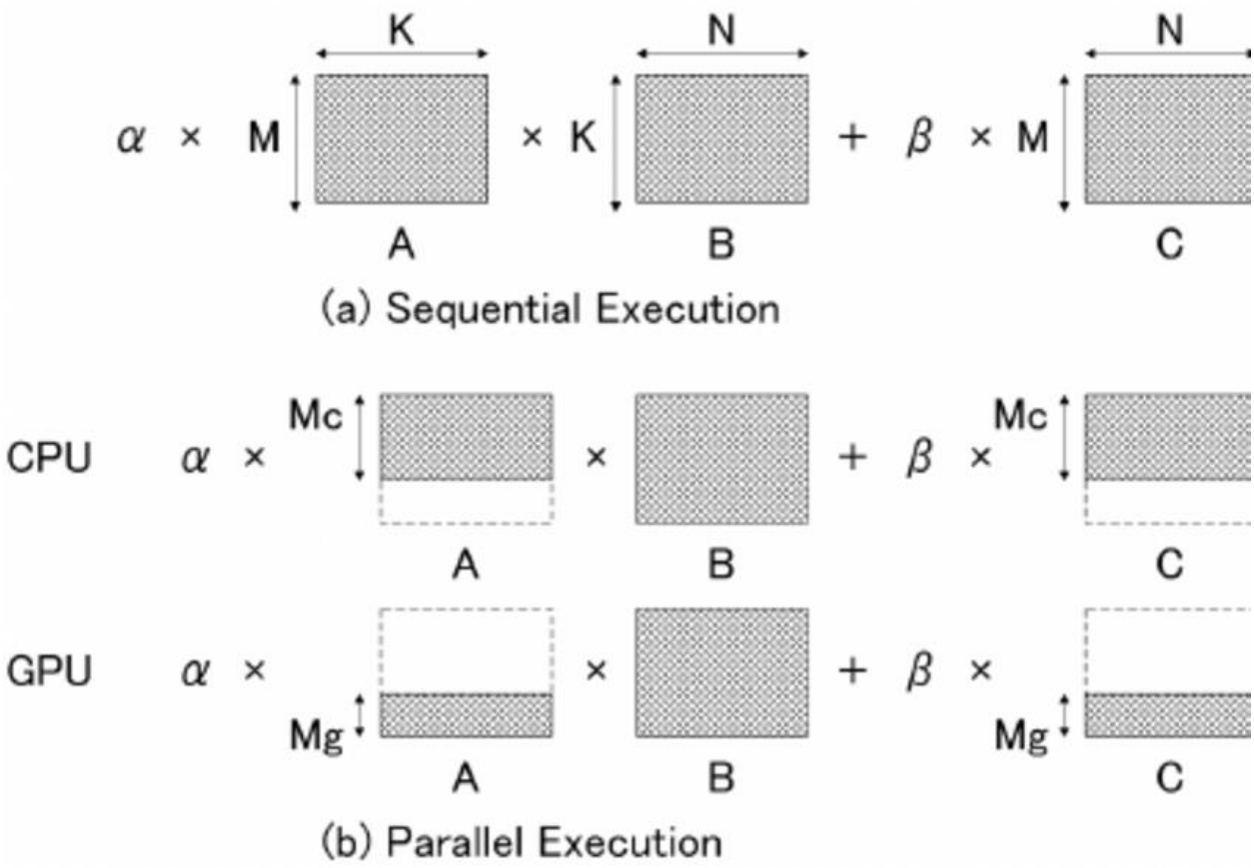


Рисунок 2.10 - Розпаралелювання перемноження матриць між девайсами

Не всі графічні процесори однакові; вони відрізняються за архітектурою, обчислювальною потужністю та об’ємом пам’яті. Графічні процесори високого класу оснащені тисячами ядер, що дозволяє виконувати паралельну обробку більшої кількості завдань. До популярних виробників графічних процесорів належать NVIDIA та AMD, а архітектура CUDA від NVIDIA широко використовується в програмах глибокого навчання.

Підсумовування впливу використання GPU під час тренування мережі:

- Графічні процесори відмінно справляються з паралельними завданнями, дозволяючи швидше виконувати матричні операції під час навчання нейронної мережі. Це призводить до значного скорочення часу навчання порівняно з використанням ЦП.
- Паралельна архітектура графічних процесорів дозволяє навчати більші та складніші моделі нейронних мереж. Моделі глибокого навчання з численними рівнями виграють від підвищеної потужності обробки.
- Фреймворки глибокого навчання, такі як TensorFlow і PyTorch, оптимізовані для використання можливостей GPU. Ця інтеграція оптимізує процес використання графічних процесорів для навчання нейронної мережі без необхідності масштабної оптимізації вручну.
- Графічні процесори пропонують економічно ефективне рішення для прискорення навчання нейронної мережі. Вони забезпечують вищу продуктивність за долар порівняно з ЦП, що робить їх ефективним вибором для дослідників і розробників.
- Кластери GPU можна використовувати для масштабованих завдань глибокого навчання, що дозволяє паралельно обробляти великі набори даних і складні моделі. Ця масштабованість має вирішальне значення для обробки різноманітних навантажень машинного навчання.
- Графічні процесори часто є більш енергоефективними для завдань глибокого навчання, забезпечуючи кращу продуктивність на ват порівняно з центральними процесорами. Це призводить до зниження енергоспоживання та експлуатаційних витрат.

Тому під час тренування мережі я використовував GPU NVIDIA Tesla T4. Завдяки 16 ГБ пам’яті GDDR6 типу та 2560 CUDA ядрам вона забезпечує підвищену продуктивність, що робить її ідеальним для програм глибокого навчання.

2.4 Висновок до розділу 2

У цьому розділі було проаналізовано різні архітектури графових нейронних мереж та обґрунтовано вибір для даної задачі гетерогенної графової мережі з блоками уваги. Розроблено структуру та математичну модель обраної архітектури – графової мережі з блоками уваги, яка має 128 входів для атрибутів препаратів та протеїнів, та вихідний шар, що містить 1317 нейронів для кожної побічної дії. Також розроблено структуру інформаційної технології прогнозування побічних ефектів поліпрагмазії.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ПРОГНОЗУВАННЯ ПОБІЧНИХ ЕФЕКТІВ ПОЛІПРАГМАЗІЇ

3.1 Обґрунтування вибору мови та середовища програмування

Для реалізації інформаційної технології передбачення побічних ефектів поліпрагмазії використовуватиметься мова програмування Python [38].

Python — це інтерпретована мова програмування високого рівня загального призначення. Він створений Guido van Rossum і вперше був випущений у 1991 році. Його дуже легко зрозуміти, з точки зору початківця, Python є найкращим вибором для початку. Він використовує інтерпретатор для перетворення мови високого рівня на мову низького рівня. Він став відомим завдяки своїм однорядковим пакетам, величезній кількості пакетів і активній спільноті розробників.

Python надає багато переваг для своїх розробників. Найкраща перевага використання інтерпретованої мови полягає в тому, що вони не залежать від платформи. На відміну від C++/C, де вам потрібно скомпілювати код, і після успішної компіляції створюється новий файл, який містить машинний код, який розуміє ЦП. У Python замість перекладу коду в машинний код код перекладається в байт-код.

Байт-код - це низькорівневий набір інструкцій, які можуть бути виконані інтерпретатором. Він виконується у віртуальній пам'яті.

Перерахую коротко переваги використання мови Python для програмування:

- Python працює на різних платформах (Windows, Mac, Linux, Raspberry Pi тощо).
- Python має простий синтаксис, подібний до англійської мови.
- Синтаксис Python дозволяє розробникам писати програми з меншою кількістю рядків, ніж деякі інші мови програмування.
- Python працює в системі інтерпретатора, тобто код може бути виконаний відразу після його написання. Це означає, що прототипування може бути дуже швидким.

- Python можна розглядати процедурно, об’єктно-орієнтовано або функціонально.

Python також є дуже популярним вибором для створення програм машинного навчання завдяки своїй універсальності, широким бібліотекам і фреймворкам, простоті використання, надійній підтримці спільноти, крос-платформній сумісності та бездоганній інтеграції з іншими технологіями. Його багата екосистема включає такі бібліотеки, як TensorFlow, Keras, PyTorch, scikit-learn тощо, спеціально створені для завдань машинного навчання, що спрощує процес розробки. Простий синтаксис і читабельність мови дозволяють розробникам стисло висловлювати складні ідеї, підвищуючи ясність коду та прискорюючи розробку проекту.

Процвітаюча спільнота Python науковців із обробки даних і практиків машинного навчання забезпечує достатні ресурси та підтримку для тих, хто наважується на машинне навчання. Кросплатформні можливості мови роблять її придатною для розробки додатків, які працюють на різних операційних системах, сприяючи широкій доступності. Python легко інтегрується з базами даних, фреймворками веб-розробки та інструментами візуалізації, що робить його адаптивним вибором для наскрізних рішень машинного навчання.

Попередня обробка та аналіз даних, які мають важливе значення для машинного навчання, отримують переваги від різноманітних бібліотек Python для обробки та обробки даних, таких як NumPy, pandas і SciPy. Хоча Python може бути не таким продуктивним, як мови нижчого рівня, він вирізняється швидким створенням прототипів і розробкою, а його продуктивність можна оптимізувати шляхом включення коду нижчого рівня, де це необхідно.

Активний розвиток Python гарантує, що він залишається в курсі останніх досягнень у машинному навчанні, дозволяючи практикам використовувати найновіші техніки та методології. Крім того, його природа з відкритим вихідним кодом сприяє доступності, оскільки його можна використовувати без витрат на ліцензування, що робить його привабливим вибором для широкого кола розробників та організацій.

Як середовище програмування обрано VS Code. Його популярність пояснюється розвитком галузі веб-розробки в ці роки та потребою розробників у легкому для роботи редакторі з невеликою кількістю функцій. Він також безкоштовний і розроблений корпорацією Майкрософт з використанням сучасного підходу з використанням Electron.

Серед його основних характеристик можна виділити:

- ***Open source.*** Безпрецедентною перевагою є той факт, що код VS здебільшого відкритий. Це означає не лише те, що програмне забезпечення безкоштовне для використання, але й те, що ви можете допомогти його покращити
- ***Простий у використанні.*** Від перших кроків до встановлення нових розширень, усе у VS Code здається простим та інтуїтивно зрозумілим. Завдяки розширюваній архітектурі (про яку ми поговоримо трохи пізніше), VS Code, навіть будучи лише редактором коду, може бути цінною альтернативою іншим складнішим IDE.
- ***Мінімальний дизайн.*** Дизайн, як правило, дуже суб’єктивна річ, але майбутнє є мінімальним, і VSCode вирішив прийняти принципи цього підходу до дизайну.
- ***Розширення.*** На VS Code marketplace буквально тисячі розширень.

Отже Visual Studio Code (VS Code) — це дуже ефективне середовище розробки на Python. Вищевказаний перелік переваг надає можливість швидкій структуризації проєкту, розробці та розгортання середовища для остаточного тестування результатів розробки.

3.2 Обгрунтування вибору фреймворку для реалізації інтелектуального модуля інформаційної технології

Оскільки графовий тип нейронних мереж значно відрізняється від звичайних, для спрощення роботи були створені окремі фреймворки для роботи з ними, які містять у собі багато компонентів для полегшення роботи та швидкої розробки.

Для написання коду архітектури мережі та налаштування пайплайн створення та обробки даних я використовуватиму PyG. [39]

PyG (PyTorch Geometric) — це бібліотека, створена на основі PyTorch для легкого написання та навчання графових нейронних мереж (GNN) для широкого спектру програм, пов’язаних зі структурованими даними. Бібліотека містить у собі реалізації різних методів глибокого навчання на графах та інших нерегулярних структурах, з різноманітних опублікованих статей. Окрім того, для роботи з даними вона надає доступ до простих у використанні міні-пакетних завантажувачів для роботи з багатьма малими та одиничними гігантськими графами, підтримки кількох графічних процесорів, підтримки torch.compile, підтримки DataPipe, великої кількості стандартних наборів даних (на основі простих інтерфейсів для створення власних наборів), менеджер експериментів GraphGym і корисні перетворення, як для навчання на довільних графах, так і на 3D-сітках або хмарах точок.

За допомогою цього фреймворка можна легко перевести дані з csv формату у графовий тип, який легко імплементуватимиться у пайплайн обробки даних, адже у даного типу даних є вбудовані інструменти для препроцесингу перед тренуванням.

Також даний фреймворк надає можливість створювати власні графові моделі зі своїх блоків або блоків, які можна створити власноруч згідно до своїх потреб. У

подальшому даний клас графової мережі буде мати усі можливості та інструменти, як вбудовані мережі через наслідування від основного класу.

3.3 Програмна реалізація прогнозування побічних ефектів поліпрагмазії

Для тренування мережі перш за все потрібно підготувати дані до спеціального вигляду, а також налаштувати загрузчик для тренувального процесу.

Також для створення зручної кодової інфраструктури з можливістю перевірити нові ідеї для аугментації базової архітектури графової мережі я використовуватиму репозиторій представлений нещодавно для поліпшення ведення обліку нових моделей та порівняння ефективності між собою. Запропоновану структуру порівняльного аналізу можна використовувати для тестування нових дослідницьких ідей на рівні попередньої обробки даних, покращення рівнів GNN і схем нормалізації або навіть для обґрунтування ефективності нової моделі GNN. На будь-якому етапі модульний компонент фреймворку, такий як дані, шари тощо, можна модифікувати, а численні експерименти з наборами даних можна проводити чесно та легко [7].

Для тренування моделі використовуватиметься спеціалізований набір даних підготовлений спеціалістами зі Stanford University та розміщений на SNAP. Stanford Network Analysis Platform (SNAP) — це високопродуктивна система загального призначення для аналізу та керування великими мережами розроблена Jure Leskovec під час аспірантури [6].

BioSNAP-Decagon набір даних складається з кількох сутностей, які описуються за допомогою гетерогенного графа. Серед них виділяють вершини графа, які містять інформацію про лікарський препарат або протеїн, який входить до складу рецепторів біологічних тканин. Основна дія препарату проявляється під час взаємодії з рецепторами та запуску каскаду біохімічних та фізіологічних процесів. Також у графа є ребра, що з’єднують вершини між собою, якщо вони

пов’язані хімічно-фізіологічним зв’язком: препарат – препарат, якщо виникає побічний ефект; препарат – протеїн, якщо препарат зв’язується з протеїном та викликає фізіологічну відповідь; протеїн – протеїн, якщо протеїни входять до складу одного сигнального ланцюга фізіологічної відповіді на взаємодію з рецептором [8].

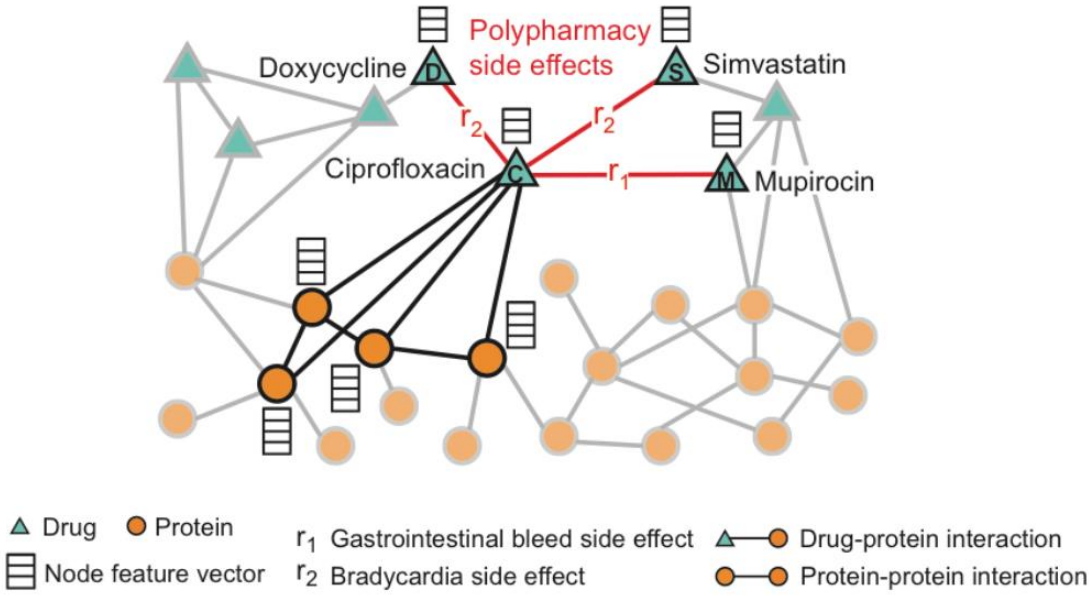


Рисунок 3.1 - Граф знань, який описує виникнення побічних ефектів від комбінації препаратів.

Загалом гетерогенний граф, зображений на рисунку 3.1, містить 645 ліків і 19 085 білкових вузлів, з’єднаних 715 612 білок-білок, 4 651 131 ліки-ліки та 18 596 ліки-білки зв’язками.

Загальний датасет для тренування складається з кількох ключових наборів даних для кожного зв’язку між сутностями.

Дані про побічні ефекти ліків представлені в наборі даних, зібраного з бази даних SIDER (Side Effect Resource) [40] і баз даних OFFSIDES і TWOSIDES [41].

Потім ці побічні ефекти класифікуються на дві групи: побічні ефекти препаратів та побічні ефекти поліфармацевтичної взаємодії між ліками. Побічні ефекти монопрепаратів — це побічні ефекти, які спостерігаються при застосуванні одного препарату з баз даних OFFSIDES і SIDER, тоді як побічні ефекти поліпрагмазії — це ті, які пов'язані із взаємодією пар ліків із баз даних TWOSIDES і SIDER. Приклад таблиці з даними зображений на рисунку 3.2.

Gene 1	Gene 2
114787	375519
114787	285613
114787	7448
114787	4914
114787	51343
114787	8089
114787	2775
114787	3184
114787	29785
114787	2781
114787	5793
114787	23528
114787	84548
114787	51150
114787	54825

Рисунок 3.2 - Приклад частини датасету взаємодії протеїнів

Далі я завантажив дані про виникнення побічних ефектів при використанні комбінації препаратів. Для відповідності порівняння результатів експерементів з встановленим еталоном я використовуватиму ту саму процедуру поділу даних, яку представили Zitnik et. al. [16], де розділяються побічні ефекти в наборі даних на дві групи відповідно до їх охоплення. Група високо представлених побічних ефектів

поліпрагмазії (понад 500 пов’язаних комбінацій ліків) розглядається для оцінки та тестування, тоді як інші низько представлені побічні ефекти побічні ефекти враховуються лише для навчання моделей.

Дані про взаємодію білків і ліків містять як взаємодію білок-білок, так і взаємодію препарат-білок. Білок-білкова взаємодія представляє сукупність фізичних взаємодій, експериментально оцінених на людях і зібраних з різних загальнодоступних джерел [43, 44, 45]. З іншого боку, взаємодію між ліками та білком отримано з бази даних STITCH [46]. Ці взаємодії позначають зв’язки між ліками та їх цільовими білками в організмі людини, що дозволяє препаратам активувати їхні очікувані терапевтичні ефекти та інші небажані ефекти (побічні ефекти). Приклад з набору даних продемонстрований на рисунку 3.3.

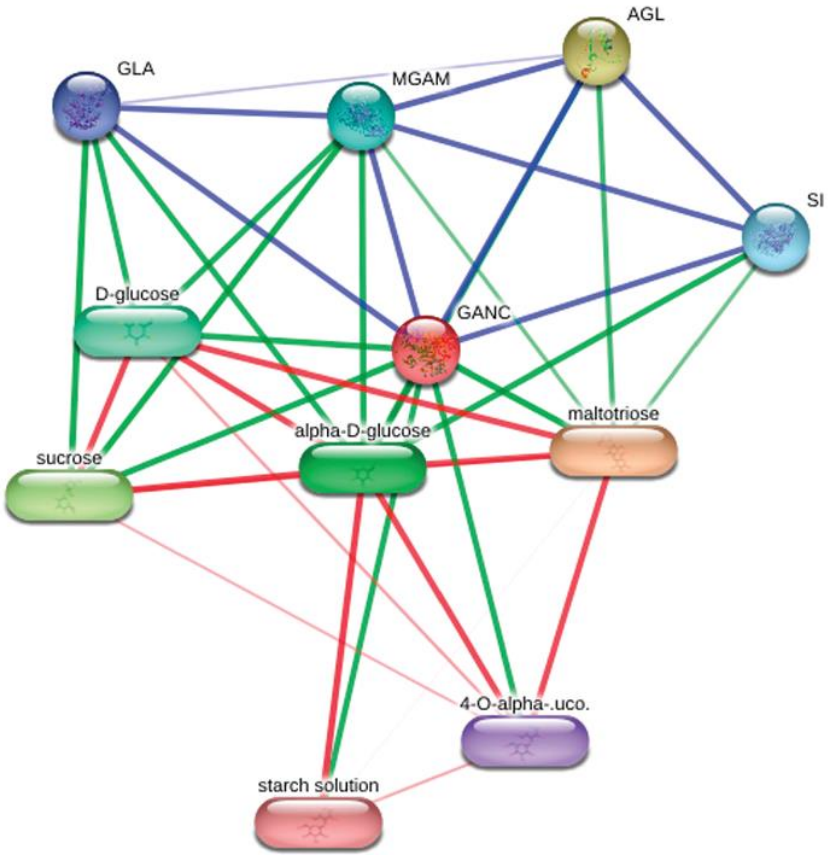


Рисунок 3.3 - Приклад графу з набору даних STITCH 3

Отже кожен файл представлений у csv форматі для зручності використання. Для перетворення у потрібний формат я використовуватиму вбудований клас гетерогенного графа, який зберігатиме кожен тип сутності або взаємодії як окремий тензор. Нижче наведений рисунок 3.4. отриманого гетерогенного графу.

```
HeteroData(
  protein={ node_id=[19089] },
  drug={
    node_id=[645],
    x=[639, 10184],
  },
  (protein, interconections, protein)={ edge_index=[2, 715612] },
  (drug, side_effects, drug)={
    edge_index=[2, 63473],
    edge_label=[63473, 1317],
  },
  (drug, bindings, protein)={ edge_index=[2, 18690] }
)
```

Рисунок 3.4 - Створений гетерогенний граф та збережений у зручній формі для роботи з ним

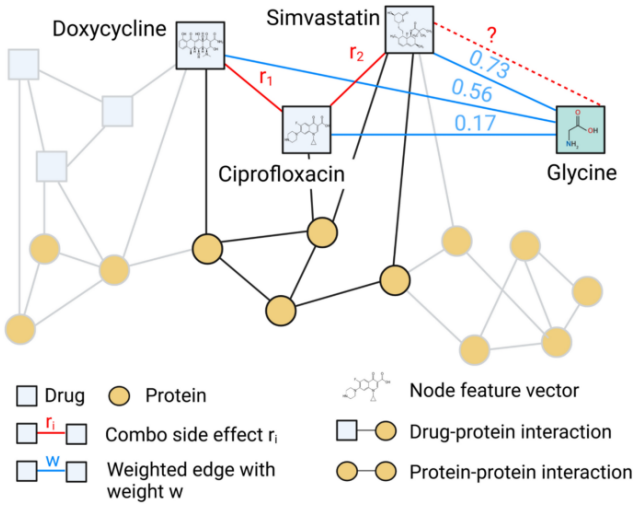


Рисунок 3.5 - Гетерогенний підграф з набору даних. Зображений для ілюстративної наглядності [47]

Як можна побачити з рисунка 3.4, у кожному тензорі є три основних матриці:

- data.x: матриця з корисними ознаками про сутність (вузол) розмірністю [кількість вузлів, кількість корисних ознак];
- data.edge_index: зв’язність графу у транзакційному форматі (COO) розмірністю [2, кількість ребер] та має формат типу torch.long;
- data.edge_attr: матриця корисних ознак ребер розмірністю [к-сть ребер, к-сть корисних ознак].

Оскільки ми маємо цілий датасет, його потрібно поділити на три основні частини, які використовуватимуться в ході тренування. Для цього використовуватиметься спеціальний метод поділу графа на основі ребер, який зображений на рисунку 3.6 та використовує властивість поділу графа, називається

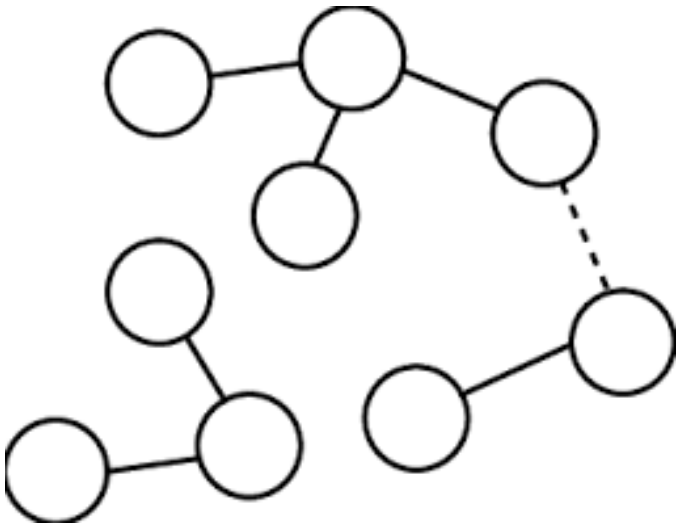


Рисунок 3.6 – Ілюстрація випадкового поділу ребер графа для розбиття графа на субграфи

RandomLinkSplit. Він виконує випадковий розподіл на рівні ребра об’єкта Data або HeteroData (функціональна назва: random_link_split). Поділ виконується таким чином, що навчальний поділ не включає ребра в перевірочних і тестових поділах; і перевірочний розділ не включає ребра в тестовому розділенні.

У вигляді коду це матиме наступний вигляд:

```
transform = T.RandomLinkSplit(
    num_val=0.1,
    num_test=0.1,
    disjoint_train_ratio=0.0,
    neg_sampling_ratio=0.0,
    add_negative_train_samples=False,
    edge_types=("drug", "side_effects", "drug"),
)

train_data, val_data, test_data = transform(data)
```

Оскільки набір даних великого розміру, який невміщатиметься в пам’ять GPU, я використовуватиму спеціальний метод завантаження даних пакетами (батчами). Фрагмент коду навединий нижче:

```
    edge_label_index = train_data["drug", "drug"].edge_label_index
    edge_label = train_data["drug", "drug"].edge_label
    train_loader = LinkNeighborLoader(
        data=train_data,
        num_neighbors=[20, 10],
        neg_sampling_ratio=2.0,
        edge_label_index=(("drug", "side_effects", "drug"), edge_label_index),
```

```
    edge_label=edge_label,
    batch_size=64,
    shuffle=True,
)
```

Передбачення наявності зв’язку у графі — це, по суті, двійкова класифікація, що передбачає наявність краю. Існуючі ребра графа вважаються позитивними прикладами, але нам також потрібно робити прогнози щодо негативних прикладів, ребер, яких насправді немає у графі. Для цього у вищевказаному блоці коду є параметр neg_sampling_ratio, який регулює створення додаткових негативних ребер для тренування.

Даний вид тренування за допомогою пакетів поліпшить результати роботи моделі за рахунок введення додаткової стохастичності в розрахунки, сприяючи регуляризації та запобіганні перенавчання.

Мережа складається з декількох основних блоків, які оброблятимуть вхідну інформацію послідовно, а саме кодувальника (encoder) та декодувальника (decoder).

До складу encoder входить HAN (гетерогенна графова нейронна мережа з увагою) та два шари для створення вбдуовувань з вхідних данних.

```
class HANEncoder(nn.Module):
    def __init__(self, in_channels: Union[int, Dict[str, int]],
                 hidden_channels=256, heads=8):
        super().__init__()
        self.han_conv = HANConv(hidden_channels, hidden_channels*2, heads=heads,
                                dropout=0.3, metadata=data.metadata(), add_self_loops=True)
        self.norm1_d = GraphNorm(hidden_channels)
        self.norm1_p = GraphNorm(hidden_channels)

    def forward(self, x_dict, edge_index_dict):
        out = self.han_conv(x_dict, edge_index_dict)
        out['drug'] = self.norm1_d(out['drug'])
        out['protein'] = self.norm1_p(out['protein'])

        return out
```

Варто зазначити, що оскільки атрибути (вхідну інформацію) мають препарати (індивідуальна побічна дія), то для протеїнів ми маємо тільки універсальну нумерацію, яку я створив попередньо під час підготовки набору даних до роботи. Тому було додатковий шар для створення вбудовувань відповідно до нумерації вершин, який під час тренування вивчав структурні закономірності у взаємодії між протеїнами, щоб охарактеризувати їх.

Наступним шаром створеної мережі був декодувальник.

```
class EdgeDecoder(nn.Module):
    def __init__(self, hidden_channels, output_channels):
        super().__init__()
        self.lin1 = Linear(4 * hidden_channels, hidden_channels)
        self.lin2 = Linear(hidden_channels, output_channels)

    def forward(self, z_dict, edge_label_index):
        row, col = edge_label_index
        z = torch.cat([z_dict['drug'][row], z_dict['drug'][col]], dim=-1)

        z = self.lin1(z).relu()
        z = self.lin2(z).sigmoid()

        return z
```

Основною задачею цього шару аналіз двох латентних векторів представлення кожного препарату, щоб визначити наявність одного з 1317 можливих зв’язків. Оскільки в більшості робіт для цього блоку використовували внутрішнє перемноження векторів, я прийняв рішення використати двохшаровий перцептрон, який повинен вивчити закономірність взаємодії між окремими розмірностями латентних векторів, щоб в кінці створити вектор розмірністю 1317, який відображатиме кожну парну побічну дію.

Загалом, коли з’єднати обидва блоки, то ми отримуємо повну мережу:

```
class Model(nn.Module):
    def __init__(self, hidden_channels, output_channels):
        super().__init__()
```

```
        self.encoder = HANEncoder(hidden_channels, hidden_channels)
        self.decoder = EdgeDecoder(hidden_channels, output_channels)

                self.protein_emb = torch.nn.Embedding(data["protein"].num_nodes, hidden_channels)
        self.drug_emb = torch.nn.Embedding(len(data["drug"].node_id), hidden_channels)

    def forward(self, data):
        x_dict = {
          "drug": self.drug_emb(data["drug"].node_id),
          "protein": self.protein_emb(data["protein"].node_id),
        }

        z_dict = self.encoder(x_dict, data.edge_index_dict)

        return self.decoder(z_dict, data['drug', 'side_effects', 'drug'].edge_label_index)
```

Дана мережа прийматиме на вхід матрицю розмірністю 2x10184 та на виході генеруватиме вектор розмірністю 1x1317. На рисунку 3.8 зображено схему з розмірностями даних на кожному шарі.

Layer	Input Shape	Output Shape	#Param
Model	[19728, 19728]	[50779, 1317]	2,896,677
├─(encoder)HANEncoder			135,168
│ └─(han_conv)HANConv			135,168
├─(decoder)EdgeDecoder	[2, 50779]	[50779, 1317]	235,557
│ └─(lin1)Linear	[50779, 512]	[50779, 128]	65,664
│ └─(lin2)Linear	[50779, 128]	[50779, 1317]	169,893
├─(protein_emb)Embedding	[19089]	[19089, 128]	2,443,392
├─(drug_emb)Embedding	[645]	[645, 128]	82,560

Рисунок 3.8. Схема створеної графової нейронної мережі, яка складається з двох блоків. Два блоки призначеня для виконання задачі прогнозування наявності зв’язку між двома препаратами

3.4 Висновки з розділу 3

У розділі розглянуто переваги використання мови програмування Python як об’єктно-орієнтованої під час розробки нейронних мереж. Також продемонстровано спеціалізовний фреймворк для розробки графових нейронних мереж PyTorch Geometric. Даний фреймворк полегшує задачу розробки мереж через надання готових класів усіх відомих графових мереж та пришвидшує розробку програмного забезпечення.

Продемонстровано набір даних для тренування нейронних мереж прогнозуванню побічних ефектів поліпрагмазії. Вказано, що набір даних для тренування складався з окремих датасетів, які містили інформацію про види окремих вузлів та ребра (взаємодію) між ними. У ході розробки показано, що мережа складається з двох основних блоків (кодувальника та декодувальника), які з’єднуються між собою та приймають на вхід матрицю розмірності 2x10184 генеруючи вектор побічних ефектів розмірністю 1x1317.

4 ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ ПРОГРАМИ ПРОГНОЗУВАННЯ ПОБІЧНИХ ЕФЕКТІВ ПОЛІПРАГМАЗІЇ

4.1 Процес тестування програми

Запустимо та протестуємо розроблену програму прогнозування побічних ефектів поліпрагмазії за допомогою графових нейронних мереж.

Початкове вікно програми показано на рисунку 4.1.

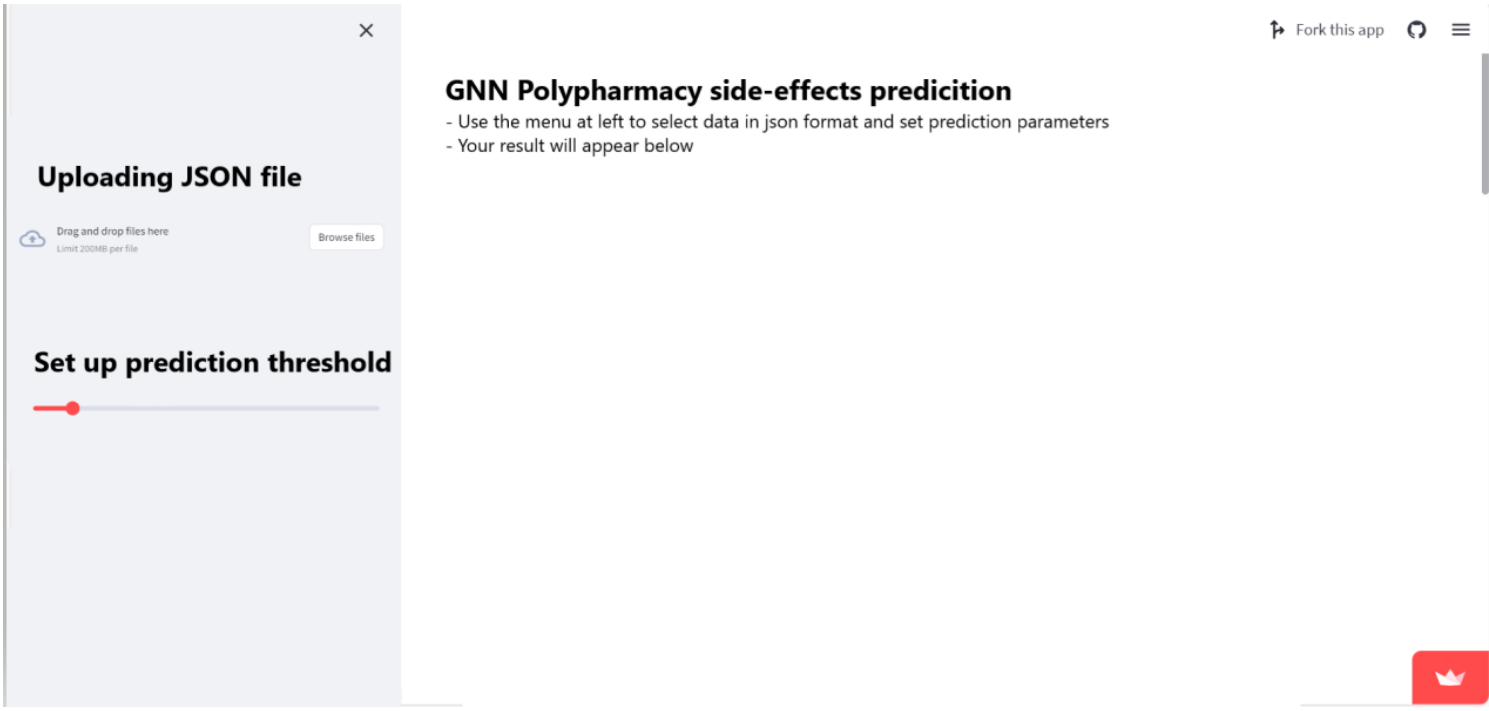


Рисунок 4.1 - Загальний вигляд web інтерфейсного вікна

Після отримання json файлу з бази даних для тестування, який попередньо був створений з наявного набору даних, я можу завантажити файл у програму для того, щоб отримати результат по можливим побічним ефектам.

Нижче наведено приклад завантаження файлу на рисунку 4.2 із виникненням загрузочного вікна програми.

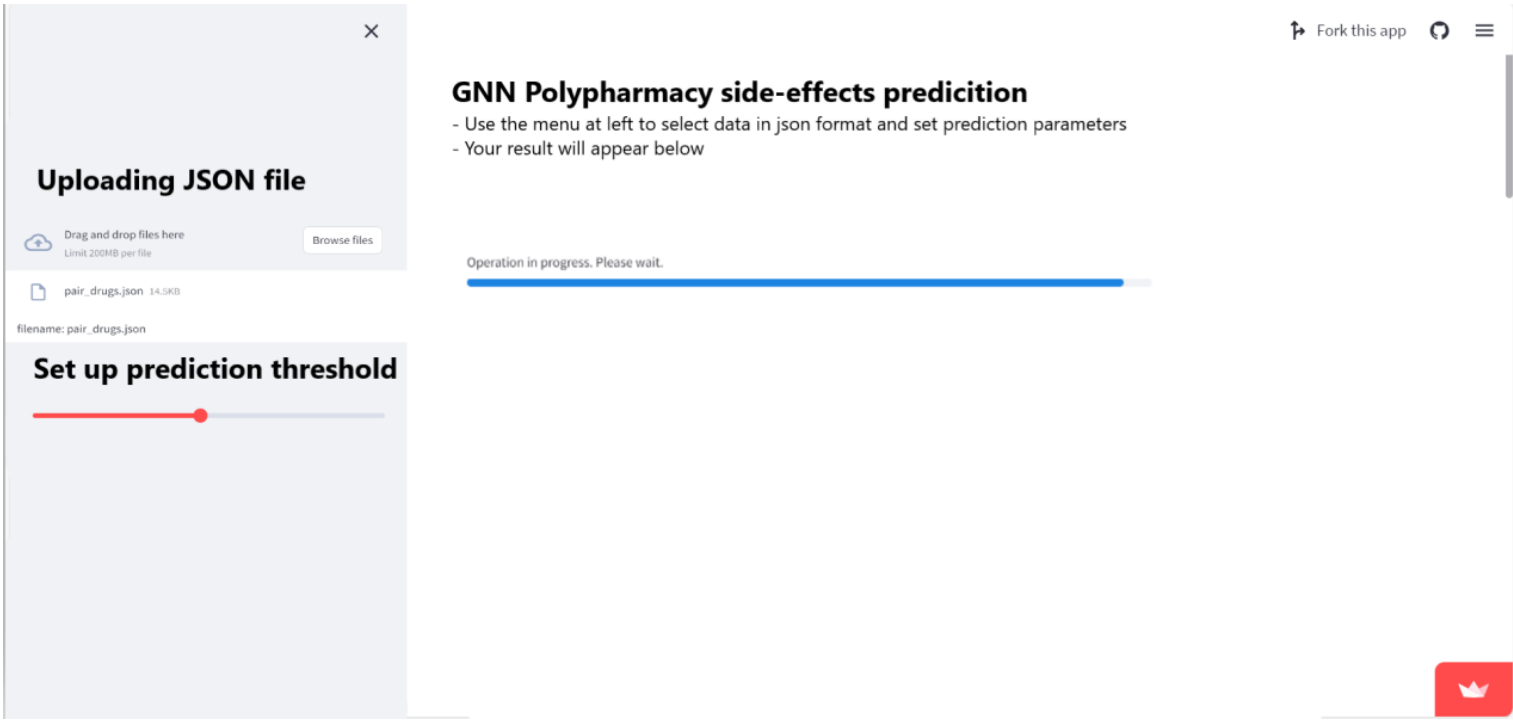


Рисунок 4.2 - Загальний вигляд web інтерфейсного вікна під час завантаження файла та генерації відповіді

Далі програма візуалізує вектор побічних дій відповідно до ймовірностей створюючи стовпчасту діаграму за допомогою фреймворка pandas, як зображено на рисунку 4.3.

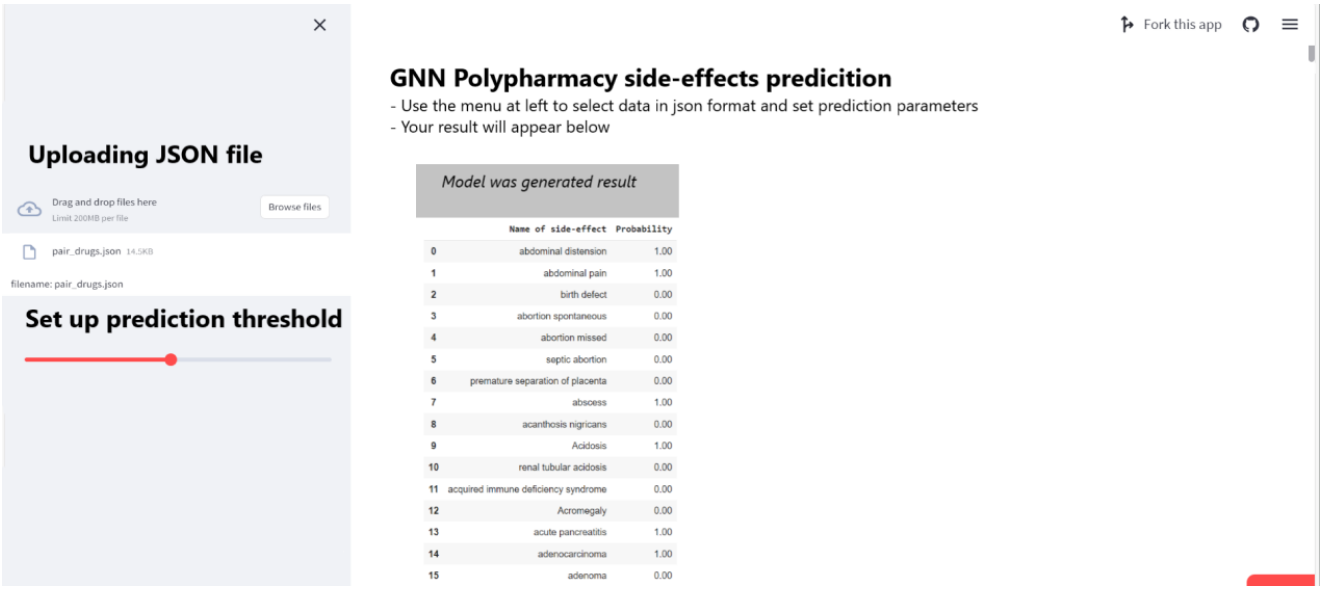


Рисунок 4.3. Результат передбачення моделі відображається у вигляді повноформатної таблиці в pandas з колонками “Назва побічної дії” та “Ймовірність”

4.2 Аналіз результатів роботи програми прогнозування побічних ефектів поліпрагмазії

Для доведення факту досягнення поставленої мети - збільшення точності прогнозування побічних ефектів поліпрагмазії, було виконано порівняння точності пронозування 10 побічних дій, які мають найкращу точність (таблиця 4.1).

Для визначення якості прогнозування використовувалася величина AUPRC. AURPC (площа під кривою precission-recall) вимірює точність передбачення поєднуючи три основних показники: true positives (TP), false positives (FP) та false negatives (FN). Даний показник використовується для оцінки результатів, якщо нам важливо оцінити наскільки модель добре виявляє TP. Для апроксимації цього показника використовується значення Average Precision, яке розраховується як зважене середнє значень precision на всіх трешхолдах.

Таблиця 4.1 - Порівняння найкращої точності

Best performing side effects	AUPRC
Mumps	0.982
Coccydynia	0.954
Carbuncle	0.951
Tympanic membrane perfor.	0.947
Dyshidrosis	0.943
Spondylosis	0.935
Schizoaffective disorder	0.922
Breast dysplasia	0.921
Ganglion	0.913
Uterine polyp	0.910

Для більш загального порівняння двох методі пропоную також виміряти три основних показники, які об’єктивно характеризують модель (таблиця 4.2.)

Таблиця 4.2 - Порівняння функціональності підходів за допомогою трьох основних критерії

Name of approach	AUROC	AUPRC	AP@50
Decagon	0.872	0.832	0.803
My approach	0.903	0.849	0.824

AP@50 розраховується як середнє значення precision по різним класам, коли з трешхолд взято 0.5 (все, що вище вважається позитивним класом – 1). AUROC є величиною подібною до AUPRC, проте для його розрахунку береться значення True Positive Rate при різних трешхолдах та все усереднюється.

Усі величини подані в десятковому варіанті та непереводилися у відсотки.

Як видно із таблиці 4.2, можна зробити висновок, що запропонований підхід досягнув кращих результатів у порівнянні з підходом аналогом по кільком критеріям.

Отже, після проведення порівняння розробленої програми прогнозування побічних ефектів поліпрагмазії за допомогою графової нейронної мережі, можна зробити висновок, що розроблена програма має вищу точність прогнозування побічних ефектів при поєднанні двох лікарських засобів.

4.3 Висновок до розділу 4

У даному розділі проведено тестування та демонстрацію веб інтерфейсу розробленої програми з прогнозуванням побічних ефектів комбінації двох препаратів. Також проведено кількісне порівяння точності прогнозування розрозробленої моделі з програмою-аналогом. Під час порівняння виявлено, що розроблена програма (AP@50 82.4%) має більшу точність прогнозування, ніж програма-аналог (AP@50 80.3%).

Тобто мета магістерської кваліфікаційної роботи досягнута – достовірність прогнозування побічних ефектів поліпрагмазії підвищена на 2.1%.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Проведення комерційного та технологічного аудиту інформаційної технології прогнозування побічних ефектів поліпрагмазії за допомогою графової нейронної мережі

Метою проведення комерційного і технологічного аудиту є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної під час виконання магістерської кваліфікаційної роботи.

Для проведення комерційного та технологічного аудиту я залучив 3-х незалежних експертів, якими є провідні викладачі випускової або спорідненої кафедр.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу здійснюється за допомогою п’ятибальної системи оцінювання за 12-ма критеріями, а результати демонструються у таблиці 4.1.

Таблиця 4.1 – Результати оцінювання науково-технічного рівня і комерційного потенціалу прогнозування побічних ефектів поліпрагмазії за допомогою графової нейронної мережі

Критерії	Експерти		
	Експерт 1	Експерт 2	Експерт 3
	Бали, виставлені експертами		
Технічна здійсненність концепції	4	4	5
Ринкові переваги (наявність аналогів)	4	3	3
Ринкові переваги (ціна продукту)	3	3	2
Ринкові переваги (технічні властивості)	4	3	4
Ринкові переваги (експлуатаційні витрати)	3	2	3

Ринкові перспективи (розмір ринку)	3	3	4
Ринкові перспективи (конкуренція)	4	3	3
Практична здійсненність (наявність фахівців)	4	4	3
Практична здійсненність (наявність фінансів)	3	3	3
Практична здійсненність (необхідність нових матеріалів)	5	4	3
Практична здійсненність (термін реалізації)	4	3	3
Практична здійсненність (розробка документів)	3	3	3
Сума балів	44	41	39
Середньоарифметична сума балів, СБ	41		

За результатами розрахунків відносно даних наведених в таблиці 4.1 можна зробити висновок, що науково-технічний рівень та комерційний потенціал інформаційної технології прогнозування побічних ефектів поліпрагмазії за допомогою графової нейронної мережі – високий.

5.2 Розрахунок витрат на здійснення науково-дослідної роботи

Витрати на оплату праці. До даного типу виплат належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим працівникам, технологам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці, також будь-які

види грошових і матеріальних доплат, які належать до елемента «Витрати на оплату праці».

Основна заробітна плата дослідників. Витрати на основну заробітну плату дослідників ($З_о$) розраховують відповідно до їх посадових окладів, за формулою:

$$З_о = \sum_{i=1}^{к} \frac{М_{ni} \cdot t_i}{Т_р}, \quad (4.1)$$

де к – кількість посад дослідників, залучених до процесу дослідження; $М_{ni}$ – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.; $Т_р$ – число робочих днів в місяці; приблизно $Т_р$ = (21…23) дні; t_i – число робочих днів роботи розробника (дослідника).

Зроблені розрахунки зведені у таблиці 4.2.

Таблиця 4.2 – Витрати на заробітну плату дослідників

Посада	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник	80000	3636	180	654480
Розробник	40000	1818	150	272700
Всього:	927180			

Основна заробітна плата робітників. Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт розраховують за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i, \quad (4.2)$$

де $С_і$ – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год; $t_і$ – час роботи робітника на виконання певної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду С розраховують за формулою:

$$С_і = \frac{М_м \cdot К_і \cdot К_с}{Т_р \cdot t_{зм}}, \quad (4.3)$$

де $М_м$ – розмір прожиткового мінімуму працездатної особи або мінімальної місячної заробітної плати (залежно від діючого законодавства), у 2023 році $М_м$=6700 грн; $К_і$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду; $К_с$ – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати; $Т_р$ – середня кількість робочих днів в місяці, приблизно $Т_р$ = 21…23 дні; $t_{зм}$ – тривалість зміни, год.

Таблиця 4.3 – Витрати на заробітну плату робітників

Найменування робіт	Трудомісткість, н-год.	Розряд роботи	Погодинна тарифна ставка	Тариф. коеф.	Величина, грн.
Створення графової моделі передбачення побічних ефектів поліпрагмазії	220	6	55,2	1,45	12144
Створення браузерного додатку	100	5	51,77	1,36	5177
Створення дизайну додатку	80	4	48,3	1,27	3864
Менеджмент проєкту	200	5	51,8	1,36	10360
Спілкування з інвесторами	80	5	51,8	1,36	4144
Всього					35689

Додаткова заробітна плата. Додаткова заробітна плата $З_д$ всіх розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховується як (10…12)% від суми основної заробітної плати всіх розробників та робітників, тобто:

$$З_д = 0{,}1 \cdot (З_о + З_р) = 0{,}1 \cdot (927180 + 35689) = 96286{,}9 \text{ грн.} \quad (4.4)$$

Відрахування на соціальні заходи. Нарахування на заробітну плату $Н_{зп}$ розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою:

$$Н_{зп} = \beta \cdot (З_о + З_р + З_д) =$$
$$= 0{,}22 \cdot (927180 + 35689 + 96286{,}9) = 233014{,}3 \text{ грн.} \quad (4.5)$$

де $З_о$ – основна заробітна плата розробників, грн.; $З_р$ – основна заробітна плата робітників, грн.; $З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.; β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % (приймаємо для 1-го класу професійності ризику 22%).

Програмне забезпечення. Для розрахунку балансової вартості програмного забезпечення необхідно враховувати витрати на його інсталяцію, тому ці витрати беруться додатково в розмірі 10…12% від вартості програмного забезпечення. Балансову вартість програмного забезпечення розраховують за формулою:

$$В_{прг} = \sum_1^к Ц_{іпрг} \cdot С_{прг.і} \cdot К_і, \quad (4.6)$$

де $Ц_{іпрг}$ – ціна придбання програмного забезпечення і-го виду, грн.; $С_{прг.і}$ – кількість одиниць програмного забезпечення відповідного виду, шт.; $К_і$ – коефіцієнт, що враховує інсталяцію, налагодження програмного забезпечення, $К_і$ = (1,1…1,12); к – кількість видів програмного забезпечення.

Таблиця 4.4 – Витрати на придбання програмного забезпечення

Найменування програмного забезпечення	Ціна за одиницю, грн.	Витрачено	Вартість програмного забезпечення, грн.
Google Colab	407	18	7236
Всього, з врахуванням коефіцієнта інсталяції та налагодження			8205

Амортизація обладнання. Амортизація обладнання, комп'ютерів та приміщень, які використовувались під час (чи для) виконання даного етапу роботи.

У спрощеному вигляді амортизаційні відрахування А в цілому бути розраховані за формулою:

$$А = \frac{Ц_б}{Т_в} \cdot \frac{t}{12}, \quad (4.7)$$

де $Ц_б$ – загальна балансова вартість всього обладнання, комп'ютерів, приміщень тощо, що використовувались для виконання даного етапу роботи, грн.; t – термін використання основного фонду, місяці; $Т_в$ – термін корисного використання основного фонду, роки.

Таблиця 4.5 – Амортизаційні відрахування за видами основних фондів

Найменування	Балансова вартість, грн.	Строк корисного використання, років	Термін використання, місяців	Сума амортизації, грн.
Ноутбук Macbook Pro 2019	35000	5	4	2333
WI-FI Роутер	1500	3	4	166
Всього	2499			

Витрати на електроенергію для науково-виробничих цілей. Витрати на електроенергію $В_е$ розраховуються за формулою:

$$Ве = \sum \frac{W_i \cdot t_i \cdot Ц_е \cdot К_{впі}}{ККД} = \frac{0{,}38 \cdot 1500 \cdot 7{,}5 \cdot 0{,}95}{0{,}98} + \frac{0{,}01 \cdot 1600 \cdot 7{,}5 \cdot 0{,}95}{0{,}98} = 4260{,}5 \text{ грн.}, \quad (4.8)$$

де W_i – встановлена потужність обладнання, кВт; t_i – тривалість роботи обладнання на етапі дослідження, год.; Це – вартість 1 кВт електроенергії, грн.; $К_{впі}$ – коефіцієнт використання потужності; ККД – коефіцієнт корисної дії обладнання.

Таблиця 4.6 – Витрати на електроенергію

Найменування обладнання	Потужність, кВт	Тривалість годин роботи
Ноутбук Macbook Pro 2019	0,38	1500
WI-FI Роутер	0,01	1600

Інші витрати. До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$Ів = \left(З_о + З_р\right) \cdot \frac{Н_{ів}}{100\%} = (927180 + 35689) \cdot \frac{50}{100} = 481434{,}5 \text{ грн.}, \qquad (4.9)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

Накладні (загальновиробничі) витрати. До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%} =$$

$$(927180 + 35689) \cdot \frac{110}{100} = 1059155{,}9 \text{ грн.}, \qquad (4.10)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

Витрати на проведення науково-дослідної роботи. Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + Н_{зп} + В_{прг} + А_{обл} + В_е +$$

$$+ І_в + В_{нзв} = 927180 + 35689 + 96317{,}3 + 233014{,}3 + 8205 + 2499 + 4260{,}5 +$$

$$481434{,}5 + 1059155{,}9 = 2847755{,}5 \text{ грн.} \qquad (4.11)$$

Загальні витрати. Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} = \frac{2847755{,}5}{0{,}7} = 4068222{,}14 \text{ грн.}, \qquad (4.12)$$

де η – коефіцієнт, що характеризує етап виконання науково-дослідної роботи. Оскільки, якщо науково-технічна розробка знаходиться на стадії розробки промислового зразка, то η=0,7.

5.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

У ринкових умовах позитивним результатом, який отримає потенційний інвестор від можливого впровадження результатів науково-технічної розробки, є збільшення величини чистого прибутку.

В даному випадку відбувається розробка програмного забезпечння, тому основу майбутнього економічного ефекту буде формувати: ΔN – збільшення кількості споживачів, яким надається відповідна інформаційна послуга в аналізовані періоди часу; N – кількість споживачів, яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки; $Ц_б$ – вартість послуги у році до впровадження інформаційної системи; $\pm\Delta Ц_о$ – зміна вартості послуги (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу.

Можливе збільшення чистого прибутку у потенційного інвестора ΔПі для кожного із років, протягом яких очікуються позитивні результати від можливого впровадження та комерціалізації науково-технічної розробки, розраховується за формулою:

$$\Delta П = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N_і)_і \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\vartheta}{100}\right), \qquad (4.13)$$

де ±ΔЦ – зміна основного якісного показника від впровадження результатів науково-технічної розробки в аналізованому році. Зазвичай, таким показником може бути зміна ціни реалізації одиниці нової розробки в аналізованому році (відносно року до впровадження цієї розробки); $\pm\Delta Ц_о$ може мати як додатне, так і від'ємне значення (від'ємне – при зниженні ціни відносно року до впровадження цієї розробки, додатне – при зростанні ціни); N – основний кількісний показник, який визначає величину попиту на аналогічні чи подібні розробки у році до

впровадження результатів нової науково-технічної розробки; $Ц_о$ – основний якісний показник, який визначає ціну реалізації нової науково-технічної розробки в аналізованому році; $Ц_б$ – основний якісний показник, який визначає ціну реалізації існуючої (базової) науково-технічної розробки у році до впровадження результатів; ΔN – зміна основного кількісного показника від впровадження результатів науково-технічної розробки в аналізованому році. Зазвичай таким показником може бути зростання попиту на науково-технічну розробку в аналізованому році (відносно року до впровадження цієї розробки); λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2023 році ставка податку на додану вартість становить 20%, а коефіцієнт λ =0,8333; ρ – коефіцієнт, який враховує рентабельність інноваційного продукту (послуги). Рекомендується брати ρ=0,2...0,5; ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році ϑ =18%.

Очікуваний термін життєвого циклу розробки 1 рік, тому:

$$\Delta П = ((300000 - 100000) \cdot 25000 - (30000 - 10000) \cdot 120000) \cdot 0{,}8333 \cdot 0{,}3 \cdot \left(1 - \frac{18}{100}\right) = 532978680 \text{ грн.} \quad (4.14)$$

Далі розраховують приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t} = \frac{532978680}{(1+0{,}1)^1} = 484526073 \text{ грн.}, \quad (4.15)$$

де $\Delta П$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн.; T – період часу, протягом якого очікується отримання позитивних результатів від

впровадження та комерціалізації науково-технічної розробки, роки (приймаємо Т=1 рік); τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,05…0,15; t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

Далі розраховують величину початкових інвестицій PV , які потенційний інвестор повинен вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати наступну формулу:

$$PV = k_{\text{інв}} \cdot \text{ЗВ} = 5 \cdot 4069456{,}4 = 20347282 \text{ грн.} \quad (4.16)$$

де $k_{\text{інв}}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. До них відносяться витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай kінв=2...5, але може бути і більшим; ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

Тоді абсолютний економічний ефект $E_{\text{абс}}$ або чистий приведений дохід для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{\text{абс}} = \text{ПП} - PV = 484526073 - 20347282 = 464178791 \text{ грн.}, \quad (4.17)$$

де ПП – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, грн.; PV – теперішня вартість початкових інвестицій, грн.

Оскільки $E_{\text{абс}}>0$, то можемо припустити про потенційну зацікавленість інвесторів у розробці.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність $Е_в$ або показник внутрішньої норми дохідності вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Внутрішня економічна дохідність інвестицій Ев, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Еабс}{PV}} = \sqrt[1]{1 + \frac{464178791}{20347282}} = 4{,}88, \quad (4.18)$$

де $Т_ж$ – життєвий цикл розробки, роки.

Далі потрібно визначити бар'єрну ставку дисконтування $\tau_{мін}$, а саме мінімальну внутрішню економічну дохідність інвестицій, нижче якої кошти у впровадження науково-технічної розробки та її комерціалізацію вкладатися не будуть.

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$ розраховується за формулою:

$$\tau_{мін} = d + f = 0{,}9 + 0{,}05 = 0{,}95, \quad (4.19)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,9...0,12; f – показник, що характеризує ризикованість вкладення інвестицій; зазвичай величина f =0,05...0,5, але може бути і значно вищою.

Оскільки показник $E_в$=4,88>$\tau_{мін}$=0,95, то потенційний інвестор може бути зацікавлений у фінансуванні впровадження науково-технічної розробки та виведенні її на ринок, тобто в її комерціалізації.

Далі розраховується період окупності інвестицій $T_о$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_о = \frac{1}{E_в} = \frac{1}{4,88} = 0,2 \text{ року}. \quad (4.20)$$

Оскільки $T_о$=0,2<1 року, то це свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження цієї розробки та виведення її на ринок.

5.4 Висновок до розділу 5

У розділі було виконано розрахунок витрат на розробку та виготовлення нового технічного рішення, сума яких складає 2847755,5 гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розаховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, знайдено термін окупності витрат для виробника та економічний ефект для споживача при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розробка у виробництві та використання дешевша за аналог і є висококонкурентоспроможним. Період окупності складе близько 0,2 року.

ВИСНОВКИ

Під час виконання магістерської кваліфікаційної роботи на ступінь магістра вирішено завдання розробки інформаційної технології та програмного забезпечення для прогнозування побічних ефектів поліпрагмазії.

Серед виконаних завдань під час виконання магістерської кваліфікаційної роботи було проведено аналіз методів прогнозування побічних ефектів поліпрагмазії у різноманітних джерелах інформації, що охоплюють книги, публікації на конференціях тощо. Також було обгрунтувано вибір алгоритму для розв’язку поставленого завдання на основі графової нейронної мережі та виконано проектування інтелектуальної системи прогнозування побічних ефектів поліпіпрагмазії на основі неї, що зумовлено демонстрацією мережею найкращих результатів на подібних задачах прогнозування зв’язку між кількома сутностями. Було проведено тестування інтелектуальної технології прогнозування побічних ефектів поліпіпрагмазії за допомогою графової нейронної мережі та аналіз результатів на основі окремого тестового набору даних, який невикористовувався під час тренування. Як результат отримано поліпшення прогнозування побічних ефектів поліпрагмазії (AP@50 82.4%) відносно програми-аналога (AP@50 80.3%). Достовірність прогнозування побічних ефектів поліпрагмазії підвищена на 2.1%.

Інформаційна технологія може бути покращена додатковою обробкою даних, оскільки виявлено нерелевантні класи для побічних дій та можливість витоку інформації між наборами даних під час тренування; заміна моделі в декодувальнику на більш складну, наприклад вставки графа знань (KGE) та пришвидшення обробки даних на GPU за допомогою додаткових фреймворків.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кузняк В. П., Колесницький О. К. «Інформаційна технологія прогнозування побічних ефектів поліпрагмазії за допомогою графової нейронної мережі», в Матеріали конференції «Молодь в науці: дослідження, проблеми, перспективи (МН2024)», Вінниця, 2023, [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/19595.
2. Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. 2018. Deepinf: Social influence prediction with deep learning. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. Association for Computing Machinery, New York, NY, USA, 2110–2119.
3. Ahmedt-Aristizabal D, Armin MA, Denman S, Fookes C, Petersson L. Graph-Based Deep Learning for Medical Diagnosis and Analysis: Past, Present and Future. Sensors (Basel). 2021 Jul 12;21(14):4758.
4. S. Rahmani, A. Baghbani, N. Bouguila and Z. Patterson, "Graph Neural Networks for Intelligent Transportation Systems: A Survey," in IEEE Transactions on Intelligent Transportation Systems, vol. 24, no. 8, pp. 8846-8885, Aug. 2023.
5. Dwivedi, Vijay Prakash et al. “Benchmarking Graph Neural Networks.” J. Mach. Learn. Res. 24 (2023): 43:1-43:48.

6. http://snap.stanford.edu/snap/index.html

7. Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, Xavier Bresson (2022) Benchmarking Graph Neural Networks.

8. Szklarczyk D. et al. (2016) STITCH 5: augmenting protein–chemical interaction networks with tissue and affinity data. *Nucleic Acids Res*., 44, D380–D384.

9. Kuhn M, Al Banchaabouchi M, Campillos M, Jensen LJ, Gross C, Gavin AC, Bork P. Systematic identification of proteins that elicit drug side effects. Mol Syst Biol. 2013;9:663.

10. Li, X. et al. (2017). Prediction of synergistic anti-cancer drug combinations based

on drug target network and drug induced gene expression profiles. Artificial Intelligence in Medicine.

11. Calabrese EJ. The Emergence of the Dose-Response Concept in Biology and Medicine. Int J Mol Sci. 2016 Dec 5;17(12):2034.

12. Vilar S, Friedman C, Hripcsak G (2018) Detection of drug–drug interactions through data mining studies using clinical sources, scientific literature and social media. Brief Bioinform 19:863–877.

13. Zong N, Kim H, Ngo V, Harismendy O. Deep mining heterogeneous networks of biomedical linked data to predict novel drug-target associations. Bioinformatics. 2017 Aug 1;33(15):2337-2344.

14. Perozzi B. et al. (2014) Deepwalk: Online learning of social representations. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. pp. 701–710

15. Cheng F. et al. (2012) Prediction of drug–target interactions and drug repositioning via network-based inference. *PLoS Comput. Biol*., 8, e1002503.

16. Zitnik M, Agrawal M, Leskovec J. Modeling polypharmacy side effects with graph convolutional networks. Bioinformatics. 2018 Jul 1;34(13):i457-i466.

17. Nováček V, Mohamed SK. Predicting Polypharmacy Side-effects Using Knowledge Graph Embeddings. AMIA Jt Summits Transl Sci Proc. 2020 May 30;2020:449-458.

18. Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous Graph Attention Network. In The World Wide Web Conference (WWW '19). Association for Computing Machinery, New York, NY, USA, 2022–2032.

19. Zhang, J., Shi, X., Xie, J., Ma, H., King, I., & Yeung, D.Y. (2018). GaAN: Gated Attention Networks for Learning on Large and Spatiotemporal Graphs. Conference on Uncertainty in Artificial Intelligence.

20. Bondy, J. A., & Murty, U. S. R. (2008). Graph theory. New York: Springer.

21. R. Vandaele, Y. Saeys and T. De Bie, "Graph Approximations to Geodesics on Metric Graphs," 2020 25th International Conference on Pattern Recognition (ICPR), Milan, Italy, 2021, pp. 7328-7334.

22. Jian Tang, Jingzhou Liu, Ming Zhang, and Qiaozhu Mei. 2016. Visualizing Large-scale and High-dimensional Data. In Proceedings of the 25th International Conference on World Wide Web (WWW '16). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 287–297.

23. Zhang, S., Tong, H., Xu, J. et al. Graph convolutional networks: a comprehensive review. Comput Soc Netw 6, 11 (2019).

24. Kipf, T.N., Welling, M., 2017. Semi-supervised classification with graph convolutional networks. Proceedings of ICLR.

25. David I Shuman, Sunil K. Narang, Pascal Frossard, Antonio Ortega, Pierre Vandergheynst. The Emerging Field of Signal Processing on Graphs: Extending High-Dimensional Data Analysis to Networks and Other Irregular Domains. IEEE Signal Processing Magazine; 2013 Mar 10.

26. Mallat, S., 1999. A Wavelet Tour of Signal Processing. Elsevier.

27. Henaff, M., Bruna, J., Lecun, Y., 2015. Deep Convolutional Networks on Graph-Structured Data. arXiv preprint arXiv:1506.05163.

28. Hamilton, W.L., Ying, Z., Leskovec, J., 2017a. Inductive representation learning on large graphs. In: Proceedings of NIPS, pp. 1024–1034.

29. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y., 2018. Graph attention networks. In: Proceedings of ICLR.

30. Bahdanau, D., Cho, K., Bengio, Y., 2015. Neural machine translation by jointly learning to align and translate. In: Proceedings of ICLR.

31. Cheng, J., Dong, L., Lapata, M., 2016. Long short-term memory-networks for machine reading. In: Proceedings of EMNLP, pp. 551–561.

32. Vaswani, A., Shazeer, N., Parmar, N., Jones, L., Uszkoreit, J., Gomez, A.N., Kaiser, L., 2017. Attention is all you need. In: Proceeding of NIPS, pp. 5998–6008.

33. Zhang, J., Shi, X., Xie, J., Ma, H., King, I., Yeung, D.-Y., 2018c. Gaan: gated attention networks for learning on large and spatiotemporal graphs. In: Proceedings of UAI.

34. Sun, R. Barber, M. Gupta, C. C. Aggarwal, and J. Han.Co-author relationship prediction in heterogeneousbibliographic networks. In ASONAM, pages 121–128, 2011.

35. Milo R, Shen-Orr S, Itzkovitz S, Kashtan N, Chklovskii D, Alon U. Network motifs: simple building blocks of complex networks. Science. 2002 Oct 25;298(5594):824-7.

36. Huang, Zhipeng et al. "Meta Structure: Computing Relevance in Large Heterogeneous Information Networks." Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2016)

37. Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous Graph Attention Network. In The World Wide Web Conference (WWW '19). Association for Computing Machinery, New York, NY, USA, 2022–2032.

38. https://www.python.org/doc/

39. https://pyg.org/

40. https://pytorch-geometric.readthedocs.io/en/latest/

41. Kuhn Michael, Letunic Ivica, Jensen Lars Juhl, Bork Peer. The sider database of drugs and side effects. Nucleic acids research. 2016;44(D1):D1075–9.

42. Tatonetti Nicholas P., Ye Patrick, Daneshjou Roxana, Altman Russ B. Data-driven prediction of drug effects and interactions. Science translational medicine. 2012;4(125):125ra31.

43. Menche Jörg, Sharma Amitabh, Kitsak Maksim, Ghiassian Susan Dina, Vidal Marc, Loscalzo Joseph, Barabási Albert-László. Uncovering disease-disease relationships through the incomplete interactome. Science. 2015:347.

44. Chatraryamontri Andrew, Breitkreutz Bobby-Joe, Oughtred Rose, Boucher Lorrie, Heinicke Sven, Chen Daici, Stark Chris, Breitkreutz Ashton, Kolas Nadine, O'Donnell Lara, Reguly Teresa, Nixon Julie, Ramage Lindsay, Winter Andrew G., Sellam Adnane, Chang Christie S., Hirschman Jodi E., Theesfeld Chandra L., Rust Jennifer M., Livstone

Michael S., Dolinski Kara, Tyers Mike. The biogrid interaction database: 2015 update. Nucleic Acids Research. 2015

45. Rolland Thomas, Tasan Murat, et al. A proteome-scale map of the human interactome network. Cell. 2014;159:1212–1226.

46. Szklarczyk Damian, Santos Alberto, von Mering Christian, Jensen Lars Juhl, Bork Peer, Kuhn Michael. Stitch 5: augmenting protein–chemical interaction networks with tissue and affinity data. Nucleic Acids Research. 2016

47. Lukashina, N., Kartysheva, E., Spjuth, O. et al. SimVec: predicting polypharmacy side effects for new drugs. J Cheminform 14, 49 (2022).

Додаток А (обов'язковий)
Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

Назва роботи: Нейромережева інформаційна прогнозування побічних ефектів поліпрагмазії за допомогою графової нейронної мережі

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 95,69% Схожість 4,31%

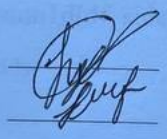
Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи  Кузняк В.П.

Керівник роботи Колесницький О.К.

Опис прийнятого рішення

Магістерську кваліфікаційну роботу допущено до захисту

Особа, відповідальна за перевірку  Озеранський В.С.

Додаток Б (обов’язковий)

Лістинг програми

preprocessing_data.py

```
import numpy as np
import torch
from torch_geometric.data import HeteroData
import torch_geometric.transforms as T
import pandas as pd

df_ppi = pd.read_csv('bio-decagon-ppi.csv')
df_side_effects = pd.read_csv('bio-decagon-combo.csv')
df_binding = pd.read_csv('bio-decagon-targets.csv')
df_individual_side_effects = pd.read_csv('bio-decagon-mono.csv')
df_catogories_side_effects = pd.read_csv('bio-decagon-effectcategories.csv')

unique_protein_values = np.concatenate([df_ppi['Gene 1'].unique(), df_ppi['Gene
2'].unique(), df_binding['Gene'].unique()])
common_unique_protein_values = np.unique(unique_protein_values)

unique_drugs_with_poly_se = np.union1d(df_side_effects['STITCH 1'].unique(),
df_side_effects['STITCH 2'].unique())

def create_mapped_id_for_identical_subst(df, columns_name, common_values):
  unique_subst_id0_values = df[columns_name[0]].unique()
  unique_subst_id1_values = df[columns_name[1]].unique()
```

```
  common_name = columns_name[0].split(" ")[0]

  common_unique_map_df = pd.DataFrame(data={
     f'{common_name}': common_values,
     'mappedID': pd.RangeIndex(len(common_values)),
  })

  interaction_subst_id0 = pd.merge(df[columns_name[0]],
common_unique_map_df,
                        left_on=columns_name[0], right_on=f'{common_name}',
how='left')
  interaction_subst_id0 =
torch.from_numpy(interaction_subst_id0['mappedID'].values)

  interaction_subst_id1 = pd.merge(df[columns_name[1]],
common_unique_map_df,
                        left_on=columns_name[1], right_on=f'{common_name}',
how='left')
  interaction_subst_id1 =
torch.from_numpy(interaction_subst_id1['mappedID'].values)

  edge_index_subst_to_subst = torch.stack([interaction_subst_id0,
interaction_subst_id1], dim=0)

  return edge_index_subst_to_subst, common_unique_map_df
```

```
interconection_among_proteins, proteins_unique_map =
create_mapped_id_for_identical_subst(df_ppi, df_ppi.columns.tolist(),
common_unique_protein_values)

proteins = torch.from_numpy(proteins_unique_map['mappedID'].values)

data = HeteroData()

data["protein"].node_id = proteins
data["protein", "interconections", "protein"].edge_index =
interconection_among_proteins
```

train.py

```
import torch
import numpy as np
import torch_geometric

data = torch.load('/content/drive/MyDrive/decagon.pt')

from torch_geometric.data import HeteroData
import torch_geometric.transforms as T

data = T.ToUndirected(merge=False)(data)

transform = T.RandomLinkSplit(
    num_val=0.1,
    num_test=0.1,
```

```
        is_undirected=True,
        disjoint_train_ratio=0.0,
        neg_sampling_ratio=0.0,
        add_negative_train_samples=False,
        edge_types=("drug", "side_effects", "drug"),
        rev_edge_types=("drug", "rev_side_effects", "drug"),
    )

    train_data, val_data, test_data = transform(data)


    from torch_geometric.loader import LinkNeighborLoader

    edge_label_index = train_data["drug", "side_effects", "drug"].edge_label_index
    edge_label = train_data["drug", "side_effects", "drug"].edge_label
    train_loader = LinkNeighborLoader(
        data=train_data,
        num_neighbors=[10, 10],
        neg_sampling_ratio=1.0,
        edge_label_index=(("drug", "side_effects", "drug"), edge_label_index),
        edge_label=edge_label,
        batch_size=64,
        shuffle=True,
    )

    from typing import Dict, List, Union

import torch
```

```
import torch.nn.functional as F
from torch import nn
from torch.nn import Linear

import copy

import torch_geometric.transforms as T
from torch_geometric.nn import HANConv
from torch_geometric.nn.norm import GraphNorm

class HANEncoder(nn.Module):
    def __init__(self, in_channels: Union[int, Dict[str, int]],
                 hidden_channels=128, heads=8):
        super().__init__()
        self.han_conv = HANConv(hidden_channels, hidden_channels, heads=heads,
                                dropout=0.3, metadata=data.metadata(), add_self_loops=True)
        self.norm1_d = GraphNorm(hidden_channels)
        self.norm1_p = GraphNorm(hidden_channels)

    def forward(self, x_dict, edge_index_dict):
        out = self.han_conv(x_dict, edge_index_dict)
        out['drug'] = self.norm1_d(out['drug'])
        out['protein'] = self.norm1_p(out['protein'])

        return out

class EdgeDecoder(nn.Module):
    def __init__(self, hidden_channels, output_channels):
```

```
        super().__init__()
        self.lin1 = Linear(2 * hidden_channels, hidden_channels)
        self.lin2 = Linear(hidden_channels, output_channels)

    def forward(self, z_dict, edge_label_index):
        row, col = edge_label_index
        z = torch.cat([z_dict['drug'][row], z_dict['drug'][col]], dim=-1)

        z = self.lin1(z).relu()
        z = self.lin2(z).sigmoid()

        return z

class Model(nn.Module):
    def __init__(self, hidden_channels, output_channels):
        super().__init__()
        self.encoder = HANEncoder(hidden_channels, hidden_channels)
        self.decoder = EdgeDecoder(hidden_channels, output_channels)

        self.protein_emb = torch.nn.Embedding(data["protein"].num_nodes,
hidden_channels)
        self.drug_emb = torch.nn.Embedding(len(data["drug"].node_id), hidden_channels)

    def forward(self, data):
        x_dict = {
          "drug": self.drug_emb(data["drug"].node_id),
          "protein": self.protein_emb(data["protein"].node_id),
        }
```

```
        z_dict = self.encoder(x_dict, data.edge_index_dict)

        return self.decoder(z_dict, data['drug', 'side_effects', 'drug'].edge_label_index)

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

model = Model(hidden_channels=128, output_channels=data['drug', 'side_effects',
'drug'].edge_label.shape[1]).to(device)

optimizer = torch.optim.Adam(model.parameters(), lr=0.01)

weight = torch.sum(data['drug', 'side_effects', 'drug'].edge_label, axis=0)
weight = weight.max() / weight

def weighted_mse_loss(pred, target):

    return F.cross_entropy(pred, target.float()) / 2

def common_accuracy(pred, target):
    pred = (pred>0.5).long().cpu().numpy()
    target = target.long().cpu().numpy()

    intersection = np.sum(pred * target, axis=1)
    common_amount = np.sum(target, axis=1)

    result = intersection / common_amount
```

```
    return np.mean(result)*100

import tqdm

train_data.to(device)
val_data.to(device)
for epoch in range(1, 1000):
    total_loss = total_examples = 0
    optimizer.zero_grad()
    pred = model(train_data)
    ground_truth = train_data["drug", "side_effects", "drug"].edge_label
    train_acc = common_accuracy(pred, ground_truth)

    loss = weighted_mse_loss(pred, ground_truth)
    loss.backward()
    optimizer.step()
    total_loss += float(loss) * pred.numel()
    total_examples += pred.numel()

    pred_val = model(val_data)
    ground_truth_val = val_data["drug", "side_effects", "drug"].edge_label
    loss_val = weighted_mse_loss(pred_val, ground_truth_val)

    # loss_val = loss_function(torch.abs(pred_val), ground_truth_val.float())
    val_acc = common_accuracy(pred_val, ground_truth_val)

    print(f"Epoch: {epoch:03d}, Loss: {total_loss / total_examples:.4f}, Train acc:
{train_acc}, Val Loss: {loss_val}, Val acc: {val_acc}")
```

Додаток В (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ПРОГНОЗУВАННЯ ПОБІЧНИХ ЕФЕКТІВ ПОЛІПРАГМАЗІЇ ЗА ДОПОМОГОЮ ГРАФОВОЇ НЕЙОННОЇ МЕРЕЖІ

Виконав: студент 2-го курсу,
групи 1КН-22м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напрямку підготовки, спеціальності)
_____ Кузняк В. П.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
_____ Колесницький О. К.
(прізвище та ініціали)
«07» 12 2023 р.

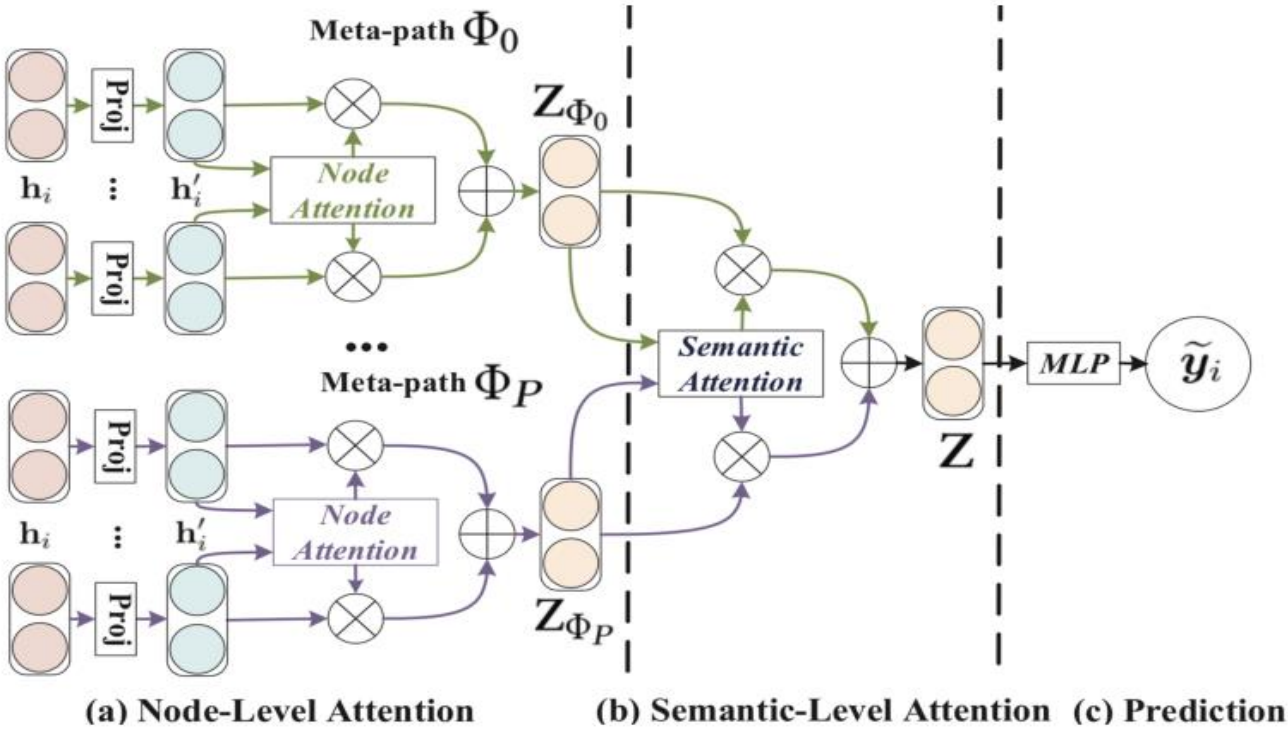


Рисунок В.1 - Архітектура гетерогенної графової мережі з увагою

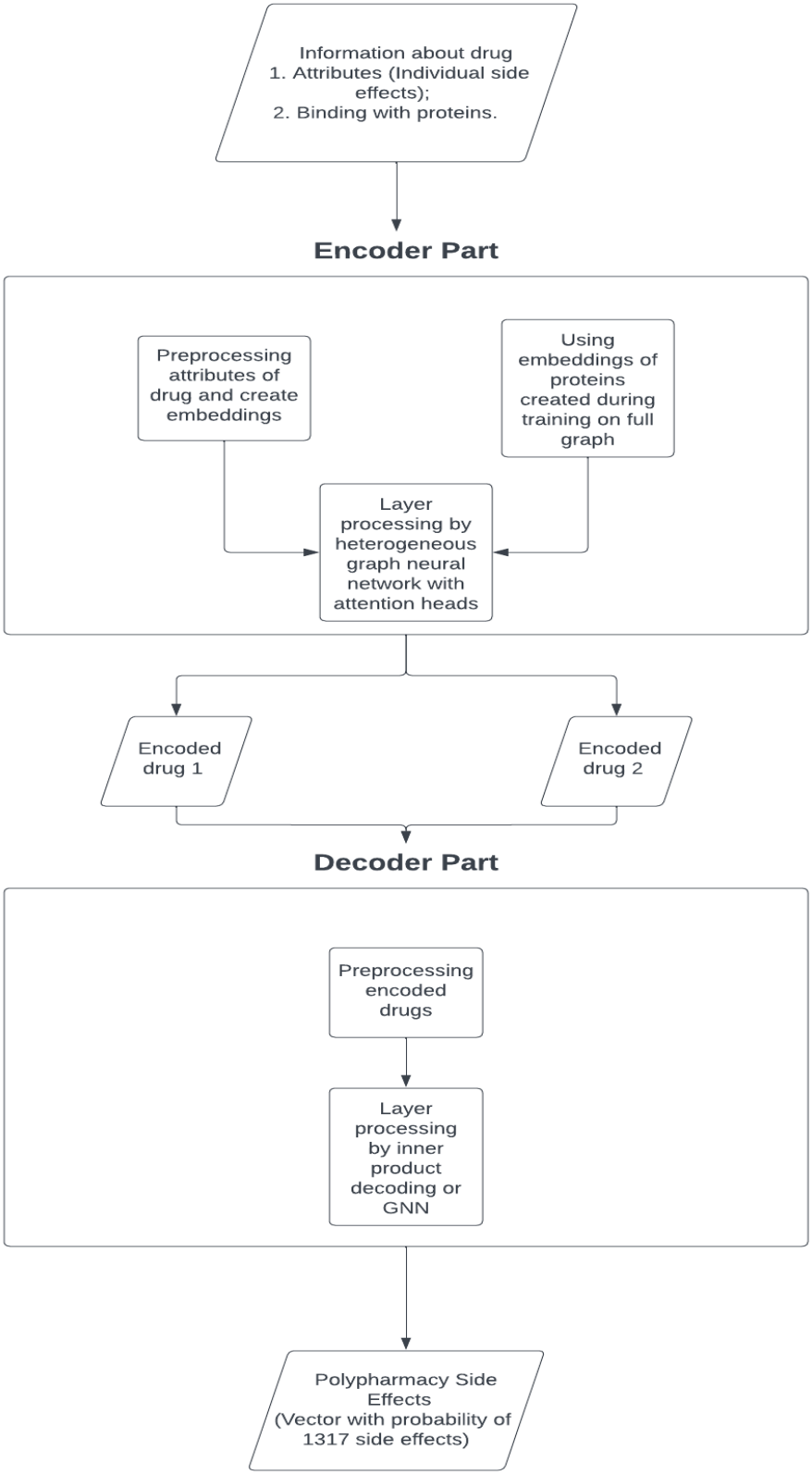


Рисунок В.2 - Схема алгоритму графової мережі

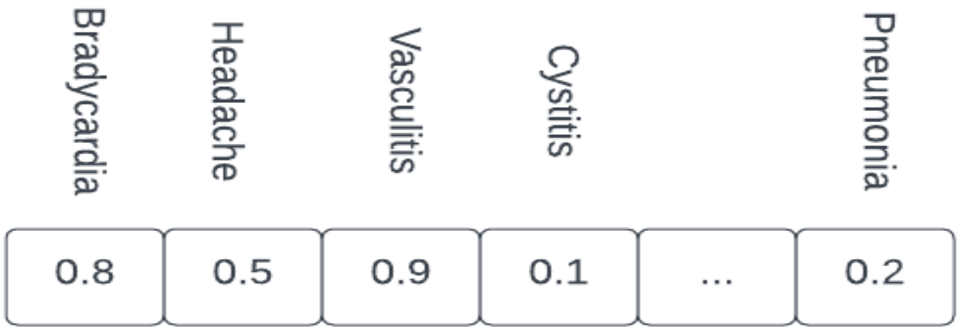


Рисунок В.3 - Приклад згенерованого вектору, який описує побічні ефекти

```
HeteroData(
  protein={ node_id=[19089] },
  drug={
    node_id=[645],
    x=[639, 10184],
  },
  (protein, interconections, protein)={ edge_index=[2, 715612] },
  (drug, side_effects, drug)={
    edge_index=[2, 63473],
    edge_label=[63473, 1317],
  },
  (drug, bindings, protein)={ edge_index=[2, 18690] }
)
```

Рисунок В.4 - Створений гетерогенний граф та збережений у зручній формі для роботи з ним

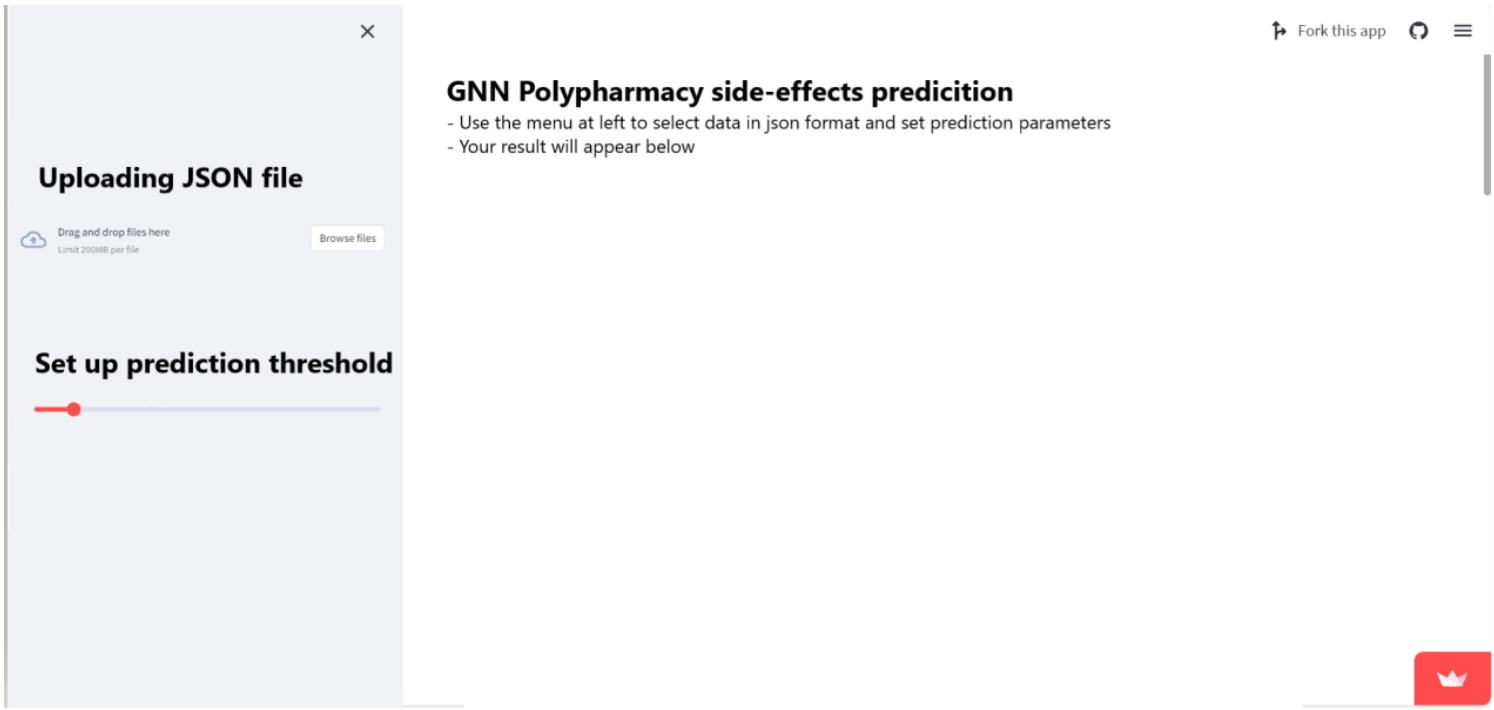


Рисунок В.5 - Загальний вигляд web інтерфейсного вікна

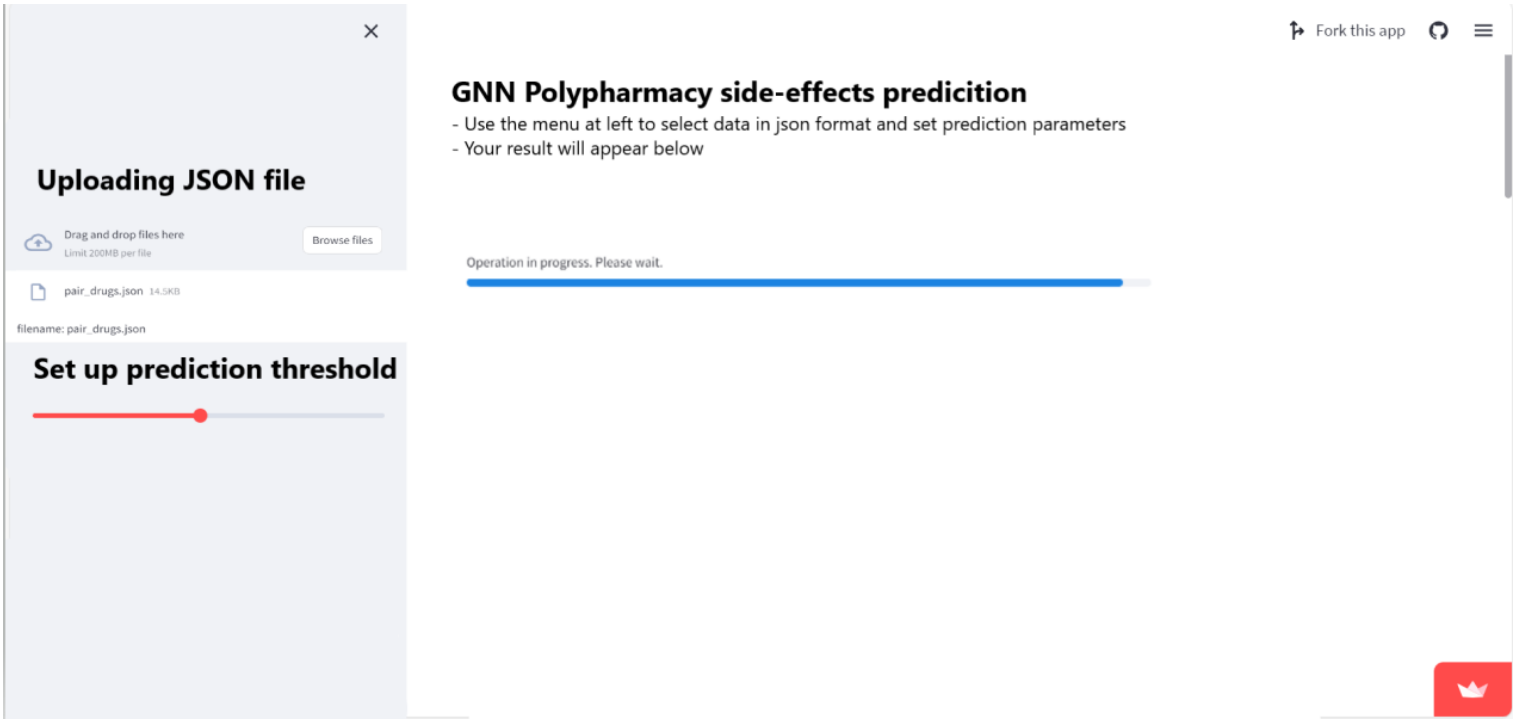


Рисунок В.6 - Загальний вигляд web інтерфейсного вікна під час завантаження файла та генерації відповіді

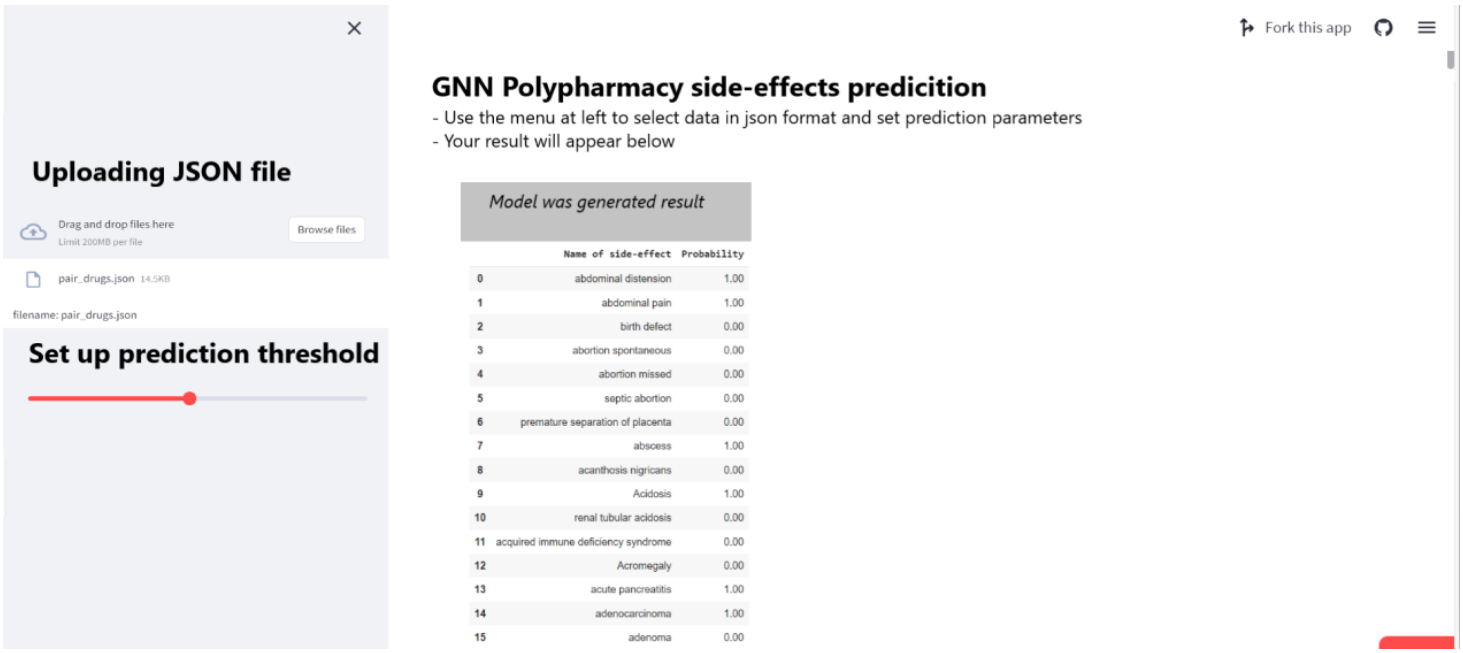


Рисунок В.7 - Результат передбачення моделі відображається у вигляді повноформатної таблиці в pandas з колонками “Назва побічної дії” та “Ймовірність”