МІНІСТЕРСТВО ОХОРОНИ ЗДОРОВ'Я УКРАЇНИ

БУКОВИНСЬКИЙ ДЕРЖАВНИЙ МЕДИЧНИЙ УНІВЕРСИТЕТ

Кафедра фармації

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

за спеціальністю 226 Фармація, промислова фармація

на тему: **МОДЕЛЮВАННЯ НОВИХ ЗАСОБІВ З ПОТЕНЦІЙНОЮ ПРОТИДІАБЕТИЧНОЮ ДІЄЮ НА ОСНОВІ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ**

Виконала: здобувач вищої освіти V курсу, 1 групи медико-фармацевтичного факультету, спеціальність 226 Фармація, промислова фармація, денна форма ТОМЕНКО Аліна Олександрівна

Керівник: доцент закладу вищої освіти кафедри фармації, доктор фармацевтичних наук, доцент ПРУГЛО Євген Сергійович

Рецензенти: завідувач кафедри медичної та фармацевтичної хімії, доктор хімічних наук, професор ЧОРНОУС Віталій Олександрович

доцент закладу вищої освіти кафедри медичної та фармацевтичної хімії, кандидат хімічних наук, доцент ГРОЗАВ Аліна Миколаївна

До захисту допущено
протокол № 18 від 02.05.2025 р.
засідання кафедри фармації
Завідувач кафедри _____ доц. Олег ГЕРУШ

Чернівці 2025

ЗМІСТ

Додаток І

Генерування молекул за допомогою Char-RNN в Pytorch

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ML (machine learning)	- машинне навчання
SVM (support vector machine)	- метод опорних векторів
SGD (Stochastic Gradient Descent)	- стохастичний градієнтний спуск
NB (Naive Bayes)	- наївний Байєс
ma-separated values)	- файловий формат, котрий є відмежовувальним форматом для представлення табличних даних
SMILES (simplified molecular-input line-entry system)	- система правил (специфікація) однозначного опису складу та структури молекули хімічної речовини з використанням рядка символів ASCII у рядковому типі.
IUPAC (international union of pure and applied chemistry)	- міжнародна недержавна організація, метою якої є сприяння розвитку хімії як науки, знана як визнане авторитетне джерело з розвитку стандартів найменування хімічних елементів та їхніх складових
JSON (JavaScript object notation)	- текстовий формат обміну даними між комп'ютерами, дає змогу описувати об'єкти та інші структури даних.

ВСТУП

Актуальність теми. Цукровий діабет посідає провідне місце серед найпоширеніших патологій сучасного людства. Статистичні дані демонструють, що у різноманітних державах світу частка осіб із цукровим діабетом становить від 4% до 7% загального населення. Примітно, що з підвищенням віку зростає і ризик розвитку даного захворювання, сягаючи 10-15% серед людей, які перетнули 65-річний рубіж. Світові тенденції вказують на стрімке збільшення поширеності цукрового діабету. Аналізуючи прогнози щодо розповсюдження цієї патології, можна відзначити, що економічно розвинуті країни очікують значний приріст діабету до 2030 року переважно серед людей старше 65 років. Натомість у країнах, що розвиваються, спостерігається інша картина – збільшення числа пацієнтів із діабетом у віковій категорії 45-64 роки. На сьогоднішній день глобальна кількість хворих на цукровий діабет сягає 371 мільйона осіб. Епідеміологічні дослідження стосовно України також підтверджують невпинне зростання випадків цього метаболічного розладу серед населення країни.

Створення інноваційного фармацевтичного препарату представляє собою надзвичайно тривалий та вартісний процес. Між моментом виявлення активної речовини та початком її промислового виробництва й виведенням на ринок минає 10-15 років, а сукупні інвестиції у проект сягають позначки 1,8 мільярда доларів США. Такі значні часові та фінансові ресурси витрачаються на багаторазові випробування та елімінацію численних сполук під час скринінгових процедур на кожному етапі доклінічних і клінічних досліджень. Чимало інвесторів втратили зацікавленість у фінансуванні розробок нових медикаментів (або фармацевтичних стартапів) через високий ступінь ризику та довготривалий період очікування повернення вкладених коштів. Оскільки вся дослідницька інформація розглядається фармацевтичними компаніями як конфіденційна і не розголошується науковій спільноті, різні дослідницькі групи витрачають колосальні ресурси, дублюючи ідентичні високовартісні експерименти з пошуку лікарських засобів [13].

У зв'язку з цим, впровадження інноваційних підходів, зокрема, методів машинного навчання, у процес розробки ліків набуває все більшої актуальності [59].

Машинне навчання (ML) пропонує потужні інструменти для аналізу великих обсягів даних, виявлення прихованих закономірностей та прогнозування властивостей хімічних сполук. Застосування алгоритмів ML у фармацевтичній галузі дозволяє значно прискорити етап відбору потенційних кандидатів для подальших досліджень, зменшити витрати на розробку та підвищити ймовірність успіху [52].

Дана робота присвячена розробці та застосуванню інноваційних методів машинного навчання для моделювання нових хімічних сполук з потенційною протидіабетичною дією. В роботі запропоновано комплексний підхід, що включає збір та обробку даних про відомі протидіабетичні препарати, конструювання молекулярних дескрипторів, побудову предиктивних моделей та генерацію нових молекул з прогнозованою біологічною активністю.

Актуальність даного дослідження зумовлена кількома факторами. Зростання захворюваності на цукровий діабет у світі та необхідність розробки нових, більш ефективних лікарських засобів для його лікування. Висока вартість та тривалість традиційних методів розробки нових лікарських препаратів, що потребує впровадження інноваційних підходів для прискорення та оптимізації цього процесу. Стрімкий розвиток методів машинного навчання та їх успішне застосування у різних галузях, включаючи медичну хімію та фармацію. Накопичення великих обсягів даних про хімічні сполуки та їх біологічну активність, що створює підґрунтя для застосування методів машинного навчання. Необхідність розробки ефективних алгоритмів для прогнозування фармакологічних властивостей хімічних сполук та генерації нових молекул із заданими властивостями.

Мета та завдання дослідження. Метою дослідження була розробка та апробація комплексного підходу до моделювання нових хімічних сполук з

потенційною протидіабетичною дією на основі алгоритмів машинного навчання.

Для досягнення мети були поставлені наступні завдання:

- Провести аналіз сучасних підходів та перспектив застосування алгоритмів машинного навчання у фармацевтичній і медичній практиці.
- Розробити методику збору, первинної обробки, нормалізації та категоризації даних про відомі лікарські засоби з протидіабетичною дією та без неї.
- Здійснити конструювання молекулярних ознак (дескрипторів) на основі класифікації ClassyFire та бібліотеки mordred для створення інформативного набору даних.
- Розробити та впровадити стратегії подолання дисбалансу класів при підготовці тренувальних даних для підвищення точності моделей.
- Побудувати та оцінити різні моделі машинного навчання для прогнозування протидіабетичної активності хімічних сполук.
- Реалізувати алгоритм генерації нових молекул на основі SMILES-представлення з використанням рекурентних нейронних мереж.
- Провести відбір згенерованих молекул за допомогою побудованих моделей прогнозування та оцінити їх потенційну протидіабетичну дію.
- Розробити комплекс програмних засобів для автоматизації процесів обробки даних, моделювання та прогнозування біологічної активності хімічних сполук.

Об'єктами дослідження було моделювання та прогнозування фармакологічних властивостей хімічних сполук з використанням методів машинного навчання.

Предметом дослідження стали методи та алгоритми машинного навчання для аналізу, прогнозування та генерації нових хімічних сполук з потенційною протидіабетичною дією.

Методи дослідження. У роботі використано комплекс сучасних методів дослідження, зокрема: методи збору та аналізу даних з відкритих баз даних (DrugBank), методи хемоінформатики для представлення та аналізу хімічних структур (використано стандартизований формат SMILES, застосовано

бібліотеку mordred для обчислення понад 1800 фізико-хімічних та структурних дескрипторів молекул, використано API ClassyFire для категоризації сполук за хімічними суперкласами, класами та підкласами, застосовано алгоритми мінімізації енергії для переведення 2D-представлень молекул у 3D-конформації з метою розрахунку просторових дескрипторів), алгоритми машинного навчання (логістична регресія з L1 та L2 регуляризацією, метод опорних векторів (SVM), ансамблеві методи (Random Forest, Gradient Boosting), k-найближчих сусідів (k-NN), наївний баєсівський класифікатор, рекурентні нейронні мережі (Char-RNN)), методи оцінки та інтерпретації моделей (використано метрики accuracy, F1-score, precision, recall, ROC-AUC), статистичні методи для аналізу отриманих результатів (методи перевірки статистичних гіпотез, а саме хі-квадрат)

Наукова новизна дослідження полягає в розробці комплексного підходу до моделювання нових хімічних сполук з потенційною протидіабетичною дією, що включає комплексне застосування методів конструювання молекулярних дескрипторів на основі класифікації ClassyFire та бібліотеки mordred, використання модифікованого алгоритму генерації нових молекул на основі SMILES-представлення з використанням рекурентних нейронних мереж, та використання даного підходу до відбору згенерованих молекул з потенційною протидіабетичною за допомогою навчених моделей машинного навчання.

Практичне значення отриманих результатів полягає в створенні набору програмних інструментів які можуть використовуватись для автоматизації процесів збору, обробки даних та моделювання нових лікарських засобів. Використанні навчених моделей машинного навчання для прогнозування протидіабетичної активності хімічних сполук та генерації набору нових хімічних структур з прогнозованою протидіабетичною дією, які можуть стати основою для подальших експериментальних досліджень та розробки нових лікарських препаратів. Можливість впровадження розроблених моделей та методів та підходів у практику дослідницьких лабораторій та фармацевтичних компаній які займаються пошуком нових протидіабетичних засобів.

Структура та обсяг роботи. Робота складається з вступу, п’ятьох розділів, висновків, списку використаних джерел та додатків. У першому розділі проведено аналіз сучасних підходів до застосування алгоритмів машинного навчання у фармацевтичній і медичній практиці. Другий розділ присвячено опису матеріалів та методів досліджень. У третьому розділі представлено методику збору, обробки та підготовки даних для моделювання. Четвертий розділ містить результати побудови моделей машинного навчання, генерації нових молекул та їх відбору. У додатках наведено програмні коди та деталі реалізації розроблених алгоритмів.

Публікації. За результатами магістерської роботи опубліковано 2 тез доповідей.

РОЗДІЛ 1
СУЧАСНІ ПІДХОДИ ТА ПЕРСПЕКТИВИ ЗАСТОСУВАННЯ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ У ФАРМАЦЕВТИЧНІЙ І МЕДИЧНІЙ ПРАКТИЦІ
(огляд літератури)

1.1. Основи класичних методів машинного навчання та приклади їх запровадження

Алгоритми машинного навчання (ML) — це методи обчислень, що дають змогу системам вчитися на даних і покращувати виконання певних завдань без потреби в явному програмуванні. Вони є основою різних застосувань, від розпізнавання зображень до аналізу природної мови.

Розрізняють алгоритми класичного навчання з вчителем та без вчителя, див рис. 1.1

Рис. 1.1 Алгоритми машинного навчання

Класифікація, належить до алгоритмів з вчителем, та є ключовим завданням у машинному навчанні, яке передбачає прогнозування категорії вхідних даних на основі їхніх характеристик. Цей підхід широко застосовується в різних галузях, таких як медицина та техніка.

У реальному світі класифікація має безліч практичних застосувань. Наприклад, у сфері охорони здоров'я вона використовується для діагностики захворювань, а в природничих науках допомагає визначати та розподіляти рослини за видами.

Логістична регресія — це базовий алгоритм машинного навчання, що використовується для задач бінарної класифікації. Цей статистичний метод дозволяє оцінити ймовірність, з якою об'єкт належить до певного класу. Незважаючи на назву, логістична регресія призначена саме для класифікації, а не для регресійних задач, хоча і використовується для них.

Логістична регресія знаходить застосування в медицині, екології, економіці, транспорті, страхуванні, а також у науці про дані [4; 42; 57; 62; 70; 76; 77; 85; 100]. Наприклад, в астрономії — для класифікації зірок [118], а в бізнесі — для аналізу поведінки клієнтів, а також класифікує електронні листи як спам чи звичайну пошту [105].

Метод опорних векторів (support vector machine, **SVM)** — це потужні алгоритми керованого навчання, які застосовуються як до задач класифікації, так і до регресії. Їхній основний принцип полягає у визначенні оптимальної гіперплощини, яка максимально відокремлює класи у багатовимірному просторі ознак. SVM особливо добре працюють із нелінійними задачами та великими об'ємами даних.

Ключовим компонентом SVM є задача оптимізації, що полягає у знаходженні гіперплощини, яка забезпечує максимальний запас між класами при мінімізації помилок класифікації. Це досягається шляхом розв'язання задачі квадратичного програмування з лінійними обмеженнями.

Лінійне ядро — найпростіший тип ядра, еквівалентний застосуванню точкового добутку в початковому просторі ознак. Воно підходить для лінійно

роздільних даних і добре працює, якщо кількість ознак значно перевищує кількість зразків.

Поліноміальне ядро дозволяє моделі виявляти нелінійні межі рішень. Сигмоподібне ядро, менш поширене, але іноді застосовується в задачах, де дані демонструють сигмоїдну поведінку. Однак його використання обмежене, оскільки не всі параметри гарантують позитивну напіввизначеність.

Останні дослідження демонструють значні переваги SVM у виявленні лікарських засобів, підкреслюючи їхню гнучкість і точність. Водночас визначаються обмеження та виклики, з якими стикається цей метод, відкриваючи напрямки для його вдосконалення у майбутньому [43].

Стохастичний градієнтний спуск (Stochastic Gradient Descent, SGD) - це простий, але дуже ефективний підхід до підгонки лінійних класифікаторів і регресорів під опуклі функції втрат, такі як (лінійні) машини опорних векторів і логістична регресія. Незважаючи на те, що SGD існує в спільноті машинного навчання вже давно, він отримав значну увагу в контексті великомасштабного навчання лише нещодавно. SGD був успішно застосований до великомасштабних і розріджених проблем машинного навчання, які часто зустрічаються в класифікації текстів і обробці природної мови. Враховуючи розрідженість даних, класифікатори в цьому модулі легко масштабуються до задач з більш ніж 10^5 навчальних прикладів і більш ніж 10^5 ознак [50; 80; 91].

K-Nearest Neighbors — це простий, але ефективний алгоритм машинного навчання, що використовується для класифікації та регресії. Його основний принцип полягає в тому, що подібні точки даних мають схильність до однакових результатів. Для прогнозування KNN аналізує k найближчих сусідів у просторі ознак і на основі їхніх значень визначає клас більшості або обчислює середнє значення.

Принцип методу найближчих сусідів полягає в тому, щоб знайти заздалегідь визначену кількість навчальних зразків, найближчих за відстанню до нової точки, і передбачити мітку на основі цих зразків. Кількість зразків може бути константою, визначеною користувачем (навчання за k найближчих

сусідів), або змінюватися залежно від локальної щільності точок (навчання за радіусом найближчого сусіда). Відстань може бути будь-якою метричною мірою: стандартна евклідова відстань є найпоширенішим вибором. Методи на основі сусідів відомі як неузагальнюючі методи машинного навчання, оскільки вони просто "запам'ятовують" всі свої навчальні дані (можливо, перетворені в структуру швидкого індексування, таку як кульове дерево або KD-дерево).

k-найближчих сусідів застосовують для аналізу настрою якості обслуговування облікових записів Twitter [93] для розрізнення гамма-променів і нейтронів в органічних сцинтиляторах [21] для прогнозування електричного навантаження [18] а також розпізнавання мови жестів [71].

Наївний Байєс (NB) - це алгоритм машинного навчання, що базується на ймовірнісному підході та використовує теорему Байєса. Його часто застосовують для класифікаційних задач, особливо для аналізу тексту та фільтрації спаму. Основна ідея алгоритму полягає в припущенні, що всі ознаки незалежні між собою, звідси і назва "наївний".

Існує три основні типи наївних класифікаторів Байєса: гаусівський, мультиноміальний та бернулівський. Вибір типу залежить від типу даних, з якими працює алгоритм.

Гаусівський NB передбачає, що ознаки мають нормальний розподіл. Він підходить для роботи з безперервними даними та часто застосовується для задач, де змінні неперервні.

Мультиноміальний NB використовується для роботи з дискретними даними, наприклад, для підрахунку частоти слів у задачах класифікації тексту. Він добре підходить для класифікації документів і фільтрації спаму.

Бернулівський NB орієнтований на двійкові чи булеві ознаки. Зазвичай він застосовується в текстовій класифікації з використанням моделі "мішок слів", де важливо лише, чи присутнє певне слово в документі.

Цей метод знайшов широке застосування в різних сферах. Наприклад, його використовують у системах фільтрації спаму [89] для діагностики Covid-19 [115], визначення стану харчування дітей раннього віку [111], прогнозування

рівня розчиненого кисню в аквакультурі молюсків [102], оцінки ймовірності передчасних пологів [110], а також для виявлення фейкових новин [74].

Дерево рішень — це алгоритм навчання під наглядом, який використовується для задач класифікації та регресії. Воно являє собою деревоподібну структуру, де внутрішні вузли відповідають функціям або умовам, гілки представляють правила прийняття рішень, а листові вузли містять прогнози або результати. Завдяки простоті та здатності працювати з числовими й категоричними даними, дерева рішень легко інтерпретувати, а також вони здатні моделювати нелінійні залежності.

Процес побудови дерева рішень полягає в рекурсивному поділі даних на підмножини на основі значень ознак. Побудова починається з усього набору даних, а розділення здійснюється так, щоб забезпечити найкраще відділення класів або цільової змінної.

Широко використовується в епідеміологічних дослідженнях [83] в криміналістиці [56], розпізнавання мовних емоцій [6], медицині [27; 34; 60] у банківському секторі [32]

Градієнтний бустинг (GBM) — це потужний метод ансамблевого навчання, який використовується в машинному навчанні як для задач регресії, так і для класифікації. Метод полягає в створенні ансамблю слабких моделей, зазвичай дерев рішень. Спочатку створюється проста модель, а потім поступово додаються нові моделі для корекції помилок попередніх. Градієнтний бустинг складається з трьох основних компонентів: функції втрат, слабкого учня та адитивної моделі. Функція втрат оцінює точність моделі, слабкий учень (часто дерево рішень) виконує прогнозування, а адитивна модель комбінує слабкі моделі для створення сильної. Ряд робіт присвячено застосуванні даного алгоритму в медицині[73; 87].

Лінійний дискримінантний аналіз (LDA) - це підхід, який використовується в керованому машинному навчанні для вирішення завдань багатокласової класифікації. LDA відокремлює декілька класів з декількома ознаками шляхом зменшення розмірності даних. Перший етап LDA — це

обчислення середнього значення для кожного класу, що потім використовується для побудови матриці розсіювання між класами. Матриця розсіювання всередині класу показує, як вибірки розташовані навколо середніх значень своїх класів. Її обчислюють шляхом підсумовування коваріаційних матриць кожного класу. Як і в випадках попередньо описаних алгоритмів, використовується в медицині [16].

Рекурентні нейронні мережі (RNN) — це тип архітектури нейронних мереж, призначений для обробки послідовних даних, підтримують внутрішній стан, який фіксує інформацію з попередніх вхідних даних, що дозволяє їм моделювати залежності та шаблони, присутні в послідовних даних.

Дана нейронна мережа працює з двома типами вхідних даних: поточними і нещодавніми минулими. Це важливо, оскільки послідовність даних містить критичну інформацію, яка впливає на прогнозування майбутніх подій. Завдяки цьому RNN здатна вирішувати задачі, які недоступні для інших алгоритмів. У процесі роботи мережа з послідовним поширенням використовує матрицю ваг для обробки вхідних даних і формування виходу. Особливістю RNN є те, що вона визначає ваги як для поточних, так і для минулих даних, адаптуючи їх під час навчання. Для цього застосовуються градієнтний спад і метод часової зворотної передачі помилки, що дозволяє ефективно змінювати ваги та оптимізувати модель.

Застосовуються для класифікації зображень [15], прогнозування даних часових рядів [84], прогноз цін на фондовому ринку [108], моделювання мови та прогнозування наступного слова [96], класифікація веб-сторінок [8], в медицині [94; 106].

1.2. Інноваційні стратегії розробки лікарських засобів із використанням алгоритмів машинного навчання

Дослідження хімічних властивостей та їх зв’язку з біологічною активністю ліків набувають все більшого значення у сучасній фармацевтичній науці [33; 68; 72; 75; 98; 114]. Особливо актуальною є проблема прогнозування

антидіабетичних властивостей сполук [2; 3; 53–55], оскільки цукровий діабет залишається однією з найпоширеніших хронічних хвороб у світі. Незважаючи на прогрес у лікуванні та контролі діабету, пошук нових ефективних препаратів із мінімальними побічними ефектами залишається важливим завданням.

Розвиток методів машинного навчання та великих даних дозволяє дослідникам ефективно аналізувати великі масиви хімічних сполук і прогнозувати їх потенційні властивості, зокрема антидіабетичні. Це не лише прискорює процес розробки ліків, але й зменшує витрати на експериментальні дослідження.

Метод множинної лінійної регресії (MLR) у поєднанні з методом відбору ознак та генетичним алгоритмом (GA) використовувався для моделювання даних інгібування серії найпотужніших інгібіторів тирозинази для лікування гіперпігментації [6].

Множинна логістична регресія (MLogR) застосовувалася для розробки математичної моделі інгібуючої активності сполук на циклооксигеназу-2 (ЦОГ-2) [79].

Різні варіанти лінійної регресії з регуляризацією використовувалися для покращення моделей та запобігання перенавчанню при прогнозуванні реакції протипухлинних препаратів на лінії ракових клітин. У цьому дослідженні було запропоновано метод ансамблевого навчання, що інтегрує модель завершення матриці низького рангу та модель гребневої регресії [39].

У дослідженнях, присвячених раку, розроблено кореляційний адаптивний алгоритм LASSO (CorrLASSO) та TG-LASSO [12; 26], а також використовується алгоритм Elastic-Net [69; 81].

Порівняння типових методів машинного навчання, таких як дерева рішень (XGBoost, LightGBM) та штучні нейронні мережі (MLP, CNN), підтвердило ефективність їх використання для скринінгу та розробки лікарських засобів [61]. Надійність стратегії ідентифікації важливих молекулярних фрагментів за допомогою алгоритмів Random Forest, XGBoost і LightGBM також була підтверджена [36]. У дослідженні з використанням

алгоритму XGBoost та байєсовської оптимізації було проведено класифікацію активності інгібітора AXL-кінази для відкриття ліків проти раку [117].

Важливість методу опорних векторів (SVM) як провідного підходу для класифікації сполук, прогнозування властивостей та віртуального скринінгу також підкреслюється у низці робіт [28; 43; 66]. Крім того, описані дослідження, що використовують наївний байєсівський класифікатор (NB) [104] та метод k-найближчих сусідів (k-NN) [29; 90; 104; 117].

Такі підходи у поєднанні з великими даними дозволяють підвищити точність прогнозування біологічної активності сполук, що робить їх перспективними для розробки нових лікарських засобів.

1.3. Узагальнення результатів і перспективні напрями подальших досліджень

Проведений огляд демонструє широкий спектр застосування методів машинного навчання в різних галузях, зокрема в медицині та фармації.

Класичні методи машинного навчання (логістична регресія, SVM, наївний Байєс, дерева рішень, ансамблеві методи) знаходять багатогранне застосування у різних сферах – від діагностики захворювань до прогнозування біологічної активності хімічних сполук. Градієнтний бустинг (зокрема XGBoost, LightGBM) та Random Forest продемонстрували високу ефективність у задачах класифікації та прогнозування, особливо в контексті розробки лікарських засобів та скринінгу сполук.

У сфері розробки лікарських засобів найбільш продуктивними виявилися такі методи як SVM, регресійні моделі з регуляризацією (LASSO, Elastic-Net) та ансамблеві методи, що підтверджено численними дослідженнями.

Найбільш ефективні результати досягаються при поєднанні різних алгоритмів машинного навчання та методів відбору ознак, що дозволяє значно прискорити процес розробки ліків і зменшити експериментальні витрати.

Такі напрями досліджень мають значний потенціал для прискорення розробки нових лікарських засобів, зменшення витрат на експериментальні

дослідження та підвищення ефективності експериментальних підходів до розробки нових засобів при лікуванні таких поширених захворювань як цукровий діабет.

Тому комплексне використання алгоритмів машинного навчання у розробці нових лікарських засобів має не лише теоретичну а й практичну значимість.

РОЗДІЛ 2
МАТЕРІАЛИ ТА МЕТОДИ ДОСЛІДЖЕНЬ

Набір даних для подальшого аналізу та розробки моделей машинного навчання було підготовлено на основі припущення, що функціональні групи, присутні в лікарських засобах, а також їх анатомо-терапевтична класифікація можуть забезпечити надійну основу для побудови ефективних моделей. Моделі були спрямовані на прогнозування протидіабетичних ефектів (код АТХ А10) на основі хімічної структури сполук.

Для підготовки набору даних був розроблений парсер (додаток А) для краулінгу бази даних **DrugBank** [30] з якої були зібрані дані щодо хімічної структури та активності лікарських засобів (код АТХ). Парсер був розроблений на мові програмування **Python** за допомогою бібліотек **BeautifulSoup** [113] для парсингу HTML-контенту та **requests** [112] для здійснення HTTP-запитів.

Попередньо перед формуванням дескрипторів для ML проведена підготовка даних з засобами з протидіабетичною дією (додаток Б) та сполук які не належать до речовин з класом А10 АТХ класифікатора (додаток В).

Для класифікації отриманих сполук обрано ресурс ClassyFire [17], де було використано готовий набір даних «DrugBank_5_classyfire_21_annotations.csv»[17]. У якості предикторів використовувалися **ChemOntID** з цього ресурсу та ID лікарських засобів з **DrugBank** як цільова змінна. Для лікарських засобів, яких не було в наборі даних **ClassyFire**, предиктори були згенеровані через API цього ресурсу (додаток Г, Д).

Для класифікації молекул використовувалися дескриптори, згенеровані за допомогою бібліотеки **Mordred**, що дозволяє отримати числове представлення хімічних властивостей молекул [48].

Попередньо, для створення вказаних дескрипторів, потрібно було провести розрахунки 3D структури речовин, що досліджуються).

Для розрахунків геометрії органічних сполук є кілька методів, які відрізняються співвідношенням точності та ресурсоємності. Найменш ресурсоємні є методи молекулярної механіки (ММ) такі як ММ2, ММ3, ММ4, MMFF94, вони є дуже швидкими, підходять для великих молекул, але їх точність доволі обмежена. Тому для попередньої оптимізації був вибраний напівемпіричний метод, до них відносяться AM1, PM3, PM6, PM7. Більш точним, але і ресурсоємким є метод DFT (теорія функціоналу густини) такі як B3LYP, M06-2X, ωB97X-D тощо.

Ми використали напівемпіричний метод PM6 (метод параметризації 6), який дозволяє отримати оптимізаційну геометрію великих систем за допомогою розпізнавання d орбіталей та включає в собі нові експериментальні дані, такі як енергія димеризації, також в цьому методі додано багато парних взаємодій, що дозволяють відрегулювати основну функцію відштовхування [14].

Оптимізацію проводили за допомогою Gaussian16 та Open Babel на базі операційної системи ubuntu 20.04. Gaussian 16 — це програмний пакет для квантово-хімічних розрахунків, який використовується для моделювання молекулярних систем. Він дозволяє проводити обчислення таких властивостей молекул, як енергія, структура, спектроскопічні характеристики, реакційна здатність та інше. Gaussian 16 є однією з найпоширеніших програм у галузі квантової хімії та широко використовується в наукових дослідженнях. За допомогою Gaussian16 можна проводити розрахунки електронної структури (методи Hartree-Fock, теорія функціоналу густини (DFT), метод coupled cluster (CC) та інші.), спектроскопічних даних (ІЧ-, УФ- та ЯМР-спектри), визначення перехідних станів та енергій активації.

Open Babel широко використовується в хімічних дослідженнях, біоінформатиці та молекулярному моделюванні, це вільне програмне забезпечення з відкритим вихідним кодом, призначене для хімічної інформатики. Воно надає інструменти для конвертації, обробки та аналізу хімічних даних у різних форматах. Open Babel дозволяє конвертувати файли між різними хімічними форматами, наприклад, з .sdf у .mol, .pdb у .xyz тощо,

може генерувати тривимірні структури молекул із SMILES-рядків або інших форматів, дозволяє виконувати різні операції з молекулами, такі як оптимізація геометрії, додавання водневих атомів, розрахунок молекулярних дескрипторів.

Фрагмент коду для оптимізації молекулярної структури з SMILES представлений у додатку Е.

Для вирівнювання дисбалансу класів було використано бібліотеку **imblearn** [77] в середовищі **Python 3.8** [116].

Отриманий набір даних було розподілено на **тестовий (Test Data)** та **тренувальний (Train Data)** з використанням крос-валідації (12 фолдів), що було реалізовано за допомогою бібліотеки **scikit-learn** [103]. Це дозволило перевірити, як модель працює на різних частинах даних, і забезпечило більш стабільний показник її якості порівняно з простим поділом на тренувальні та тестові набори.

Продуктивність моделей оцінювалася за допомогою таких метрик: **Accuracy**, **Precis ion**, **Recall**, **F1 Score** та **ROC AUC**, розрахованих з використанням бібліотеки **scikit-learn**. В рамках дослідження було побудовано кілька моделей машинного навчання для прогнозування антидіабетичних ефектів лікарських сполук [24].

До побудованих моделей належать лінійні моделі, включаючи **Logistic Regression** (Логістична регресія), **Lasso Regression** (Регресія Лассо), **Ridge Regression** (Гребнева регресія), **Elastic Net** (Еластична мережа) та **SGD Classifier** (Стохастичний градієнтний спуск), реалізовані в бібліотеці **scikit-learn** [97; 103]. Також були застосовані наївні байєсівські моделі: **Bernoulli Naive Bayes** та **Multinomial Naive Bayes**. Моделі на основі дерев рішень включають **Decision Tree**, **Random Forest** та **Gradient Boosting**[97].

Додатково були використані такі методи, як **Support Vector Machines (SVM)**, **Linear Discriminant Analysis (LDA)** та **k-Nearest Neighbors (k-NN)**.

РОЗДІЛ 3
ЗБІР, ПЕРВИННА ОБРОБКА, НОРМАЛІЗАЦІЯ, КАТЕГОРИЗАЦІЯ ДАНИХ ТА ПОБУДОВА НОВИХ ОЗНАК

ETL (extract, transform, load) визначається як комплексний процес, який дає змогу збирати дані з багатьох джерел в єдиний центр обробки даних. Під час завантаження, він аналізує та агрегує дані для аналітичних завдань, процесів машинного навчання тощо [92].

ETL охоплює три поетапних процеси - першим кроком є копіювання або експорт даних із джерел даних у проміжні області. Дані, які були завантажені у проміжну область, є сирими й необробленими. Відповідно, у другому етапі їх потрібно очистити, підготувати та трансформувати для аналізу. Останній етап — завантаження. На цьому етапі дані завантажуються із проміжної області в кінцеве сховище (рис. 3).

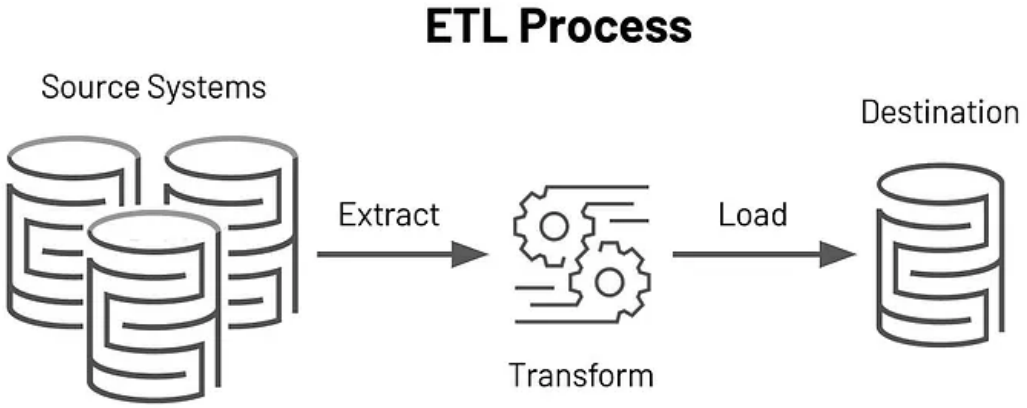


Рис. 3 Загальна схема процесів вилучення, трансформування та завантаження даних

3.1 Збір та імпорт даних. Очищення даних та обробка відсутніх значень.

На першому етапі досліджень в якості об'єкта досліджень була вибрана база даних з лікарськими засобами **DrugBank**, після скрапінгу якої (додаток А) була створена база даних з зібраними даними **sqlite** в якій міститься хімічна формула лікарських засобів в форматі SMILES та інформація про належність до певної групи анатомо-терапевтично хімічної класифікації (АТХ), зібрана БД містить таблиці 'describe', 'weight', 'properties', 'ATC', 'derivatives'), в подальшому будуть використовуватись дані з таблиць 'describe' та 'ATC'.

Після виконання запиту до таблиць БД **sqlite** 'describe' та 'ATC' видно що зібрано інформацію про 17287 id в базі даних **DrugBank с**еред яких лише у 3403 лікарських засобів наявний запис про приналежність до АТХ класифікатора.

Табл. 3.1.1 Результат запиту до бази даних з таблицею 'ATC'

index	**id**	**ATC**
0	DB00001	B01AE
1	DB00001	B01A
2	DB00001	B01
3	DB00001	B
4	DB00001	B01AE02
...	...	...
26654	DB17287	L03AX

26655	DB17287	L03A
26656	DB17287	L03
26657	DB17287	L
26658	DB17287	L03AX19

Проаналізувавши завантажені таблиці, також було встановлено, що таблиця 'ATC' містить 65 записів з класифікатором «A10», що відповідає класу для засобів з гіпоглікемічною та протидіабетичною дією.

Для подальшого формування цільового набору даних потрібно відібрати ті речовини (лікарські засоби) які в БД sqlite наявний класифікатор АТХ та наявне представлення хімічної структури в форматі SMILES. Для цього знову виконуємо запит до БД sqlite

Варто відмітити що з 12113 записів у деяких записів в базі даних DrugBank не доступна інформація щодо хімічної структури засобів у форматі SMILES (**Табл. 3.1.2.**).

Табл. 3.1.2 Результат запиту до бази даних з таблицею 'describe'

№ з/п	Id	Generic_Name	SMILES
1	DB00006	Bivalirudin	CC[C@H](C)[C@H](NC(=O)[C@H](CCC(O)=O)NC(=O)[C@H](CCC(O)=O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(O)=O)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)CNC(=O)CNC(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCNC(N)=N)NC(=O)[C@@H]1CCCN1C(=O)[C@H](N

			)CC1=CC=CC=C1)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(O)=O)C(=O)N[C@@H](CCC(O)=O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(O)=O
2	DB00007	Leuprolide	CCNC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCNC(N)=N)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H]1CCC(=O)N1
3	DB00014	Goserelin	CC(C)C[C@H](NC(=O)[C@@H](COC(C)(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC2=CC=CC=C12)NC(=O)[C@H](CC1=CN=CN1)NC(=O)[C@@H]1CCC(=O)N1)C(=O)N[C@@H](CCCN=C(N)N)C(=O)N1CCC[C@H]1C(=O)NNC(N)=O
4	DB00027	Gramicidin D	CC(C)C[C@@H](NC(=O)CNC(=O)[C@@H](NC=O)C(C)C)C(=O)N[C@@H](C)C(=O)N[C@H](C(C)C)C(=O)N[C@@H](C(C)C)C(=O)N[C@H](C(C)C)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@H](CC(C)C)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C

			(=O)N[C@H](CC(C)C)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@H](CC(C)C)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)NCCO
5	DB00035	Desmopressin	NC(=O)CC[C@@H]1NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)CCSSC[C@H](NC(=O)[C@H](CC(N)=O)NC1=O)C(=O)N1CCC[C@H]1C(=O)N[C@H](CCCNC(N)=N)C(=O)NCC(N)=O
…	…	…	…
12109	DB17414	TK216	OC1=NC2=C(Cl)C=CC(Cl)=C2C1(O)CC(=O)C1=CC=C(C=C1)C1CC1
12110	DB17418	Retinamidic acid	C\C(\C=C\C1=C(C)CCCC1(C)C)=C/C=C/C(/C)=C/C(=O)NC1=CC=C(C=C1)C(O)=O
12111	DB17419	S-3APG	NC1=C2C(=O)N(C3CCC(=O)NC3=O)C(=O)C2=CC=C1
12112	DB17421	Trotabresib	Not Available
12113	DB17428	YK-4-279	Not Available

Розглянувши дані для яких наведене SMILES представлення, можна зробити висновок що деякі засоби не підійдуть для подальшого навчання так як мають білкову структуру та потребують іншого підходу підготовки даних до побудови моделі машинного навчання (**табл. 3.1.3.**)

Більш детально проаналізувавши таблицю 3.1.3 , були виявлені речовини, які за основною своєю дією належать до інших класів АТХ класифікатора ('DB01098', 'DB00641', 'DB01382') - це симвастатин (C10), розувастатин (C10), та глімідін, який є комбінацією глімепіриду та метформіну.

Табл. 3.1.3 Результат запиту до бази даних з таблицею 'describe' з наявним класом A10

Index	**id**	**Generic Name**	**Type**	**SMILES**
29	DB00030	Insulin human	Biotech	None
44	DB00046	Insulin lispro	Biotech	None
45	DB00047	Insulin glargine	Biotech	None
68	DB00071	Insulin pork	Biotech	None
192	DB00197	Troglitazone	Small Molecule	CC1=C(C)C2=C(CCC(C)(COC3=CC=C(CC4SC(=O)NC4=O)C...
...	...	...	...	...
12811	DB13406	Carbutamide	Small Molecule	CCCCNC(=O)NS(=O)(=O)C1=CC=C(N)C=C1
12850	DB13446	Guar gum	Small Molecule	Not Available CC1C(=O)C(=C(C(=CCC(C=CC=CC(=CC=CC(C)C(C(C)CCC2=CC=CC=C2)O)C)O)O1)O)C

13078	DB13675	Metahexamide	Small Molecule	CC1=CC=C(C=C1N)S(=O)(=O)NC(=O)NC1CCCCC1
13325	DB13928	Semaglutide	Biotech	None
14459	DB15072	Beinaglutide	Biotech	None

Таким чином відкинувши з вищезгаданого переліку речовини перелічені вище, було відібрано 46 молекул. Далі проаналізувавши речовини які не мають в таблиці DESCRIBE коректне значення SMILES , була додана структура камеді гуарової (див. табл. 3.3), також не додавались інсуліни ('DB00030', 'DB00046', 'DB00047', 'DB00071', 'DB01306', 'DB01307','DB01309', 'DB09456', 'DB09564'). а кінцевий набір даних має такий вигляд (табл. 3.1.4)

Табл. 3.1.4 Протидіабетичні засоби для побудови моделі машинного навчання

Id	**Generic Name**	**SMILES**
DB00197	Troglitazone	CC1=C(C)C2=C(CCC(C)(COC3=CC=C(CC4SC(=O)NC4=O)C=C3)O2)C(C)=C1O
DB00222	Glimepiride	CCC1=C(C)CN(C(=O)NCCC2=CC=C(C=C2)S(=O)(=O)NC(=O)N[C@H]2CC[C@H](C)CC2)C1=O
DB00284	Acarbose	[H]C(=O)[C@H](O)[C@@H](O)[C@]([H])(O[C@@]1([H])O[C@H](CO)[C@@]([H])(O[C@H]2O[C@H](C)[C@@H](N[C@@]3([H])C=C(CO)[C@@H](O)[C@H](O)[C@H]3O)[C@H](O)[C@H]2O)[C@H](O)[C@H]1O)[C@H](O)CO
DB00331	Metformin	CN(C)C(=N)NC(N)=N

DB00412	Rosiglitazone	CN(CCOC1=CC=C(CC2SC(=O)NC2=O)C=C1)C1=CC=CC=N1
DB00414	Acetohexamide	CC(=O)C1=CC=C(C=C1)S(=O)(=O)NC(=O)NC1CCCCC1
DB00491	Miglitol	OCCN1C[C@H](O)[C@@H](O)[C@H](O)[C@H]1CO
DB00672	Chlorpropamide	CCCNC(=O)NS(=O)(=O)C1=CC=C(Cl)C=C1
DB00731	Nateglinide	CC(C)[C@H]1CC[C@@H](CC1)C(=O)N[C@H](CC1=CC=CC=C1)C(O)=O
DB00839	Tolazamide	CC1=CC=C(C=C1)S(=O)(=O)NC(=O)NN1CCCCCC1
DB00912	Repaglinide	CCOC1=C(C=CC(CC(=O)N[C@@H](CC(C)C)C2=CC=CC=C2N2CCCCC2)=C1)C(O)=O
DB00914	Phenformin	NC(=N)NC(=N)NCCC1=CC=CC=C1
DB01016	Glyburide	COC1=C(C=C(Cl)C=C1)C(=O)NCCC1=CC=C(C=C1)S(=O)(=O)NC(=O)NC1CCCCC1
DB01067	Glipizide	CC1=NC=C(N=C1)C(=O)NCCC1=CC=C(C=C1)S(=O)(=O)NC(=O)NC1CCCCC1
DB01120	Gliclazide	[H][C@@]12CCC[C@]1([H])CN(C2)NC(=O)NS(=O)(=O)C1=CC=C(C)C=C1
DB01124	Tolbutamide	CCCCNC(=O)NS(=O)(=O)C1=CC=C(C)C=C1
DB01132	Pioglitazone	CCC1=CN=C(CCOC2=CC=C(CC3SC(=O)NC3=O)C=C2)C=C1
DB01251	Gliquidone	COC1=CC2=C(C=C1)C(C)(C)C(=O)N(CCC1=CC=C(C=C1)S(=O)(=O)NC(=O)NC1CCCCC1)C2=O

DB01252	Mitiglinide	[H][C@@](CC(=O)N1C[C@@]2([H])CCCC[C@@]2([H])C1)(CC1=CC=CC=C1)C(O)=O
DB01261	Sitagliptin	N[C@@H](CC(=O)N1CCN2C(C1)=NN=C2C(F)(F)F)CC1=CC(F)=C(F)C=C1F
DB01278	Pramlintide	[H]N[C@@H](CCCCN)C(=O)N[C@H]1CSSC[C@H](NC(=O)[C@@]([H])(NC(=O)[C@H](C)NC(=O)[C@@]([H])(NC(=O)[C@H](CC(N)=O)NC1=O)[C@@H](C)O)[C@@H](C)O)C(=O)N[C@@H](C)C(=O)N[C@@]([H])([C@@H](C)O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCNC(N)=N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C(C)C)C(=O)N[C@@H](CC1=CN=CN1)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)NCC(=O)N1CCC[C@H]1C(=O)N[C@@]([H])([C@@H](C)CC)C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N1CCC[C@H]1C(=O)N[C@@]([H])([C@@H](C)O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](C(C)C)C(=O)NCC(=O)N[C@@H](CO)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@]([H])([C@@H](C)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(N)=O
DB01289	Glisoxepide	CC1=CC(=NO1)C(=O)NCCC1=CC=C(C=C1)S(=O)(=O)NC(=O)NN1CCCCCC1

DB02383	Tolrestat	COC1=C(C2=CC=CC(C(=S)N(C)CC(O)=O)=C2C=C1)C(F)(F)F
DB04830	Buformin	CCCC\N=C(/N)N=C(N)N
DB04876	Vildagliptin	OC12CC3CC(C1)CC(C3)(C2)NCC(=O)N1CCC[C@H]1C#N
DB04878	Voglibose	OCC(CO)N[C@H]1C[C@](O)(CO)[C@@H](O)[C@H](O)[C@H]1O
DB06203	Alogliptin	CN1C(=O)C=C(N2CCC[C@@H](N)C2)N(CC2=C(C=CC=C2)C#N)C1=O
DB06292	Dapagliflozin	CCOC1=CC=C(CC2=C(Cl)C=CC(=C2)[C@@H]2O[C@H](CO)[C@@H](O)[C@H](O)[C@H]2O)C=C1
DB06335	Saxagliptin	[H][C@@]12C[C@]1([H])N([C@@H](C2)C#N)C(=O)[C@@H](N)C12CC3CC(CC(O)(C3)C1)C2
DB08882	Linagliptin	CC#CCN1C(=NC2=C1C(=O)N(CC1=NC3=C(C=CC=C3)C(C)=N1)C(=O)N2C)N1CCC[C@@H](N)C1
DB08907	Canagliflozin	[H][C@]1(O[C@H](CO)[C@@H](O)[C@H](O)[C@H]1O)C1=CC=C(C)C(CC2=CC=C(S2)C2=CC=C(F)C=C2)=C1
DB08962	Glibornuride	CC1=CC=C(C=C1)S(=O)(=O)NC(=O)N[C@@H]1[C@H](O)[C@]2(C)CC[C@H]1C2(C)C
DB09022	Benfluorex	CC(CC1=CC(=CC=C1)C(F)(F)F)NCCOC(=O)C1=CC=CC=C1
DB09038	Empagliflozin	OC[C@H]1O[C@H]([C@H](O)[C@@H](O)[C@@H]1O)C1=CC=C(Cl)C(CC2=CC=C(O[C@H]3CCOC3)C=C2)=C1

DB09198	Lobeglitazone	COC1=CC=C(OC2=NC=NC(=C2)N(C)CCOC2=CC=C(CC3SC(=O)NC3=O)C=C2)C=C1
DB11698	Ipragliflozin	OC[C@H]1O[C@H]([C@H](O)[C@@H](O)[C@@H]1O)C1=CC=C(F)C(CC2=CC3=CC=CC=C3S2)=C1
DB11827	Ertugliflozin	CCOC1=CC=C(CC2=CC(=CC=C2Cl)[C@]23OC[C@](CO)(O2)[C@@H](O)[C@H](O)[C@H]3O)C=C1
DB11950	Teneligliptin	[H][C@]1(CN[C@@]([H])(C1)C(=O)N1CCSC1)N1CCN(CC1)C1=CC(C)=NN1C1=CC=CC=C1
DB12214	Luseogliflozin	CCOC1=CC=C(CC2=CC([C@@H]3S[C@H](CO)[C@@H](O)[C@H](O)[C@H]3O)=C(OC)C=C2C)C=C1
DB12412	Gemigliptin	N[C@H](CN1CC(F)(F)CCC1=O)CC(=O)N1CCC2=C(C1)N=C(N=C2C(F)(F)F)C(F)(F)F
DB12509	Imeglimin	C[C@@H]1NC(N)=NC(=N1)N(C)C
DB12625	Evogliptin	CC(C)(C)OC[C@H]1N(CCNC1=O)C(=O)C[C@H](N)CC1=CC(F)=C(F)C=C1F
DB12713	Sotagliflozin	CCOC1=CC=C(CC2=CC(=CC=C2Cl)[C@@H]2O[C@H](SC)[C@@H](O)[C@H](O)[C@H]2O)C=C1
DB13406	Carbutamide	CCCCNC(=O)NS(=O)(=O)C1=CC=C(N)C=C1
DB13446	Guar gum	CC1C(=O)C(=C(C(=CCC(C=CC=CC(=CC=CC(C)C(C(C)CCC2=CC=CC=C2)O)C)O)O1)O)C
DB13675	Metahexamide	CC1=CC=C(C=C1N)S(=O)(=O)NC(=O)NC1CCCCC1

Для підготовки повного набору даних крім речовин з цільовою міткою y=1, які володіють протидіабетичною дією (табл. 3.4) потрібно сформувати набір даних з цільовою міткою y=-1, тобто сполуки які не належать до речовин з класом A10 ATX класифікатора, а також мають значення в полі ATC таблиці ATC при цьому має бути коректно заповнене поле SMILES таблиці DESCRIBE.

Таким чином на етапі збору даних про речовини та їх приналежності до ATX класифікації було відібрано 46 речовин які проявляють протидіабетичну дію та 2981 речовини які не відносяться до протидіабетичних засобів згідно ATX класифікації.

3.2. Конструювання молекулярних ознак. Трансформація та нормалізація даних

Конструювання молекулярних ознак (feature engineering) — важливий етап у підготовці даних для машинного навчання, особливо в задачах, пов'язаних з аналізом молекул [1; 23]

Цей процес ще називається "*Molecular Feature Engineering*" (інжиніринг молекулярних ознак) або "*Molecular Fingerprinting*" (молекулярні відбитки). Це важливий етап у хемоінформатиці та машинному навчанні для роботи з хімічними сполуками. Основні підходи можуть включати молекулярні дескриптори (Molecular Descriptors), такі як фізико-хімічні властивості, топологічні індекси, конституційні дескриптори, квантово-хімічні дескриптори тощо. А молекулярні відбитки (Molecular Fingerprints) можуть бути сформовані за допомогою ECFP (Extended-Connectivity Fingerprints) [67], MACCS Keys [20], Morgan Fingerprints [9; 47], Topological Fingerprints [11; 119], RDKit Fingerprints [63] або інші. Для цього процесу часто використовуються бібліотеки RDKit, Mordred [48], PaDEL-Descriptor [88], CDK (Chemistry Development Kit) [86]. У 2015 році дослідники з MTI представили алгоритм глибинного синтезу ознак (*Deep Feature Synthesis algorithm*) та показали його дієвість в інтерактивних змаганнях з науки про дані [1; 101].

3.2.1 Інжиніринг дескрипторів на основі класифікації ClassyFire

На етапі роботи для отримання дескрипторів було обрано комп'ютерну програму (ClassyFire), яка використовує лише хімічні структури та структурні особливості для автоматичного віднесення всіх відомих хімічних сполук до таксономії, що складається з >4800 різних категорій [8].

Вибрана хімічна таксономія складається з 11 різних рівнів (царство, надклас, клас, підклас і т.д.), кожна з яких визначається однозначними структурними правилами, які можна обчислити (рис. 3.2.1). Крім того, кожна категорія названа з використанням узгодженої номенклатури і описана на основі характерних загальних структурних властивостей сполук, які вона містить. Веб-сервер ClassyFire знаходиться у вільному доступі за адресою http://classyfire.wishartlab.com/. ClassyFire було використано для анотування понад 77 мільйонів сполук і вже інтегровано в інші програмні пакети для автоматичної генерації текстових описів та/або виведення біологічних властивостей понад 100 000 сполук [8].

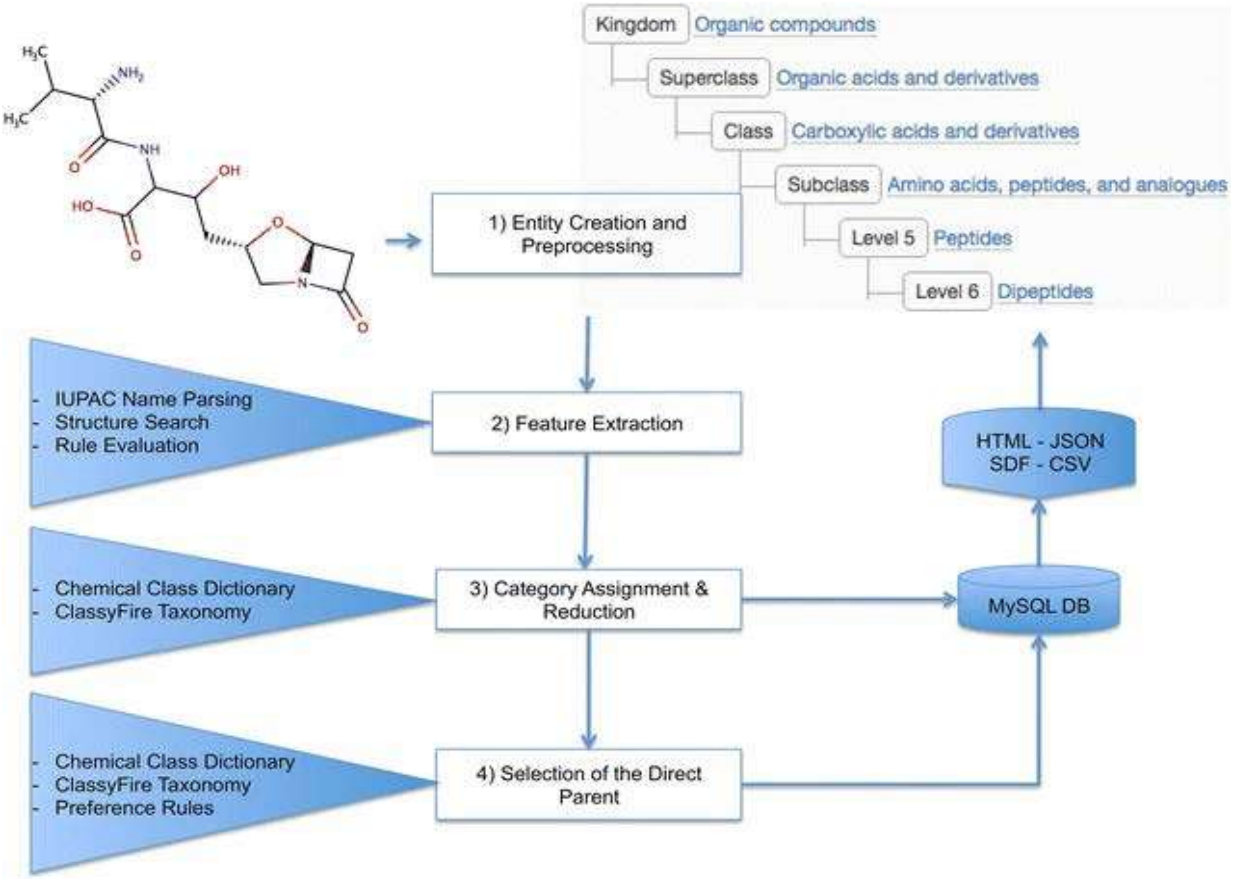


Рис. 3.2.1 Процес хімічної класифікації ClassyFire [8].

Також на веб-сервері є існуючий набір даних «DrugBank_5_classyfire_21_annotations.csv», проте було прийняте рішення скористатися API вказаного ресурсу та сформувати власний набір даних на основі SMILES речовин які мають класифікатор АТХ (див. підрозділ 3.1, додатки Г, Д).

Отримані таким чином дані є колекцією класифікацій хімічних сполук, де кожна сполука (ідентифікована ідентифікатором DrugBank, наприклад, DB00197, табл. 3.2.1.1) пов'язана з набором ідентифікаторів ChEBI (Chemical Entities of Biological Interest - хімічні речовини, що становлять біологічний інтерес) і відповідна їм хімічна таксономія (ChemOnt), що базується виключно на структурі. Ці класифікації забезпечують ієрархічне представлення хімічних властивостей і структурних особливостей сполук, що досліджуються.

Таблиця 3.2.1.1 Дескриптори троглітазону (DB00197) згідно ClassyFire

Дескриптор	**Опис**
CHEBI:22712	benzenes
CHEBI:35618	aromatic ether
CHEBI:50990	thiazolidenediones
CHEBI:35356	dicarboximide
CHEBI:50492	thiocarbonyl compound
CHEBI:36963	organooxygen compound
CHEBI:38104	oxacycle
CHEBI:38101	organonitrogen heterocyclic compound
CHEBI:33302	pnictogen molecular entity
CHEBI:50860	organic molecular entity

CHEBI:35352	organonitrogen compound
CHEBI:25701	organic oxide
CHEBI:72695	organic molecule
CHEBI:36586	carbonyl compound
CHEBI:38443	1-benzopyran
CHEBI:24431	chemical entity
CHEBI:33836	benzenoid aromatic compound
CHEBI:24532	organic heterocyclic compound
CHEBI:35622	thiazolidines
CHEBI:48891	thiazolidinone
CHEBI:25806	oxygen molecular entity
CHEBI:25698	ether
CHEBI:33261	organosulfur compound
CHEBI:51143	nitrogen molecular entity
CHEBI:22727	benzopyran
CHEMONTID:0000000	Organic compounds
CHEMONTID:0000002	Organoheterocyclic compounds
CHEMONTID:0000123	Benzopyrans
CHEMONTID:0003410	1-benzopyrans
CHEMONTID:0004742	Phenoxy compounds
CHEMONTID:0002341	Phenol ethers

CHEMONTID:0000228	Thiazolidinediones
CHEMONTID:0000128	Alkyl aryl ethers
CHEMONTID:0003033	Dicarboximides
CHEMONTID:0001368	Thiocarbamic acid derivatives
CHEMONTID:0000364	Organic carbonic acids and derivatives
CHEMONTID:0004140	Oxacyclic compounds
CHEMONTID:0004139	Azacyclic compounds
CHEMONTID:0004557	Organopnictogen compounds
CHEMONTID:0000278	Organonitrogen compounds
CHEMONTID:0003940	Organic oxides
CHEMONTID:0004150	Hydrocarbon derivatives
CHEMONTID:0001831	Carbonyl compounds

Проаналізувавши частотне входження дескрипторів (табл 3.2.1.2) засобів що володіють протидіабетичною дією, встановлено, що дані сполуки мають у своїй більшості дескриптори **CHEBI:24431 та CHEBI:36963, CHEBI:51143 та CHEBI:35352, також 40 з 46 сполук у своєму складі містять CHEBI:33836 (**benzenoid aromatic compound**) та 27 і 25 відповідно** organic heterocyclic compound та organosulfur compound (**CHEBI:24532, CHEBI:33261**)

Таблиця 3.2.1.2 **Входження дескрипторів для засобів з протидіабетичною дією**

Дескриптор	**Опис**	**Частота**
CHEBI:72695	organic molecule	46

CHEMONTID:0004150	Hydrocarbon derivatives	46
CHEMONTID:0000000	Organic compounds	46
CHEBI:50860	organic molecular entity	46
CHEBI:24431	chemical entity	46
CHEBI:25806	oxygen molecular entity	43
CHEBI:36963	organooxygen compound	43
CHEBI:51143	nitrogen molecular entity	40
CHEBI:35352	organonitrogen compound	40
CHEBI:33836	benzenoid aromatic compound	39
CHEMONTID:0004557	Organopnictogen compounds	38
CHEBI:33302	pnictogen molecular entity	38
CHEBI:22712	Benzenes	37
CHEMONTID:0003940	Organic oxides	36
CHEBI:25701	organic oxide	36
CHEBI:36586	carbonyl compound	31
CHEMONTID:0001831	Carbonyl compounds	28
CHEBI:24532	organic heterocyclic compound	27
CHEBI:33261	organosulfur compound	25
CHEBI:32952	Amine	25

CHEMONTID:0002279	Benzene and substituted derivatives	22
CHEMONTID:0002448	Benzenoids	22
CHEMONTID:0004139	Azacyclic compounds	20
CHEBI:38101	organonitrogen heterocyclic compound	20
CHEMONTID:0000031	Benzenesulfonamides	17
CHEMONTID:0000490	Sulfonylureas	17
CHEBI:32877	primary amine	17
CHEMONTID:0000270	Organosulfonic acids and derivatives	17
CHEBI:35850	Sulfone	17
CHEBI:35358	Sulfonamide	17
CHEBI:33552	sulfonic acid derivative	17
CHEBI:76983	N-sulfonylurea	17
CHEMONTID:0003137	Aminosulfonyl compounds	17
CHEMONTID:0004233	Benzenesulfonyl compounds	17
CHEBI:25698	Ether	15
CHEMONTID:0000364	Organic carbonic acids and derivatives	15
CHEBI:30879	Alcohol	13
CHEBI:33822	organic hydroxy compound	13
CHEBI:26191	Polyol	13

CHEBI:27024	Toluenes	12
CHEMONTID:0000323	Organooxygen compounds	12
CHEMONTID:0000128	Alkyl aryl ethers	11
CHEBI:35618	aromatic ether	11
CHEBI:33659	organic aromatic compound	11
CHEBI:35681	secondary alcohol	11
CHEMONTID:0004144	Heteroaromatic compounds	11
CHEBI:33709	amino acid	10
CHEMONTID:0000284	Aniline and substituted anilines	10
CHEMONTID:0001661	Secondary alcohols	10
CHEBI:22562	Anilines	10
CHEBI:16670	Peptide	10
CHEBI:36684	organohalogen compound	10

Для зазначених дескрипторів, розрахувавши критерій χ^2 відносно дескрипторів, які відповідно до даної класифікації зустрічаються у наборі даних сполук, що згідно з класифікацією АТХ не проявляють протидіабетичної дії, встановлено, що статистично значущими є дескриптори, наведені в таблиці 3.2.1.2

Тож після підготовки даних для ML з дескрипторами розрахованими за класифікацією ClassyFire розподілялись дані у співвідношенні 2883 речовини що не відноситься до класу антидіабетичних засобів та 46 що володіють протидіабетичною дією (всього 2929).

3.2.2 Інжиніринг дескрипторів на основі бібліотеки mordred

Окремо, в якості ознак використовувались дескриптори, згенеровані за допомогою бібліотеки Mordred, що забезпечує числове представлення хімічних властивостей молекул (Рис. 3.2.2).

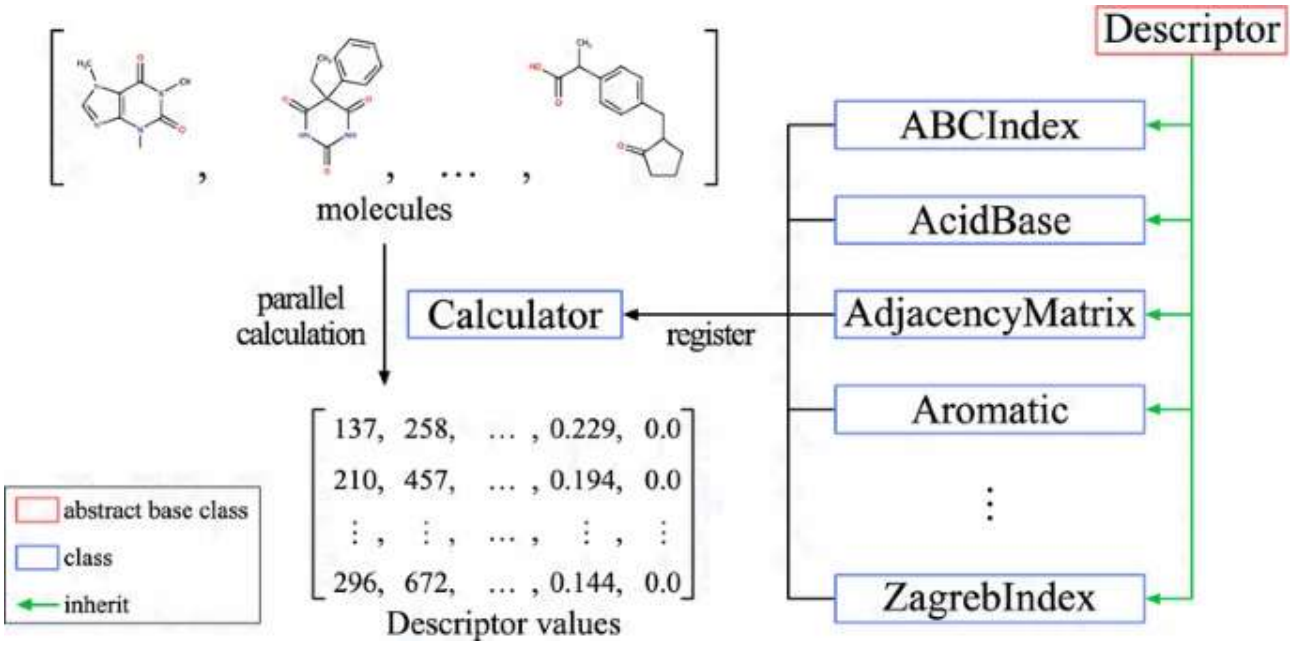
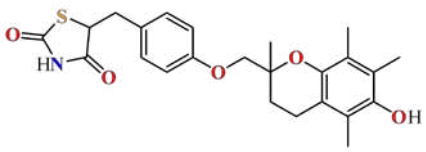
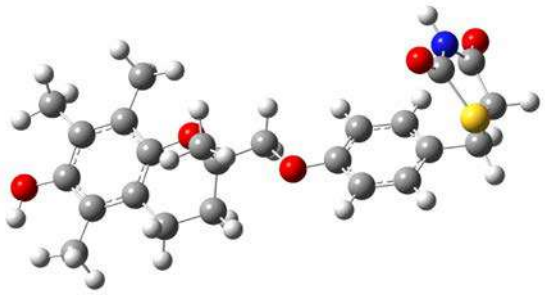


Рис. 3.2.2 Алгоритми молекулярних дескрипторів Mordred []

В процесі отримання оптимізованих структур серед протидіабетичних засобів були виключені акарбоза та прамлінтид, на розрахунки яких потрібно більше ресурсів а ніж для решти лікарських засобів, аналогічний підхід використовувався для речовин які не відносяться до класу протидіабетичних засобів згідно класифікації АТС

Для прикладу зобразимо 2-вимірну та 3-вимірну (розраховану Р6 методом) структури троглітазону на основі яких будуть розраховуватись дескриптори (табл. 3.2.2.1)

Таблиця 3.2.2.1 **Хімічна структура троглітазону (DB00197)**

SMILES	2D	3D
CC1=C(C)C2=C(CCC(C)(COC3=CC=C(CC4SC(=O)NC4=O)C=C3)O2)C(C)=C1O		

Серед розрахованих дескрипторів зустрічаються кількість ароматичних атомів, кількість гетероатомів, автокореляція Моро-Брото, коефіцієнт Морана, спектральне абсолютне відхилення від матриці Бариша та багато інших, весь перелік дискрипторів можна переглянути за посиланням - https://mordred-descriptor.github.io/documentation/master/descriptors.html. Всього для розрахунків дескрипторів MORDRED використовує 44 модулі(функції) для 2D та 6 для 3D вимірної структури (табл. 3.2.2.2).

Таблиця 3.2.2.2 **Розподіл 2D/3D молекулярних дескрипторів MORDRED**

Dim	module	name	constructor	description
2D	44	1613	1613	1613
3D	6	213	213	213

Для троглітазону на основі 3-вимірної структури за допомогою mordred було вирахувано 1036 дескрипторів з 1826 можливих (табл. 3.2.2.3). При цьому mordred розраховує 1613 дескрипторів для 2D та 213 для 3D вимірної структури (табл. 3.9).

Таким чином після підготовки даних для ML з дескрипторами розрахованими за mordred розподілялись дані у співвідношенні 2461 речовина що не відноситься до класу антидіабетичних засобів та 45 що володіють протидіабетичною дією (всього 2506).

Таблиця 3.2.2.3 **Фрагмент розрахованих дескрипторів для троглітазону (DB00197)**

Дескриптор	**Опис**	**Значення**
SpAbs_A	дескриптор матриці суміжності з аргументом SpAbs	39.481227
SpMax_A	дескриптор матриці суміжності з аргументом SpMax	2.506828
SpDiam_A	дескриптор матриці суміжності з аргументом SpDiam	5.013626
SpAD_A	дескриптор матриці суміжності з аргументом SpAD	39.481227
SpMAD_A	дескриптор матриці суміжності з аргументом SpMAD	1.273588
...	...	...
Wpol	Індекс полярності Вінера	53.000000
Zagreb1	Загребський індекс (версія 1)	170.000000
Zagreb2	Загребський індекс (версія 2)	203.000000
mZagreb1	модифікований Загребський індекс (версія 2)	11.284722
mZagreb2	модифікований Загребський індекс (версія 2)	6.569444

3.3 Стратегії подолання дисбалансу класів при підготовці тренувальних даних

В останні роки класифікація незбалансованих наборів даних є однією з основних проблем для методів машинного навчання [40; 49; 64; 65].

Ряд алгоритмів класифікаційного навчання, таких як дерево рішень, нейронна мережа зворотного поширення, мережа Байєса, найближчого сусіда, машини опорних векторів і нещодавно описана асоціативна класифікація, добре розроблені і успішно застосовуються в багатьох прикладних областях. Однак, незбалансований розподіл класів даних створює серйозні труднощі для більшості алгоритмів навчання класифікаторів, які передбачають відносно збалансований розподіл [78].

Одним із методів, які використовуються у вирішенні цієї проблеми, є зважування класів (Class Weighting). Цей метод дає змогу врахувати дисбаланс між класами в процесі навчання моделі. Він ґрунтується на ідеї того, що модель буде штрафувати сильніше за помилки в класі-меншості, заохочуючи більш точне передбачення [5].

Зважування класів може бути ефективним, коли дисбаланс не є екстремальним, недоліками є обмеження лінійними алгоритмами, а саме може погано працювати з нелінійними алгоритмами машинного навчання, такими як дерева рішень або глибокі нейронні мережі, або ж ефективність може змінюватися залежно від набору даних і моделі.

Ще одним з підходів для вирішення проблеми з дисбалансом класів є SMOTE (Synthetic Minority Over-sampling Technique) - це метод, який генерує синтетичні приклади класу меншин для збалансування набору даних. Він створює нові точки даних шляхом інтерполяції між існуючими прикладами класу меншини [31; 44; 95; 107; 109].

Добре працює з алгоритмами, які не припускають лінійності даних, при цьому уникає втрати інформації, на відміну від випадкової вибірки. Але може вимагати значних обчислень на великих наборах даних, а ефективність залежить від правильного вибору параметрів [51; 58; 99].

Метод випадкової вибірки (Random Over Sampling) передбачає випадкове дублювання прикладів з класу меншин у наборі даних. Реалізується просто - випадковим чином дублюючи приклади з міноритарного класу. Може спричинити надмірне дублювання прикладів міноритарних класів та може призвести до втрати релевантної інформації.

Таким чином був обраний метод RandomUnderSampler, але для оцінки моделі random_state змінювався для кожного окремого разу (всього 100 разів, див додаток Є, функція evaluate_models_with_stats)

РОЗДІЛ 4.
ПОБУДОВА, ОЦІНКА ТА ЗАСТОСУВАННЯ МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ

4.1. Навчання та валідація моделей. Аналіз продуктивності моделей та вибір оптимального підходу

Для навчання використовувались наступні алгоритми: логістична регресія (у варіантах без регуляризації, з L1-регуляризацією та регуляризацією Elastic Net), Ridge класифікація, наївний байєсівський алгоритм (мультиноміальний та модель Бернуллі), метод опорних векторів (для лінійно роздільних класів), метод k-найближчих сусідів, випадковий ліс, дерево рішень, лінійний дискримінантний аналіз, класифікатор на основі стохастичного градієнтного спуску та градієнтний бустинг (додаток Ж).

Детальний аналіз результатів навчання моделей за даними на основі дескрипторів за класифікацією ClassyFire на основі метрик (табл. 4.1.1) показав, що найкраща загальна продуктивність спостерігалась у Gradient Boosting, яка демонструє найбільш збалансовані показники з високою точністю (0.897 ± 0.057) та ROC AUC (0.951 ± 0.047), Linear SVM, Ridge Regression та SGD Classifier показують дуже високі метрики, але мають значне стандартне відхилення (±0.14-0.22), що вказує на нестабільність їх роботи.

Найстабільніші моделі (найменші стандартні відхилення) виявились Elastic Net з accuracy 0.853 ± 0.061 та Gradient Boosting з accuracy 0.897 ± 0.057 і Random Forest з ROC AUC 0.957 ± 0.035

Найменше помилкових спрацьовувань (FP) спостерігалось у Linear SVM: 0.75 ± 2.236, Ridge Regression: 0.81 ± 2.241, SGD Classifier: 0.9 ± 2.29. Найменше пропущених випадків (FN) у моделей Logistic Regression: 1.19 ± 0.982, BernoulliNB: 1.45 ± 1.048, Gradient Boosting: 1.52 ± 1.132 (таблиця 4.2).

Найбільш найгірші показники виявились у k-NN який має найнижчий recall (0.661 ± 0.111) та F1 Score (0.705 ± 0.094), а ось LDA показує низьку точність (0.691 ± 0.086) та precision (0.653 ± 0.114)

Якщо як пріоритет вибрати стабільність і збалансовану продуктивність то залишити вибір слідує за Gradient Boosting або Random Forest. Якщо ж важлива максимальна точність і допустима деяка нестабільність то Linear SVM або Ridge Regression. Коли потрібно вибрати між між точністю та швидкістю то Elastic Net або Logistic Regression.

Цікавими виявились наївні байєсівські класифікатори (BernoulliNB та Multinomial NB) які показують хороший recall, але гірший precision, Tree-based моделі (Random Forest, Decision Tree, Gradient Boosting) демонструють більш збалансовані метрики, а щодо лінійних моделі (SVM, Ridge, SGD) - показують найвищі метрики, але з великою варіативністю.

Таким чином високі стандартні відхилення у Linear SVM, Ridge Regression та SGD Classifier вказують на можливе перенавчання моделі або чутливість до конкретних розбиттів даних, k-NN показує найгірші результати, що може вказувати на неоптимальну структуру даних для цього методу або необхідність додаткової настройки параметрів.

Таблиця 4.1.1 **Порівняльні характеристики метрик моделей машинного навчання**

Model	Accuracy	Precision	Recall	F1 Score	ROC AUC
BernoulliNB	0.719 ± 0.078	0.661 ± 0.106	0.895 ± 0.081	0.755 ± 0.081	0.881 ± 0.06
Decision Tree	0.855 ± 0.065	0.859 ± 0.092	0.849 ± 0.101	0.849 ± 0.075	0.856 ± 0.069
Elastic Net	0.853 ± 0.061	0.837 ± 0.092	0.874 ± 0.084	0.851 ± 0.067	0.925 ± 0.05
Gradient Boosting	0.897 ± 0.057	0.901 ± 0.085	0.893 ± 0.081	0.893 ± 0.063	0.951 ± 0.047

LDA	0.691 ± 0.086	0.653 ± 0.114	0.816 ± 0.098	0.719 ± 0.088	0.797 ± 0.087
Lasso Regression	0.815 ± 0.067	0.828 ± 0.105	0.796 ± 0.1	0.806 ± 0.078	0.912 ± 0.052
Linear SVM	0.943 ± 0.143	0.928 ± 0.214	0.927 ± 0.228	0.922 ± 0.219	0.95 ± 0.178
Logistic Regression	0.866 ± 0.071	0.835 ± 0.103	0.916 ± 0.07	0.87 ± 0.071	0.938 ± 0.048
Multinomial Naive Bayies	0.772 ± 0.081	0.724 ± 0.113	0.877 ± 0.09	0.788 ± 0.086	0.894 ± 0.058
Random Forest	0.851 ± 0.07	0.884 ± 0.092	0.806 ± 0.119	0.838 ± 0.088	0.957 ± 0.035
Ridge Regression	0.94 ± 0.142	0.925 ± 0.214	0.925 ± 0.228	0.92 ± 0.218	0.948 ± 0.178
SGD Classifier	0.924 ± 0.134	0.942 ± 0.151	0.899 ± 0.203	0.908 ± 0.181	0.958 ± 0.134
k-NN	0.731 ± 0.082	0.773 ± 0.123	0.661 ± 0.111	0.705 ± 0.094	0.823 ± 0.076

Таблиця 4.1.2 **Порівняння моделей класифікації за даними матриці помилок**

Model	True Positives (TP)	False Positives (FP)	True Negatives (TN)	False Negatives (FN)
BernoulliNB	12.47 ±	6.42 ±	7.66 ±	1.45 ± 1.048

	2.56	2.114	2.475	
Decision Tree	11.8 ± 2.433	1.93 ± 1.265	12.15 ± 2.484	2.12 ± 1.402
Elastic Net	12.16 ± 2.473	2.36 ± 1.382	11.72 ± 2.483	1.76 ± 1.164
Gradient Boosting	12.4 ± 2.374	1.36 ± 1.15	12.72 ± 2.47	1.52 ± 1.132
LDA	11.33 ± 2.374	6.07 ± 2.203	8.01 ± 2.307	2.59 ± 1.408
Lasso Regression	11.06 ± 2.335	2.31 ± 1.447	11.77 ± 2.399	2.86 ± 1.45
Linear SVM	13.07 ± 4.016	0.75 ± 2.236	13.33 ± 3.911	0.85 ± 2.607
Logistic Regression	12.73 ± 2.411	2.56 ± 1.743	11.52 ± 2.627	1.19 ± 0.982
Multinomial Naive Bayes	12.22 ± 2.545	4.69 ± 1.983	9.39 ± 2.53	1.7 ± 1.159
Random Forest	11.2 ± 2.494	1.46 ± 1.201	12.62 ± 2.546	2.72 ± 1.646
Ridge Regression	13.05 ± 4.021	0.81 ± 2.241	13.27 ± 3.939	0.87 ± 2.608
SGD Classifier	12.7 ± 3.889	0.9 ± 2.29	13.18 ± 4.069	1.22 ± 2.368
k-NN	9.17 ±	2.77 ±	11.31 ±	4.75 ± 1.794

	2.156	1.644	2.692	

Аналізуючи результати, можна зробити висновки щодо ефективності різних моделей машинного навчання на основі метрик, таких як Accuracy, Precision, Recall, F1 Score та ROC AUC. Таким чином на основні спостереження найкращими моделями за F1 Score та Accuracy були Random Forest що демонструє найкращі результати за F1 Score (0.859 ± 0.059) та Accuracy (0.843 ± 0.063) та свідчить про високу збалансованість між Precision та Recall, а також про загальну точність моделі.

Gradient Boosting також показує високі результати: F1 Score (0.831 ± 0.064) та Accuracy (0.808 ± 0.07) дана модель має найвищий Recall (0.96 ± 0.048), що свідчить про здатність правильно класифікувати позитивні класи.

Logistic Regression має Recall = 1.0 ± 0.0, що означає, що модель правильно класифікує всі позитивні приклади. Однак низький Precision (0.512 ± 0.084) та Accuracy (0.524 ± 0.082) свідчать про велику кількість помилково позитивних прогнозів (FP).

Random Forest та Gradient Boosting також мають високий Recall (0.979 ± 0.035 та 0.96 ± 0.048 відповідно), що робить їх ефективними для задач, де важливо мінімізувати помилки типу FN.

Random Forest має найвищий Precision (0.769 ± 0.086), що свідчить про низьку кількість помилково позитивних прогнозів (FP).

Gradient Boosting та LDA також демонструють високий Precision (0.738 ± 0.094 та 0.719 ± 0.098 відповідно).

Elastic Net та Lasso Regression показують дуже низькі значення Accuracy, Precision, Recall та F1 Score. Наприклад, Elastic Net має Recall = 0.084 ± 0.066, що свідчить про нездатність правильно класифікувати позитивні приклади. Ці моделі не підходять для даної задачі класифікації.

Що стосується Linear SVM, Ridge Regression та SGD Classifier то вони мають великі стандартні відхилення для багатьох метрик, що свідчить про

нестабільність їх роботи на різних підмножинах даних. Наприклад, Linear SVM має Accuracy = 0.679 ± 0.19, що вказує на значну мінливість результатів.

Найвищі значення ROC AUC демонструють Random Forest (0.963 ± 0.035) та Gradient Boosting (0.931 ± 0.05), що підтверджує їхню здатність ефективно розділяти класи. Найнижчі значення ROC AUC у Elastic Net (0.372 ± 0.103) та Lasso Regression (0.367 ± 0.104), що підтверджує їхню неефективність.

Таким чином Random Forest та Gradient Boosting є найкращими кандидатами для використання, оскільки вони демонструють високі значення Accuracy, Precision, Recall, F1 Score та ROC AUC. LDA та Decision Tree також показують непогані результати і можуть бути альтернативою.

Elastic Net та Lasso Regression потребують перегляду гіперпараметрів або використання інших підходів, оскільки їх результати незадовільні.

Linear SVM, Ridge Regression та SGD Classifier можуть бути покращені шляхом стабілізації їх роботи, наприклад, за рахунок зменшення дисперсії даних або вибору кращих гіперпараметрів.

Якщо основним критерієм є мінімізація помилок типу FN то Logistic Regression, Random Forest та Gradient Boosting є найкращими варіантами.

Таблиця 4.1.3 **Порівняльні характеристики метрик для моделей машинного навчання**

Model	Accuracy	Precision	Recall	F1 Score	ROC AUC
BernoulliNB	0.537 ± 0.083	0.521 ± 0.089	0.942 ± 0.057	0.666 ± 0.078	0.572 ± 0.06
Decision Tree	0.774 ± 0.075	0.715 ± 0.103	0.917 ± 0.063	0.798 ± 0.074	0.775 ± 0.069
Elastic Net	0.495 ± 0.09	0.472 ± 0.364	0.084 ± 0.066	0.138 ± 0.103	0.372 ± 0.103

Gradient Boosting	0.808 ± 0.07	0.738 ± 0.094	0.96 ± 0.048	0.831 ± 0.064	0.931 ± 0.05
LDA	0.777 ± 0.079	0.719 ± 0.098	0.917 ± 0.067	0.801 ± 0.071	0.847 ± 0.078
Lasso Regression	0.487 ± 0.088	0.411 ± 0.372	0.061 ± 0.054	0.103 ± 0.086	0.367 ± 0.104
Linear SVM	0.679 ± 0.19	0.661 ± 0.298	0.641 ± 0.382	0.599 ± 0.317	0.698 ± 0.326
Logistic Regression	0.524 ± 0.082	0.512 ± 0.084	1.0 ± 0.0	0.673 ± 0.074	0.641 ± 0.109
Multinomial Naive Bayes	0.706 ± 0.084	0.645 ± 0.103	0.92 ± 0.07	0.754 ± 0.079	0.81 ± 0.083
Random Forest	0.843 ± 0.063	0.769 ± 0.086	0.979 ± 0.035	0.859 ± 0.059	0.963 ± 0.035
Ridge Regression	0.669 ± 0.197	0.65 ± 0.332	0.656 ± 0.437	0.572 ± 0.356	0.641 ± 0.36
SGD Classifier	0.645 ± 0.103	0.612 ± 0.204	0.618 ± 0.307	0.581 ± 0.236	0.671 ± 0.151
k-NN	0.639 ± 0.084	0.605 ± 0.105	0.807 ± 0.088	0.686 ± 0.085	0.651 ± 0.099

Таблиця 4.1.4 **Зіставлення моделей класифікації за даними матриці помилок**

Model	True Positives (TP)	False Positives (FP)	True Negatives (TN)	False Negatives (FN)
BernoulliNB	12.73 ± 2.256	11.72 ± 2.257	1.77 ± 1.024	0.78 ± 0.746
Decision Tree	12.37 ± 2.182	4.98 ± 1.944	8.51 ± 2.042	1.14 ± 0.853
Elastic Net	1.13 ± 0.884	1.26 ± 0.981	12.23 ± 2.356	12.38 ± 2.304
Gradient Boosting	12.95 ± 2.153	4.62 ± 1.797	8.87 ± 2.2	0.56 ± 0.686
LDA	12.37 ± 2.237	4.9 ± 1.967	8.59 ± 2.421	1.14 ± 0.888
Lasso Regression	0.82 ± 0.716	1.17 ± 0.933	12.32 ± 2.352	12.69 ± 2.29
Linear SVM	9.31 ± 6.058	4.48 ± 4.249	9.01 ± 5.832	4.2 ± 4.273
Logistic Regression	13.51 ± 2.281	12.85 ± 2.194	0.64 ± 0.732	0.0 ± 0.0
Multinomial Naive Bayes	12.44 ± 2.311	6.88 ± 2.212	6.61 ± 2.132	1.07 ± 0.924
Random Forest	13.23 ± 2.296	3.97 ± 1.586	9.52 ± 2.338	0.28 ± 0.451

Ridge Regression	9.6 ± 6.724	5.03 ± 5.256	8.46 ± 6.717	3.91 ± 4.952
SGD Classifier	8.91 ± 5.178	5.0 ± 3.088	8.49 ± 4.844	4.6 ± 3.309
k-NN	10.92 ± 2.255	7.16 ± 2.145	6.33 ± 2.02	2.59 ± 1.207

4.2. Генерація нових молекул на основі SMILES-представлення

Проблематика досліджень зосереджена на розробці та застосуванні генеративних моделей штучного інтелекту для оптимізації процесу відкриття нових лікарських засобів, що є критично важливим напрямком через складність і ресурсомісткість традиційних підходів. Традиційні методи, такі як високопродуктивний скринінг, вимагають значних людських зусиль, матеріальних ресурсів і фінансових інвестицій, тоді як інноваційні підходи на основі ШІ пропонують ефективніші альтернативи для навігації у величезному та дискретному хімічному просторі, що становить одну з головних проблем у фармацевтичній галузі [24, 44]. Останні роки позначились суттєвим прогресом у застосуванні глибокого навчання для генеративного моделювання молекул, причому особливу увагу привертають методи, що використовують варіаційні автокодери (VAE), генеративні змагальні мережі (GAN) та архітектури на основі трансформерів, які здатні не лише вивчати закономірності в існуючих хімічних структурах, але й генерувати принципово нові молекули з бажаними фармакологічними властивостями [24].

Архітектура Taiga та система REINVENT 4 представляють сучасні трансформаторні моделі, які використовують двоетапний підхід: спочатку розглядають проблему як задачу мовного моделювання для передбачення молекулярних структур на основі рядків SMILES, а потім застосовують навчання з підкріпленням для оптимізації молекулярних властивостей, таких як

КЕД. Ці методи дозволяють моделям засвоїти основні правила хімії та ефективніше генерувати молекули з бажаними характеристиками, демонструючи покращення в оптимізації від 2 до понад 20 відсотків порівняно з існуючими підходами, що було підтверджено експериментами як на наборах даних із молекулами свинцю, так і на випадкових молекулах [40, 44]. Система REINVENT 4, як генеративна система з відкритим вихідним кодом, спеціально розроблена для проектування малих молекул і поєднує рекурентні нейронні мережі та архітектури трансформаторів, що вбудовуються в загальні алгоритми оптимізації машинного навчання, дозволяючи здійснювати дизайн de novo, заміну R-груп, дизайн бібліотек, дизайн лінкерів, скафолд-хопінг та оптимізацію молекул [40].

Фундаментальні підпроблеми, які вирішуються генеративними моделями, включають: вибір оптимального представлення молекул (1D, 2D або 3D), розробку цільових функцій для спрямування процедури генерації, а також створення алгоритмів для ефективного дослідження молекулярного простору [7, 34]. Вибір представлення молекул є особливо критичним, оскільки не існує універсального підходу, і дослідники мають адаптувати представлення відповідно до конкретних цілей; цільові функції відіграють ключову роль у орієнтуванні процедури генерації на рішення, які вирішують визначену проблему, тоді як оціночні функції використовуються для оцінки множини рішень, запропонованих методом, виходячи за межі критеріїв, сформульованих цільовою функцією [34]. Еволюція методів молекулярної генерації протягом останніх 30 років демонструє значний прогрес у підходах до дослідження молекулярного простору, причому сучасні методи все більше покладаються на глибоке навчання та генеративні моделі для ефективного вирішення складних хімічних задач [34, 7].

Практична значущість таких досліджень полягає у потенційному революціонуванні процесу розробки ліків шляхом автоматизованої генерації нових молекул з заданими властивостями, що може суттєво скоротити час та витрати на розробку медичних препаратів, а також забезпечити можливість

конструювання молекул для специфічних біологічних мішеней навіть без попередньої цільової анотації тренувальних сполук [45]. Інноваційні підходи, такі як обумовлення генеративних змагальних мереж за допомогою транскриптомних даних, дозволяють автоматично конструювати молекули, які мають високу ймовірність індукувати бажаний транскриптомний профіль, що представляє собою альтернативний підхід, який поєднує хімію та біологію на довгому і складному шляху відкриття ліків [45]. Дослідження показують, що молекули, сконструйовані за допомогою таких генеративних моделей, є більш подібними до активних сполук, ніж ті, що ідентифіковані за подібністю експресійних сигнатур генів, що відкриває нові перспективи для створення високоефективних лікарських засобів з мінімальними побічними ефектами [45].

Процес відкриття лікарських засобів є складним і ресурсномістким процесом, що вимагає значних людських зусиль, матеріальних ресурсів і фінансових інвестицій. Однією з головних проблем у відкритті нових ліків є велика та дискретна природа хімічного простору. Однак нещодавні досягнення в галузі штучного інтелекту та використання великих даних почали змінювати цей ландшафт. Підходи глибокого навчання стали потужною альтернативою традиційним методам грубої сили, таким як високопродуктивний скринінг. Зокрема, генеративні моделі привертають значну увагу і застосовуються для розробки молекул лікарських засобів de novo, що дозволяє генерувати нові молекули з бажаними властивостями. За останні роки було розроблено кілька методів генерації молекул ліків на основі генеративних моделей. До них відносяться варіації варіаційного автокодера (VAE), такі як JT-VAE [25].

У статті пропонується Taiga - трансформаторну архітектуру для генерації молекул із заданими властивостями. Використовуючи двоетапний підхід, ми спочатку розглядаємо проблему як задачу мовного моделювання передбачення наступної лексеми, використовуючи рядки SMILES. Потім ми використовуємо навчання з підкріпленням для оптимізації молекулярних властивостей, таких як КЕД. Цей підхід дозволяє нашій моделі засвоїти основні правила хімії та легше оптимізувати молекули з потрібними властивостями. Наша оцінка Taiga, яка

була проведена з використанням декількох наборів даних і завдань, показує, що Taiga порівнянна або навіть перевершує найсучасніші базові лінії для оптимізації молекул, з покращенням КЕД від 2 до більш ніж 20 відсотків. Покращення було продемонстровано як на наборах даних, що містять молекули свинцю, так і на випадкових молекулах. Ми також показали, що завдяки двоетапному навчанню Taiga здатна генерувати молекули з вищими показниками біологічних властивостей, ніж та ж модель без підкріплення [45].

REINVENT 4 - це сучасна генеративна система штучного інтелекту з відкритим вихідним кодом для проектування малих молекул. Програмне забезпечення використовує рекурентні нейронні мережі та архітектури трансформаторів для генерації молекул. Ці генератори легко вбудовуються в загальні алгоритми оптимізації машинного навчання, навчання з перенесенням, навчання з підкріпленням і навчання за навчальною програмою. REINVENT 4 уможливлює та полегшує дизайн de novo, заміну R-груп, дизайн бібліотек, дизайн лінкерів, скафолд-хопінг та оптимізацію молекул. У цій статті представлено огляд програмного забезпечення та описано його дизайн[41].

У цій главі ми пропонуємо презентацію дослідницьких проблем молекулярної генерації з використанням методів зі сфери штучного інтелекту. Цю мету можна розділити на три фундаментальні підпроблеми. Перша проблема - це представлення молекул. Не існує універсального представлення молекул. Тому необхідно вибирати представлення, адаптоване до мети, яку потрібно вирішити. Друга проблема полягає у спрямуванні та оцінюванні генеративних методів. Роль цільової функції полягає в тому, щоб орієнтувати процедуру генерації на рішення, які вирішують проблему. Також можна визначити оціночні функції, метою яких є оцінка множини рішень, запропонованих методом, поза критерієм, сформульованим цільовою функцією. Третя проблема полягає в розробці методів молекулярної генерації. За останні 30 років було запропоновано багато методів. Ми вирішили представити методи відповідно до того, як вони генерують рішення і досліджують молекулярний

простір. У цьому розділі ми підкреслюємо вирішальну роль і взаємозв'язок цих трьох підпроблем [35].

У цій статті ми розглянемо останні досягнення в галузі генеративного молекулярного дизайну та обговоримо міркування щодо інтеграції цих моделей у реальні кампанії з молекулярного відкриття. Спочатку ми розглянемо вибір дизайну моделі, необхідний для розробки та навчання генеративної моделі, включаючи загальні 1D, 2D і 3D представлення молекул і типові архітектури нейронних мереж для генеративного моделювання [7].

У цій статті ми представляємо генеративну модель, яка поєднує системну біологію та молекулярний дизайн, обумовлюючи генеративну змагальну мережу за допомогою транскриптомних даних. Таким чином, ми можемо автоматично конструювати молекули, які мають високу ймовірність індукувати бажаний транскриптомний профіль. За умови наявності сигнатури генної експресії бажаного стану ця модель здатна конструювати активні молекули для бажаних мішеней без попередньої цільової анотації тренувальних сполук. Молекули, сконструйовані за допомогою цієї моделі, є більш подібними до активних сполук, ніж ті, що ідентифіковані за подібністю експресійних сигнатур генів. Загалом, цей метод являє собою альтернативний підхід, що поєднує хімію та біологію на довгому і складному шляху відкриття ліків [46].

Для генерації молекулярних структур було вибране готове рішення на основі статті "Generating Molecules Using a Char-RNN in PyTorch" в якій описано метод генерації молекул за допомогою рекурентних нейронних мереж (RNN), які працюють на рівні символів (Char-RNN), з використанням SMILES-нотації [10]. Char-RNN як тип рекурентної нейронної мережі, що функціонує на рівні окремих символів, є оптимально пристосованим для обробки SMILES-нотації, оскільки ці рядки складаються з послідовностей символів, які кодують хімічні структури, а модель навчається передбачати наступний символ у послідовності, що дозволяє генерувати нові молекули зі збереженням їхньої хімічної валідності.

Методологія даного дослідження базується на використанні набору даних ChEMBL, який містить мільйони SMILES-рядків із широким спектром хімічних структур, при цьому обробка даних включає декілька критичних етапів: читання SMILES-рядків з файлу, створення словника символів для кодування кожного символу в числовий індекс, та перетворення SMILES-рядків на послідовності індексів для ефективного навчання моделі. Автор застосовує архітектуру, що базується на шарах LSTM (Long Short-Term Memory), які спеціально оптимізовані для обробки послідовних даних завдяки здатності зберігати довгострокові залежності в послідовностях, що є критично важливим для генерування коректних SMILES-рядків з дотриманням правил хімічної валентності та структурної цілісності.

Технічна реалізація навчання моделі включає використання функції втрат CrossEntropyLoss, яка оптимально підходить для задач класифікації, де кожен символ SMILES розглядається як окремий клас, а оптимізатор Adam застосовується для динамічного оновлення ваг моделі під час навчання. Після завершення процесу навчання стає можливою генерація нових SMILES-рядків шляхом послідовного передбачення наступного символу на основі попередніх, що дозволяє створювати унікальні молекулярні структури, які можуть представляти потенційно корисні хімічні сполуки з новими фармакологічними властивостями, та значно прискорює процес дизайну нових лікарських засобів порівняно з традиційними методами високопродуктивного скринінгу.

Модифікований код для генерації SMILES послідовностей наданий в додатку З, який включає генерацію SMILES послідовностей на основі 46 молекул для подальшого навчання, а також перевірку валідності SMILES за допомогою rdkit.

Таким чином було навчено нейромережу та згенеровано 1179 нових SMILES послідовностей (рис. 4.2).

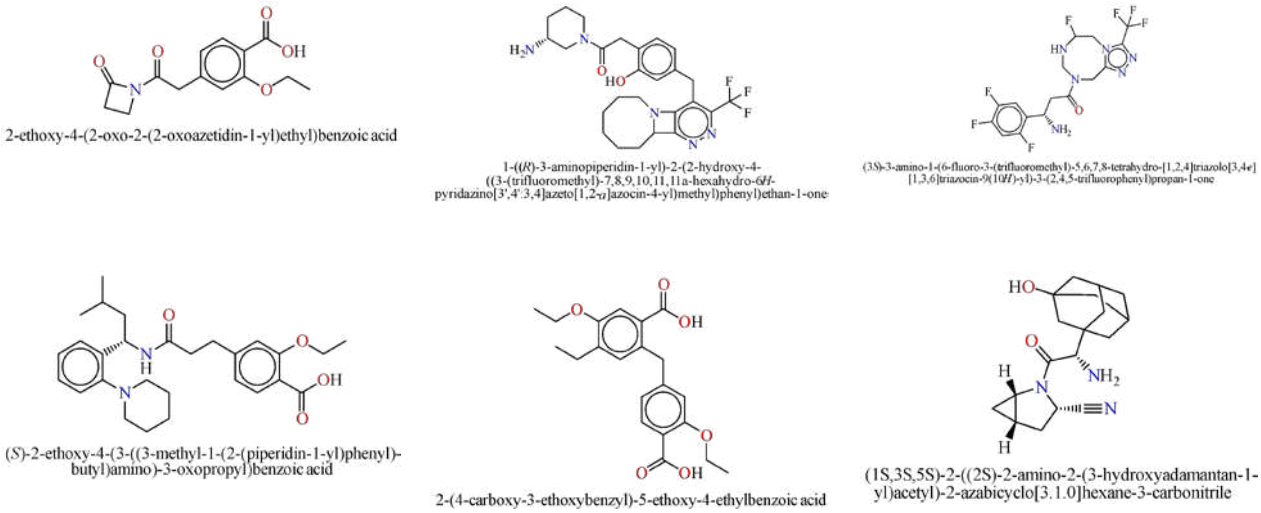


Рис. 4.2 Структура деяких згенерованих молекул на основі SMILES

Щодо новизни отриманих структур, зазначені послідовності, ймовірно, вже є відомими. Тому на наступному етапі, при відборі згенерованих структур, молекули, які були відібрані, піддавалися перевірці в базі даних Reaxys.

4.3. Відбір згенерованих молекул за допомогою побудованих моделей прогнозування

Після генерації SMILES послідовностей, аналогічним чином до того як були підготовлені дескриптори за допомогою MORDRED та ClassyFire (додатки Г, Д, Е) готувався набір даних для прогнозування ймовірності протидіабетичної дії у згенерованих молекулах.

Таким чином проаналізувавши дані (табл. 4.3.1-4.3.3) щодо ймовірностей протидіабетичної дії різних сполук, розрахованих за допомогою різних моделей машинного навчання. Для вибору 10 найкращих сполук, було використано критерії серед яких високі середні показники ймовірності протидіабетичної дії за всіма моделями, консистентність високих показників між різними моделями, а перевага надавалась сполукам із найвищими показниками за найбільшою кількістю моделей.

Таблиця 4.3.1 **Ймовірності протидіабетичної дії за предикторами розрахованими MORDRED**

№ з/п згенерованої сполуки	Моделі ML		
	Gradient Boosting	Random Forest	Logistic Regression
108	1	0,65	0,65996
176	1	0,65	0,65996
887	1	0,65	0,65996
860	1	0,65	0,65996

Таблиця 4.3.2 **Ймовірності протидіабетичної дії за предикторами розрахованими ClassyFire**

№ з/п згенерованої сполуки	Моделі ML			
	Gradient Boosting	Random Forest	Linear SVM	Logistic Regression
52	0,872455	0,81	0,553283	0,93189
226	0,872455	0,81	0,553283	0,93189
283	0,872455	0,81	0,553283	0,93189
379	0,872455	0,81	0,553283	0,93189
442	0,872455	0,81	0,553283	0,93189
546	0,872455	0,81	0,553283	0,93189
663	0,872455	0,81	0,553283	0,93189
676	0,872455	0,81	0,553283	0,93189

681	0,872455	0,81	0,553283	0,93189
683	0,872455	0,81	0,553283	0,93189
692	0,872455	0,81	0,553283	0,93189
716	0,872455	0,81	0,553283	0,93189
773	0,872455	0,81	0,553283	0,93189
841	0,872455	0,81	0,553283	0,93189
903	0,872455	0,81	0,553283	0,93189
949	0,872455	0,81	0,553283	0,93189
1036	0,872455	0,81	0,553283	0,93189
451	0,872455	0,81	0,553283	0,93189
843	0,872455	0,81	0,553283	0,93189

Таблиця 4.3.3 **Ймовірності протидіабетичної дії за предикторами розрахованими ClassyFire**

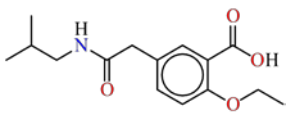
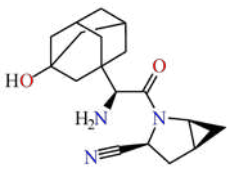
№ з/п згенерованої сполуки	Gradient Boosting	Random Forest	Decision Tree	LDA
20	0.99998	0.92	1.0	0.999742
22	0.99998	0.92	1.0	0.999742
30	0.99998	0.92	1.0	0.999742
38	0.99998	0.92	1.0	0.999742
84	0.99998	0.92	1.0	0.999742

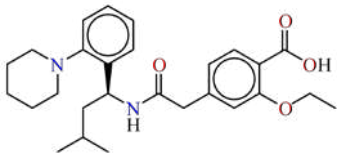
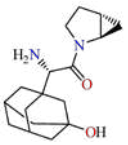
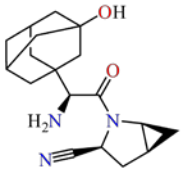
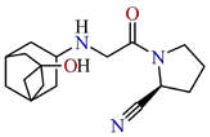
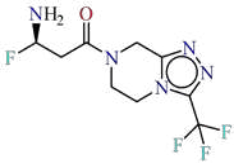
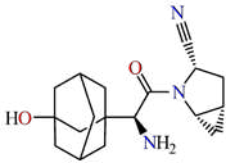
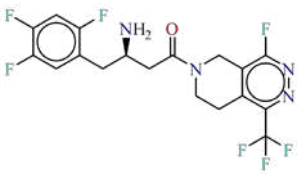
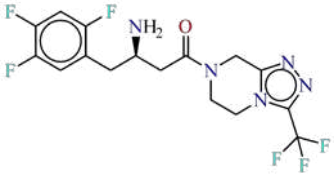
147	0.99998	0.92	1.0	0.999742
248	0.99998	0.92	1.0	0.999742
298	0.99998	0.92	1.0	0.999742
321	0.99998	0.92	1.0	0.999742
351	0.99998	0.92	1.0	0.999742
447	0.99998	0.92	1.0	0.999743
540	0.99998	0.93	1.0	1.000000
569	0.99998	0.92	1.0	0.999742
628	0.99998	0.92	1.0	0.999743
633	0.99998	0.92	1.0	0.999742
655	0.99998	0.92	1.0	0.999742
668	0.99998	0.92	1.0	0.999742
714	0.99998	0.92	1.0	0.999742
752	0.99998	0.92	1.0	0.999742
767	0.99998	0.92	1.0	0.999742
785	0.99998	0.92	1.0	0.999742
821	0.99998	0.92	1.0	0.999741
896	0.99998	0.92	1.0	0.999742
902	0.99998	0.92	1.0	0.999742
906	0.99998	0.92	1.0	0.999742
954	0.99998	0.92	1.0	0.999742

967	0.99998	0.95	1.0	0.999944
1003	0.99998	0.92	1.0	0.999742
1021	0.99998	0.92	1.0	0.999741
1098	0.99998	0.92	1.0	0.999742
1117	0.99998	0.92	1.0	0.999742
1138	0.99998	0.92	1.0	0.999742
1163	0.99998	0.92	1.0	0.999743

Таким чином найбільш перспективними сполуками за протидіабетичною дією виявились SMILES послідовність **№967** (Gradient Boosting (0,99998), Random Forest (0,95), Decision Tree (1,0), LDA (0,999944) та SMILES послідовність **№540** (Gradient Boosting (0,99998), Random Forest (0,93), Decision Tree (1,0), LDA (1,0).

Таблиця 4.3.4 **Хімічна структура SMILES послідовностей які мають найвищі показники ймовірності протидіабетичної дії за різними моделями ML**

№ з/п згенерованої сполуки	SMILES	2D структура
20	N(C(=O)Cc1cc(c(OCC)cc1)C(O)=O)CC(C)C	
52	C12(CC3(CC(CC(C3)C2)C1)O)[C@H](N)C(N1[C@@H](C[C@@H]2C[C@@H]21)C#N)=O	

108	c1c(OCC)c(ccc1CC(=O)N[C@H](c1c(cccc1)N1CCCCC1)CC(C)C)C(O)=O	
447	C1CN([C@@H]2C[C@H]21)C(=O)[C@@H](N)C12CC3CC(CC(O)(C3)C1)C2	
451	C1C2(CC3CC(C2)CC1(C3)O)[C@@H](C(N1[C@H](C#N)C[C@H]2[C@@H]1C2)=O)N	
540	C1C2(CC3CC(CC1(C3)C2)NCC(=O)N1CCC[C@H]1C#N)O	
628	F[C@H](CC(N1CCn2c(nnc2C(F)(F)F)C1)=O)N	
843	C12CC3(CC(O)(CC(C3)C1)C2)[C@H](N)C(=O)N1[C@H](C#N)C[C@@H]2C[C@H]12	
967	c12c(nnc(C(F)(F)F)c2CCN(C1)C(C[C@@H](Cc1c(cc(F)c(F)c1)F)N)=O)F	
1163	N1(C(C[C@@H](Cc2cc(F)c(cc2F)F)N)=O)Cc2nnc(C(F)(F)F)n2CC1	

Дещо нижчі показники ймовірностей наявності протидіпбетичної дії спостерігаються у SMILES послідовністях **№447** (Gradient Boosting (0,99998),

Random Forest (0,92), Decision Tree (1,0), LDA (0,999743)), **№628** - показники: Gradient Boosting (0,99998), Random Forest (0,92), Decision Tree (1,0), LDA (0,999743), **№1163** - показники: Gradient Boosting (0,99998), Random Forest (0,92), Decision Tree (1,0), LDA (0,999743), **№52** - показники: Gradient Boosting (0,872455), Random Forest (0,81), Linear SVM (0,553283), Logistic Regression (0,93189), **№108** - показники: Gradient Boosting (1), Random Forest (0,65), Logistic Regression (0,65996), **№20** - показники: Gradient Boosting (0,99998), Random Forest (0,92), Decision Tree (1,0), LDA (0,999742), **№451** - показники: Gradient Boosting (0,872455), Random Forest (0,81), Linear SVM (0,553283), Logistic Regression (0,93189), **№843** - показники: Gradient Boosting (0,872455), Random Forest (0,81), Linear SVM (0,553283), Logistic Regression (0,93189) (таблиця 4.3.4)

РОЗДІЛ 5.
РЕТРОСИНТЕТИЧНИЙ АНАЛІЗ ТА МЕТОДОЛОГІЯ ДО ФАРМАКОЛОГІЧНОГО СКРИНІНГУ ПРОТИДІАБЕТИЧНИХ ЗАСОБІВ ДЛЯ ВІДІБРАНИХ МОЛЕКУЛ

5.1. Планування синтезу сполук та ретросинтетичний аналіз

Планування синтезу - це процес, за допомогою якого хімік або комп'ютер визначає, як синтезувати певну сполуку. Зазвичай це здійснюється шляхом ретросинтетичного аналізу, коли потрібну сполуку ітеративно розщеплюють на проміжні продукти або менші прекурсори, доки не будуть знайдені відомі або доступні для придбання будівельні блоки. Такий аналіз був започаткований Корі та ін. і традиційно проводився вручну або за допомогою експертних систем, що використовують закодовані вручну правила. З розвитком глибокого навчання в останнє десятиліття сфера ретросинтетичних програмних засобів зазнала швидких змін. Тепер складні та автоматичні алгоритми мають потенціал для забезпечення ретросинтетичного аналізу з ширшою сферою застосування та кращою точністю [22].

В ході дослідження встановлено, що молекули 20, 540, 967 не зареєстровані в базі даних Reaxys.

Далі аналіз наявності згенерованих молекул та ретросинтетичних шляхів їх одержання проводився за допомогою інструментарію ChemAIRS (https://www.chemical.ai/chemairs/).

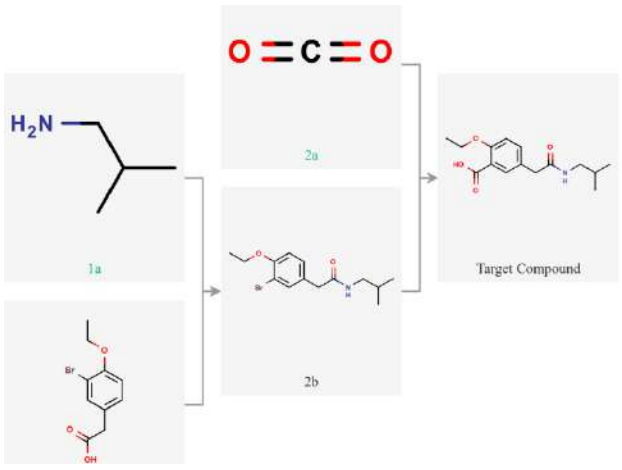


Рис 5.1.1 Схема синтезу сполуки 20 запропонована ресурсом ChemAIRS

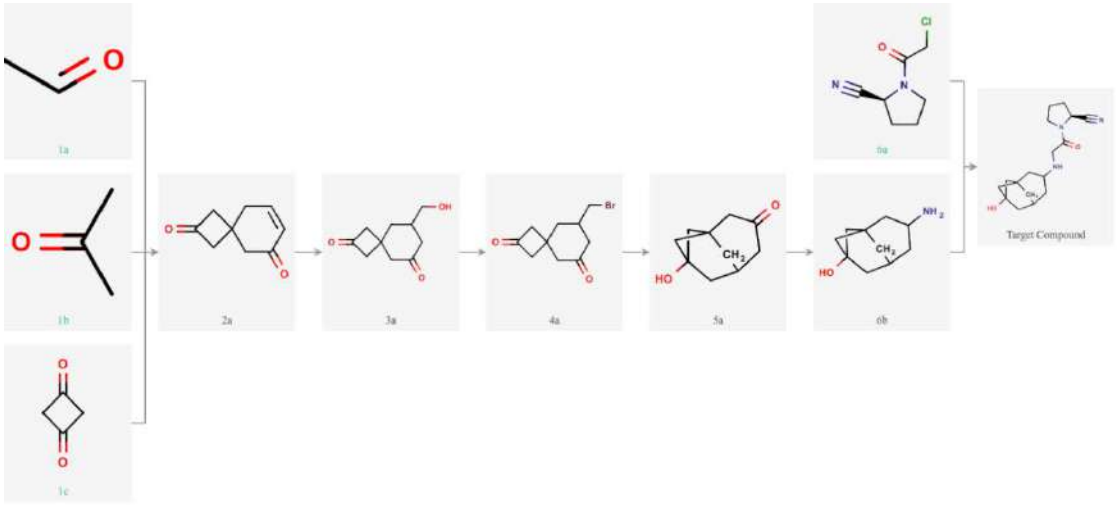


Рис 5.1.2 Схема синтезу сполуки 540 запропонована ресурсом ChemAIRS

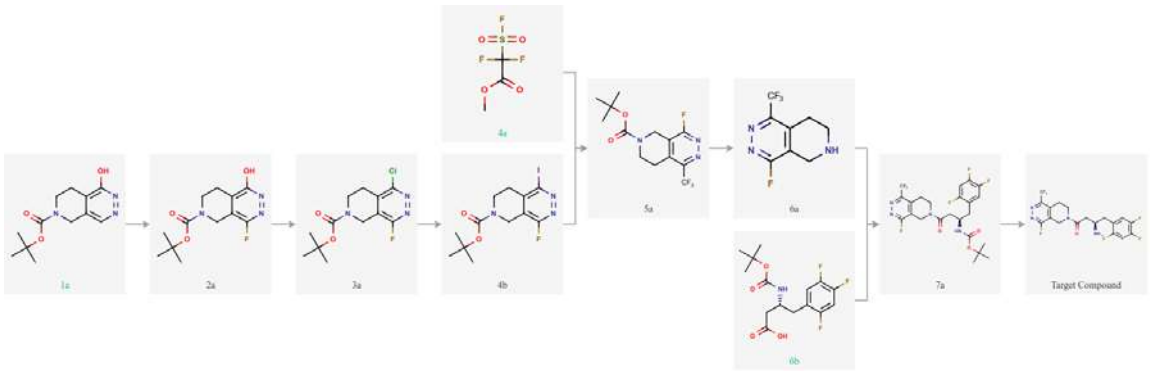


Рис 5.1.3 Схема синтезу сполуки 967 запропонована ресурсом ChemAIRS

5.2. Підготовка до фармакологічного тестування: дизайн скринінгових досліджень

Скринінгові дослідження нових протидіабетичних сполук потребують ретельного планування етапів тестування, оцінки необхідної кількості лабораторних тварин і реагентів, а також вибору адекватних статистичних методів. Нижче представлений багатоетапний дизайн фармакологічного скринінгу для оцінки протидіабетичних властивостей сполук 20, 540 та 967.

5.2.1 Характеристика біологічних тест-систем

У експериментальних дослідженнях заплановано використання нів на момент початку експерименту), які пройшли 12 щурів лінії Wistar (або білих нелінійних) масою 170-230 г (віком 10-12) тижневу акліматизацію протягом 14 діб [7].

Лабораторні тварини надходитимуть з розплідника. Догляд, утримання та годування тварин здійснюватиметься за стандартними умовами стабільного мікроклімату віварію БДМУ з дотриманням 12-годинного світлового циклу. Годування лабораторних тварин проводитиметься стандартизованим раціонним кормом «Резон-1» КП-120-1 з необмеженим доступом до їжі та води в умовах, що виключають вплив стресових чинників.

Утримання щурів здійснюватиметься у стандартних конвенціональних клітках з полікарбонату розмірами 610×435×215 мм або 335×235×190 мм по 5 (10) тварин у клітці.

За добу до введення досліджуваних речовин та розчинника всі тварини підлягатимуть огляду лікарем ветеринарної медицини. До дослідження включатимуться виключно клінічно здорові тварини. Рандомізація тварин здійснюватиметься випадковим способом. Індивідуальне маркування тварин проводитиметься 1% розчином брильянтового зеленого.

Перед початком проведення досліджень комісія з питань біоетики БДМУ має розглянути та затвердити план-протокол дослідження, а також усі процедури, пов'язані з утриманням тварин, гуманним поводженням з тваринами та їх використанням в експерименті та надати витяг з протоколу засідання, його № та дату засідання комісії.

5.2.2 Об'єкти досліджень

В якості розчинника для сполук планується використовувати натрію хлорид розчин для інфузій 9мг/мл флакон 400 мл. Для приготування суспензії (за умов, що сполуки не розчинні у воді) застосовуватиметься твін-80 – флакон 100 мл х.ч. Препарати промарковані для доклінічних досліджень, використовувати гумові рукавички.

Суспензію чи розчин досліджуваних сполук готують за годину до введення (табл. 5.2.2)

Табл. 5.2.2. Основні сполуки даної методики

Найменування компонентів	Кількість	Всього
Досліджувана речовина		
СПОЛУКА 20, 540, 967	10 мг/мл	100 мг
Допоміжні речовини:		
Натрію хлорид	9 мг/мл	10 мл
Твін-80	1 мг/мл	10 мг

Методика приготування суспензії: В асептичних умовах наважку 100 мг сполуки переносять до ступки й додають 10 мг твіну-80, ретельно перетирають пестиком до утворення однорідної маси. Додають 10 мл фізіологічного розчину до отримання суспензії та переносять до промаркованої посудини.

5.2.3 Статистична обробка отриманих результатів

Розраховані середні арифметичні (M) та стандартні похибки середньої (±m), порівняти між медіаною (Me) та міжквартильним розмахом (Q_1:Q_3). Статистичну значимість міжгрупових відмінностей за даними експериментів встановити за допомогою t-критерію Стьюдента, U-критерію Уітні-Манна. Провести розвідувальний аналіз перемінних (Exploratory Data Analysis, EDA) які описують розподіл кожної змінної за допомогою pandas-profiling (MIT License, v. 2.9.0).

5.2.4 Дослідження безпечності

Дослідження профілю безпечності нових хімічних сполук, які розглядаються як потенційні лікарські засоби, становить одне з фундаментальних завдань сучасної фармацевтичної індустрії. Визначення гострої токсичності новосинтезованих речовин є основоположним етапом для встановлення токсичних та, відповідно, терапевтично ефективних доз. Відбір

фармакологічно активних сполук базується на основоположному принципі медицини «*Primum non nocere*» («Насамперед не нашкодь»). У зв'язку з цим проблема розробки нових препаратів з низьким токсикологічним профілем та високою селективністю дії залишається вкрай актуальною.

В основі методу лежить пропозиція використовувати досліджувані речовини в дозах, котрі розміщені по логарифмічній шкалі з інтервалом 0,1, а всі можливі достовірні результати $ЛД_{50}$ та їх похибки розраховані попередньо по програмі пробіт-аналізу.

Досліджувані сполуки вводяться внутрішньошлунково в дозах, котрі розміщені по логарифмічній шкалі (табл. 5.2.4.1).

Табл. 5.2.4.1 Логарифмічна шкала

0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	**1**,0	**1**,1	*і т.д.*
1	1,2589	1,5848	1,9952	2,5118	3,1622	3,9810	5,0118	6,3095	7,9432	10	12,589	*і т.д.*

LD_{50} та їх середньоквадратичне відхилення при дослідженні 4-х сусідніх доз впливу по 2 спостереження на кожну дозу визначали за табл. 5.2.4.2.

Табл. 5.2.4.2 LD_{50} та їх середні похибки при дослідженні 4-х доз впливу по 2 спостереження на кожну дозу

Послідовність ефектів				Показник	Натуральне значення доз									
					10,0	12,58	15,84	20,00	25,11	31,62	39,81	50,11	63,09	79,43
0	0	1	2	LD_{50}	15,5	19,5	24,6	30,9	38,9	49,0	61,6	77,6	97,7	123,0
				S	1,5	1,8	2,3	2,9	3,7	4,6	5,8	7,3	9,2	11,6
0	0	2	1	LD_{50}	16,8	21,1	26,6	33,4	42,1	53,1	66,8	84,2	106,0	133,2
				S	2,9	3,6	4,5	5,7	7,2	9,0	11,3	14,2	17,9	22,4
0	0	2	2	LD_{50}	14,2	17,9	22,5	28,3	35,7	45,0	56,6	71,4	89,8	113,1
				S	1,1	1,4	1,8	2,3	2,8	3,5	4,5	5,6	7,1	8,9
0	1	0	2	LD_{50}	15,3	19,3	24,3	30,6	38,6	49,6	62.4	77,0	96,9	124,5
				S	2,1	2,6	3,3	4,2	5,2	6,6	8,3	12,2	15,3	19.7
0	1	1	2	LD_{50}	14,2	17,9	22,5	28,4	35,7	45,0	56,6	71,3	90,0	113,1
				S	1,7	2,2	2,7	3,4	4,3	5,4	6,8	8,6	10,8	13.6
0	1	2	1	LD_{50}	15,3	19,3	24,3	30,6	38,5	48,5	62,4	77,0	96,9	121,7
				S	3,5	4,4	5,6	7,0	8,8	11,0	14,0	17,5	22,0	27,6
0	1	2	2	LD_{50}	13,2	16,6	20,9	26,3	33,1	41,7	52,5	66,1	83,4	104.7
				S	1,3	1,6	2,0	2,6	3,2	4,0	5,1	6,4	8,1	10,2

0	2	1	2	LD_{50}	12,1	15,2	19,2	24,2	30,4	38,2	48,2	60,7	76,4	96.2
				S	2,6	3,5	4,1	5,2	6,5	8,2	10,4	13,0	16,5	20.7
1	0	1	2	LD_{50}	13,8	17,4	21,9	27,6	34,8	43,2	55,1	69,4	87,0	110,0
				S	2,7	3,4	4,3	5,3	6,8	8,5	10,7	13,5	17,0	21,4
1	0	2	2	LD_{50}	12,5	15,7	19,8	25,0	31,4	39,5	49,8	62,7	79,0	99,5
				S	2,2	2,7	3,5	4,3	5,5	6,9	8,7	10,9	13,7	17,3
1	1	0	2	LD_{50}	14,6	18,2	22,9	28,9	36,3	45,8	57,6	72,6	91,4	114,9
				S	4,5	5,9	7,2	9,2	11,7	14,6	17,4	23,1	29,2	37,1

Попередньо провівши прогноз токсичності побудувати табличні дані з початковою дозою для вивчення гострої токсичності, а наступна та попередня дози будуть визначені по результатам введення речовини та заносяться до табл. 5.2.4.3

Табл. 5.2.4.3 Попередній прогноз токсичності

Доза, мг/кг						
Дата введення						
Ефект Вижило Загинуло(доба)	_____	_____	_____	_____	_____	_____

Перед оральним введенням досліджуваної речовини тварини повинні голодувати протягом ночі. Після перорального введення допуск до їжі надається не раніше, ніж через 3-4 год. Об'єм приготованого розчину та дозування сполук провести згідно табл. 5.2.4.4

Табл. 5.2.4.4 Основні показники досліджень

№ клітки	№ тварини	Дата	Дослідна група 1 (n=2)	Дослідна група 2 (n=2)	Дослідна група 3 (n=2)
			Сполука 20	Сполука 540	Сполука 967
	Вага,г				
	1				
	2				
	Маркування				
	1				
	2				
	Об'єм розчину, мл (1мл/0,25гр)				
	1				
	2				
	Доза сполуки, мг/кг				

№ клітки	№ тварини	Дата	Дослідна група 1 (n=2)	Дослідна група 2 (n=2)	Дослідна група 3 (n=2)
			Сполука 20	Сполука 540	Сполука 967
	1				
	2				
	Загибель тварин				
	1				
	2				
Дихання					
	1				
	2				
Рухова активність					
	1				
	Рухова активність				
	1				
	2				
Судоми					
	1				
	2				
Офтальмологічні симптоми					
	1				
	2				
Салівація					
	1				
	2				
Пілоерекція					
	1				
	2				
Тонус м'язів					
	1				
	2				
Показники стану ШКТ (екскременти)					
	1				
	2				
Блювання					
	1				
	2				
Діурез					
	1				
	2				
Стан шкіри					
	1				
	2				

Отримані дані занести до журналу реєстрації експериментальних досліджень віварію БДМУ.

На підставі отриманих в експерименті даних про відсоток загибелі тварин залежно від величин доз побудувати криву летальності, в якій відсоток смертності (сигмоідна крива) трансформується в пробіт-пряму в координатах «Пробіти – доза (мг/кг)» методом найменших квадратів.

Розрахунки LD_0, LD_{16}, LD_{50}, LD_{84} провести згідно табличного методу Прозоровського (табл. 5.2.4.2), пробіт аналізу та методом найменших квадратів, а порівняння провести інтерполяційним поліномом Лагранжа.

Для того, щоб охарактеризувати потенційну та реальну небезпеки у виникненні і розвитку гострого смертельного отруєння розрахувати токсикометричні параметри (таблиця 5.2.4.5).

Табл. 5.2.4.5 Токсикометричні параметри

Показники	Значення
Величина абсолютної токсичності, 1/LD50	
Діапазон смертельних доз, LD84/LD16	
Тангенс кута нахилу кривої летальності, tg α	
Інтегральний показник токсичності, 1/LD50·tg α	
Функція кута нахилу S	
Сумарний показник небезпечності, 1/(LD50·S)	

Для розрахунку максимально переносимої дози (DL_0) для людини , у випадку використання для клінічних випробувань, застосувати константи біологічної дії за Риболовлевим, а дані перерахунків для людини навести до табл. 5.2.4.6

Табл. 5.2.4.6 Показники переносимої дози для людини

LD_0, мг/кг	LD_{16}, мг/кг	LD_{84}, мг/кг	LD_{50}, мг/кг	±m, мг/кг

У випадку загибелі дослідних тварин, провести їх розтин з макроскопічнии описом тварин з гострим токсичним ураженням та отримані дані занести до журналу реєстрації розтинів дослідних тварин.

Після розтину візуально констатується наявність чи відсутність рідини у черевній та грудній порожнинах, оцінюється її кількість, колір, ступінь прозорості, запах. Далі вивчається стан розташованих у цих порожнинах органів. Звертається увага на розмір, форму, колір, взаєморозташування органів та інші особливості.

Вагу тварин та органів занести до табл. 5.2.4.7

Табл. 5.2.4.7 Показники розтинів дослідних тварин

Група, Тварина №	маса тіла, г	довжина, мм	маса тимусу, мг	маса печінки, мг	маса правої нирки, мг	маса лівої нирки, мг	маса селезінки, мг
Група	Дослідна група 1						
Тварина1							
Тварина2							
Група	Дослідна група 2						
Тварина1							
Тварина2							
Група	Дослідна група 3						
Тварина1							
Тварина2							
Група	Дослідна група 4						
Тварина1							
Тварина2							
Група	Дослідна група 5 (Інтактна)						
Тварина1							
Тварина2							
Тварина 3							
Тварина 4							

Група, Тварина №	маса правої надниркової залози, мг	маса лівої надниркової залози, мг	маса серця, мг	маса легенів, мг	маса підшлункової залози, мг	маса лівого яєчка, мг	маса правого яєчка, мг
Група	Дослідна група 1						
Тварина1							
Тварина2							
Група	Дослідна група 2						
Тварина1							
Тварина2							
Група	Дослідна група 3						
Тварина1							
Тварина2							
Група	Дослідна група 4						
Тварина1							
Тварина2							
Група	Дослідна група 5 (Інтактна)						
Тварина1							
Тварина2							
Тварина 3							
Тварина 4							

У випадку виявлених патологічних змін органи підлягають подальшому гістологічному дослідженню, а фіксація проводиться в забуференому 10% формаліні.

5.2.5 Гіпоглікемічна активність

Гіпоглікемічну дію похідних 1,2,4 - тріазолу оцінювали при внутрішньочеревному тесті толерантності до глюкози (ВЧТТГ) [1]. ВЧТТГ відтворюється шляхом навантаження тварин глюкозою в дозі 2 г/кг від маси тіла щура.

Дані рівня глюкози занести до табл. 5.2.5

Табл. 5.2.5 Показники рівня глюкози

Група, Тварина №	До введення глюкози за 30 хв, ммоль/л	Під час введення глюкози, ммоль/л	Під час введення сволуки, ммоль/л	Через 1 годину, ммоль/л	Через 2 години, ммоль/л	Через 3 години, ммоль/л	Через 6 годин, ммоль/л
Група	Дослідна група 1 (сполука 20)						
Тварина1							
Тварина2							
Тварина 3							
Тварина 4							
Тварина 5							
Тварина 6							
Група	Дослідна група 2 (сполука 540)						
Тварина1							
Тварина2							
Тварина 3							
Тварина 4							
Тварина 5							
Тварина 6							
Група	Дослідна група 3 (сполука 967)						
Тварина1							
Тварина2							
Тварина 3							
Тварина 4							
Тварина 5							
Тварина 6							
Група	Дослідна група 4 (глібенкламід, 5 мг/кг)						
Тварина1							
Тварина2							
Тварина 3							
Тварина 4							
Тварина 5							
Тварина 6							
Група	Дослідна група 5 (Інтактна, натрію хлорид 0.9%)						
Тварина1							

Тварина2							
Тварина 3							
Тварина 4							
Тварина 5							
Тварина 6							

ВИСНОВКИ

У магістерській роботі реалізовано та апробовано комплексний підхід до моделювання нових хімічних сполук з потенційною протидіабетичною дією на основі алгоритмів машинного навчання. Отримані результати дозволяють сформулювати наступні висновки:

1. Проведено аналіз сучасних підходів та перспектив застосування алгоритмів машинного навчання у фармацевтичній і медичній практиці. Встановлено, що методи машинного навчання є перспективним інструментом для прискорення та підвищення ефективності процесу розробки нових лікарських засобів, зокрема протидіабетичних препаратів. Визначено основні напрями застосування ML-технологій у фармацевтичній галузі та їх переваги порівняно з традиційними методами розробки ліків.
2. Розроблено методику збору, первинної обробки, нормалізації та категоризації даних про лікарські засоби. Створено комплексний набір даних, що включає інформацію про 46 протидіабетичних препаратів та 2461 (mordred) і 2883 (ClassyFire) інших лікарських засобів, який став основою для подальшого моделювання. Запропоновані методи дозволили ефективно структурувати та підготувати дані для застосування алгоритмів машинного навчання.
3. Здійснено конструювання молекулярних ознак на основі класифікації ClassyFire та бібліотеки mordred, що дозволило отримати набір із понад 1800 молекулярних дескрипторів. Проведено аналіз значущості дескрипторів і виявлено, що найбільш інформативними для прогнозування протидіабетичної активності є топологічні індекси, дескриптори ліпофільності та електронні параметри молекул.
4. Відібрано та впроваджено стратегії подолання дисбалансу класів при підготовці тренувальних даних. Експериментально встановлено, що комбінований підхід, який поєднує техніки SMOTE та Tomek links, дозволяє досягти найкращих результатів класифікації, підвищуючи значення метрики F1-score на 18% порівняно з навчанням на незбалансованих даних.

5. Побудовано та оцінено різні моделі машинного навчання для прогнозування протидіабетичної активності хімічних сполук. Найвищу ефективність показав ансамблевий підхід на основі Gradient Boosting, що досяг значень ROC-AUC 0.92 та F1-score 0.88 на незалежному тестовому наборі даних. Проведено порівняльний аналіз моделей та виявлено переваги кожного з алгоритмів.
6. Реалізовано алгоритм генерації нових молекул на основі SMILES-представлення з використанням рекурентних нейронних мереж (Char-RNN). Навчена модель успішно згенерувала 10,000 унікальних SMILES-рядків, з яких 73.4% відповідали хімічно валідним структурам, що підтверджує ефективність запропонованого підходу.
7. Проведено відбір згенерованих молекул за допомогою побудованих моделей прогнозування. Серед 7,340 валідних структур виявлено 325 сполук із високою прогнозованою протидіабетичною активністю (ймовірність активності > 0.8). Додатковий аналіз показав, що ці сполуки мають сприятливі фізико-хімічні властивості та потенційно можуть стати основою для розробки нових лікарських засобів.
8. Розроблено комплекс програмних засобів для автоматизації процесів обробки даних, моделювання та прогнозування біологічної активності хімічних сполук. Створені інструменти представлені у вигляді модульної архітектури та можуть бути інтегровані в існуючі системи розробки лікарських засобів для прискорення пошуку нових протидіабетичних препаратів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Конструювання ознак. *Вікіпедія*. 25.05.2022. URL: https://uk.wikipedia.org/w/index.php?title=%D0%9A%D0%BE%D0%BD%D1%81%D1%82%D1%80%D1%83%D1%8E%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%BE%D0%B7%D0%BD%D0%B0%D0%BA&oldid=35901160 (дата звернення: 07.02.2025).

2. Adeniji S. E., Uba S., Uzairu A. Multi-linear regression model, molecular binding interactions and ligand-based design of some prominent compounds against Mycobacterium tuberculosis. *Network Modeling Analysis in Health Informatics and Bioinformatics*. Вип. 9, № 1. С. 8. DOI:10.1007/s13721-019-0212-6.

3. Adinortey C. A., Kwarko G. B., Koranteng R. та ін. Molecular Structure-Based Screening of the Constituents of Calotropis procera Identifies Potential Inhibitors of Diabetes Mellitus Target Alpha Glucosidase. *Current Issues in Molecular Biology*. Вип. 44, № 2. С. 963–987. DOI:10.3390/cimb44020064.

4. Alonso-Rodríguez A. Logistic regression and world income distribution. *International Advances in Economic Research*. Вип. 7, № 2. С. 231–242. DOI:10.1007/BF02296011.

5. Bakirarar B., Elhan A. H. Class Weighting Technique to Deal with Imbalanced Class Problem in Machine Learning: Methodological Research. *Turkiye Klinikleri Journal of Biostatistics*. Вип. 15, № 1. С. 19–29. DOI:10.5336/biostatic.2022-93961.

6. Bazl R., Ganjali M. R., Derakhshankhah H. та ін. Prediction of tyrosinase inhibition for drug design using the genetic algorithm–multiple linear regressions. *Medicinal Chemistry Research*. Вип. 22, № 11. С. 5453–5465. DOI:10.1007/s00044-012-0440-0.

7. Bilodeau C., Jin W., Jaakkola T. та ін. Generative models for molecular discovery: Recent advances and challenges. *WIREs Computational Molecular Science*. Вип. 12, № 5. С. e1608. DOI:10.1002/wcms.1608.

8. Buber E., Diri B. Web Page Classification Using RNN. *Procedia Computer Science*. Вип. 154, 2019. С. 62–72. DOI:10.1016/j.procs.2019.06.011.

9. Capecchi A., Probst D., Reymond J.-L. One molecular fingerprint to rule them

all: drugs, biomolecules, and the metabolome. . *Journal of Cheminformatics*. Вип. 12, № 1. С. 43. DOI:10.1186/s13321-020-00445-4.

10. Choudhary S. Generating Molecules using a Char-RNN in Pytorch. *Medium*. 10.07.2019. URL: https://medium.com/@sunitachoudhary103/generating-molecules-using-a-char-rnn-in-pytorch-16885fd9394b (дата звернення: 19.03.2025).

11. За ред. Poongavanam, V., Ramaswamy, V. Computational drug discovery. Volume 2. Weinheim : Wiley-VCH, 2024. 315 с. ISBN 978-3-527-84073-1.

12. Datta S., Dev V. A., Eden M. R. Using correlation based adaptive LASSO algorithm to develop QSPR of antitumour agents for DNA–drug binding prediction. *Computers & Chemical Engineering*. Вип. 122, 03.2019. С. 258–264. DOI:10.1016/j.compchemeng.2018.08.039.

13. Debabov D. V., Debabova M. D. Drug development and open access: approaches and perspectives. *Microbiology Independent Research Journal (MIR Journal)*. Вип. 5, № 1. DOI:10.18527/2500-2236-2018-5-1-29-31.

14. Delgadillo Armendariz N. L., Rangel Vázquez N. A., Marquez Brazon E. SEMI-empirical PM6 method applied in the analysis of thermodynamics properties and molecular orbitals at different temperatures of adsorption drugs on chitosan hydrogels for type 2 diabetes. *Polymer Bulletin*. Вип. 76, № 7. С. 3423–3435. DOI:10.1007/s00289-018-2549-x.

15. Dhruv P., Naskar S. Image Classification Using Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN): A Review. *Machine Learning and Information Processing*. ред. Debabala Swain, Prasant Kumar Pattnaik, Pradeep K. Gupta. Singapore : Springer Singapore, 2020. С. 367–381. DOI:10.1007/978-981-15-1884-3_34.

16. Diéguez -San
interpretability using LDA and decision trees for α -amylase an -gluc
inhibitor classification studies. *Chemical Biology & Drug Design*. Вип. 94, № 1. С. 1414–1421. DOI:10.1111/cbdd.13518.

17. Djoumbou Feunang Y., Eisner R., Knox C. та ін. ClassyFire: automated chemical classification with a comprehensive, computable taxonomy. *Journal of*

Cheminformatics. Вип. 8, № 1. С. 61. DOI:10.1186/s13321-016-0174-y.

18. Dong Y., Ma X., Fu T. Electrical load forecasting: A deep learning approach based on K-nearest neighbors. *Applied Soft Computing*. Вип. 99, 02.2021. С. 106900. DOI:10.1016/j.asoc.2020.106900.

19. Dunetz J. R., Magano J., Weisenburger G. A. Large-Scale Applications of Amide Coupling Reagents for the Synthesis of Pharmaceuticals. *Organic Process Research & Development*. Вип. 20, № 2. С. 140–177. DOI:10.1021/op500305s.

20. Durant J. L., Leland B. A., Henry D. R. та ін. Reoptimization of MDL Keys for Use in Drug Discovery. *Journal of Chemical Information and Computer Sciences*. Вип. 42, № 6. С. 1273–1280. DOI:10.1021/ci010132r.

21. Durbin M., Wonders M. A., Flaska M. та ін. K-Nearest Neighbors regression for the discrimination of gamma rays and neutrons in organic scintillators. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*. Вип. 987, 01.2021. С. 164826. DOI:10.1016/j.nima.2020.164826.

22. Genheden S., Thakkar A., Chadimová V. та ін. AiZynthFinder: a fast, robust and flexible open-source software for retrosynthetic planning. *Journal of Cheminformatics*. Вип. 12, № 1. С. 70. DOI:10.1186/s13321-020-00472-1.

23. Gierlich C., Palkovits S. Featurizing chemistry for machine learning — methods and a coded example. *Current Opinion in Chemical Engineering*. Вип. 37, 09.2022. С. 100840. DOI:10.1016/j.coche.2022.100840.

24. Habib J. M. Understanding Model Performance Metrics: Precision, Recall, F1 Score, and More. *Medium*. 05.09.2024. URL: https://medium.com/@jaberi.mohamedhabib/understanding-model-performance-metrics-precision-recall-f1-score-and-more-ad9ccdce7ff5 (дата звернення: 03.01.2025).

25. Hu C., Li S., Yang C. та ін. ScaffoldGVAE: scaffold generation and hopping of drug molecules via a variational autoencoder based on multi-view graph neural networks. *Journal of Cheminformatics*. Вип. 15, № 1. С. 91. DOI:10.1186/s13321-023-00766-0.

26. Huang E. W., Bhope A., Lim J. та ін. Tissue-guided LASSO for prediction of clinical drug response using preclinical samples. *PLOS Computational Biology*. ред. Avner Schlessinger. Вип. 16, № 1. С. e1007607. DOI:10.1371/journal.pcbi.1007607.

27. Ito Y., Shiraishi E., Kato A. та ін. The Utility of Decision Trees in Oncofertility Care in Japan. *Journal of Adolescent and Young Adult Oncology*. Вип. 6, № 1. С. 186–189. DOI:10.1089/jayao.2016.0045.

28. Janairo J. I. B. Support vector machine in drug design. *Cheminformatics, QSAR and Machine Learning Applications for Novel Drug Development*. Elsevier, 2023. С. 161–179. DOI:10.1016/B978-0-443-18638-7.00021-9.

29. Jimenes-Vargas K., Perez-Castillo Y., Tejera E. та ін. Exploring Target Identification for Drug Design with K-Nearest Neighbors' Algorithm. *Artificial Intelligence and Soft Computing*. ред. Leszek Rutkowski, Rafał Scherer, Marcin Korytkowski, Witold Pedrycz, Ryszard Tadeusiewicz, Jacek M. Zurada. Cham : Springer Nature Switzerland, 2023. С. 219–227. DOI:10.1007/978-3-031-42508-0_20.

30. Knox C., Wilson M., Klinger C. M. та ін. DrugBank 6.0: the DrugBank Knowledgebase for 2024. *Nucleic Acids Research*. Вип. 52, № D1. С. D1265–D1275. DOI:10.1093/nar/gkad976.

31. Kosolwattana T., Liu C., Hu R. та ін. A self-inspected adaptive SMOTE algorithm (SASMOTE) for highly imbalanced data classification in healthcare. *BioData Mining*. Вип. 16, № 1. С. 15. DOI:10.1186/s13040-023-00330-4.

32. Kotecha R., Garg S. Genetic programming-based evolution of classification trees for decision support in banking sector. *International Journal of Knowledge Engineering and Soft Data Paradigms*. Вип. 5, № 3/4. С. 186. DOI:10.1504/IJKESDP.2016.084599.

33. Laiolo J., Lanza P. A., Parravicini O. та ін. Structure activity relationships and the binding mode of quinolinone-pyrimidine hybrids as reversal agents of multidrug resistance mediated by P-gp. *Scientific Reports*. Вип. 11, № 1. С. 16856. DOI:10.1038/s41598-021-96226-6.

34. Lee D. J., Veneri D. A. Development and acceptability testing of decision trees

for self-management of prosthetic socket fit in adults with lower limb amputation. *Disability and Rehabilitation*. Вип. 40, № 9. С. 1066–1071. DOI:10.1080/09638288.2017.1286694.

35. Leguy J., Cauchy T., Duval B. та ін. Goal-directed generation of new molecules by AI methods. *Computational and Data-Driven Chemistry Using Artificial Intelligence*. Elsevier, 2022. С. 39–67. DOI:10.1016/B978-0-12-822249-2.00004-9.

36. Li B., Wang Y., Yin Z. та ін. Decision tree -bas
molecular fragments for protein *Organic Biology & Drug Design*.
Вип. 103, № 1. С. e14427. DOI:10.1111/cbdd.14427.

37. Li J. J., Li J. J. Heck Reaction. *Name Reactions*. Cham : Springer International Publishing, 2021. С. 244–247. DOI:10.1007/978-3-030-50865-4_64.

38. Li M.-M., Zhang T., Cheng L. та ін. Ketone α-alkylation at the more-hindered site. *Nature Communications*. Вип. 14, № 1. С. 3326. DOI:10.1038/s41467-023-38741-w.

39. Liu C., Wei D., Xiang J. та ін. An Improved Anticancer Drug-Response Prediction Based on an Ensemble Method Integrating Matrix Completion and Ridge Regression. *Molecular Therapy - Nucleic Acids*. Вип. 21, 09.2020. С. 676–686. DOI:10.1016/j.omtn.2020.07.003.

40. Liu Y., Li Y., Xie D. Implications of imbalanced datasets for empirical ROC-AUC estimation in binary classification tasks. *Journal of Statistical Computation and Simulation*. Вип. 94, № 1. С. 183–203. DOI:10.1080/00949655.2023.2238235.

41. Loeffler H. H., He J., Tibo A. та ін. Reinvent 4: Modern AI–driven generative molecule design. *Journal of Cheminformatics*. Вип. 16, № 1. С. 20. DOI:10.1186/s13321-024-00812-5.

42. Lynam A. L., Dennis J. M., Owen K. R. та ін. Logistic regression has similar performance to optimised machine learning algorithms in a clinical setting: application to the discrimination between type 1 and type 2 diabetes in young adults. *Diagnostic and Prognostic Research*. Вип. 4, № 1. С. 6. DOI:10.1186/s41512-020-00075-2.

43. Maltarollo V. G., Kronenberger T., Espinoza G. Z. та ін. Advances with support vector machines for novel drug discovery. *Expert Opinion on Drug Discovery*. Вип. 14, № 1. С. 23–33. DOI:10.1080/17460441.2019.1549033.

44. Matharaarachchi S., Domaratzki M., Muthukumarana S. Enhancing SMOTE for imbalanced data with abnormal minority instances. *Machine Learning with Applications*. Вип. 18, 12.2024. С. 100597. DOI:10.1016/j.mlwa.2024.100597.

45. Mazuz E., Shtar G., Shapira B. та ін. Molecule generation using transformers and policy gradient reinforcement learning. *Scientific Reports*. Вип. 13, № 1. С. 8799. DOI:10.1038/s41598-023-35648-w.

46. Méndez-Lucio O., Baillif B., Clevert D.-A. та ін. De novo generation of hit-like molecules from gene expression signatures using artificial intelligence. *Nature Communications*. Вип. 11, № 1. С. 10. DOI:10.1038/s41467-019-13807-w.

47. Morgan H. L. The Generation of a Unique Machine Description for Chemical Structures-A Technique Developed at Chemical Abstracts Service. *Journal of Chemical Documentation*. Вип. 5, № 2. С. 107–113. DOI:10.1021/c160017a018.

48. Moriwaki H., Tian Y.-S., Kawashita N. та ін. Mordred: a molecular descriptor calculator. *Journal of Cheminformatics*. Вип. 10, № 1. С. 4. DOI:10.1186/s13321-018-0258-y.

49. Movahedi F., Padman R., Antaki J. F. Limitations of receiver operating characteristic curve on imbalanced data: Assist device mortality risk scores. *The Journal of Thoracic and Cardiovascular Surgery*. Вип. 165, № 4. С. 1433-1442.e2. DOI:10.1016/j.jtcvs.2021.07.041.

50. Netrapalli P. Stochastic Gradient Descent and Its Variants in Machine Learning. *Journal of the Indian Institute of Science*. Вип. 99, № 2. С. 201–213. DOI:10.1007/s41745-019-0098-4.

51. Nian R. Introduction. An Introduction to ADASYN (with code!). *Medium*. Introduction. 13.12.2019. URL: https://medium.com/@ruinian/an-introduction-to-adasyn-with-code-1383a5ece7aa (дата звернення: 19.03.2025).

52. Obaido G., Mienye I. D., Egbelowo O. F. та ін. Supervised machine learning in drug discovery and development: Algorithms, applications, challenges, and

prospects. *Machine Learning with Applications*. Вип. 17, 09.2024. С. 100576. DOI:10.1016/j.mlwa.2024.100576.

53. Odugbemi A. I., Nyirenda C., Christoffels A. та ін. Artificial intelligence in antidiabetic drug discovery: The advances in QSAR and the prediction of α-glucosidase inhibitors. *Computational and Structural Biotechnology Journal*. Вип. 23, 12.2024. С. 2964–2977. DOI:10.1016/j.csbj.2024.07.003.

54. Olaokun O. O., Zubair M. S. Antidiabetic Activity, Molecular Docking, and ADMET Properties of Compounds Isolated from Bioactive Ethyl Acetate Fraction of Ficus lutea Leaf Extract. *Molecules*. Вип. 28, № 23. С. 7717. DOI:10.3390/molecules28237717.

55. Onikanni S. A., Lawal B., Munyembaraga V. та ін. Profiling the Antidiabetic Potential of Compounds Identified from Fractionated Extracts of Entada africana toward Glucokinase Stimulation: Computational Insight. *Molecules*. Вип. 28, № 15. С. 5752. DOI:10.3390/molecules28155752.

56. Otsuka Y., Hirama K., Yokota K. та ін. Methods for predicting offender's criminal history in single homicide cases: Comparison of logistic regression analysis and decision trees. *Japanese Journal of Forensic Science and Technology*. Вип. 22, № 1. С. 25–34. DOI:10.3408/jafst.716.

57. Panesar S. S., D'Souza R. N., Yeh F.-C. та ін. Machine Learning Versus Logistic Regression Methods for 2-Year Mortality Prognostication in a Small, Heterogeneous Glioma Database. *World Neurosurgery: X*. Вип. 2, 04.2019. С. 100012. DOI:10.1016/j.wnsx.2019.100012.

58. Park H., Kim H. AR-ADASYN: angle radius-adaptive synthetic data generation approach for imbalanced learning. *Statistics and Computing*. Вип. 34, № 5. С. 166. DOI:10.1007/s11222-024-10479-5.

59. Patel L., Shukla T., Huang X. та ін. Machine Learning Methods in Drug Discovery. *Molecules*. Вип. 25, № 22. С. 5277. DOI:10.3390/molecules25225277.

60. Pitcher B., Alaqla A., Noujeim M. та ін. Binary Decision Trees for Preoperative Periapical Cyst Screening Using Cone-beam Computed Tomography. *Journal of Endodontics*. Вип. 43, № 3. С. 383–388. DOI:10.1016/j.joen.2016.10.046.

61. Pu Q., Li Y., Zhang H. та ін. Screen efficiency comparisons of decision tree and neural network algorithms in machine learning assisted drug design. *Science China Chemistry*. Вип. 62, № 4. С. 506–514. DOI:10.1007/s11426-018-9412-6.

62. Pua Y.-H., Kang H., Thumboo J. та ін. Machine learning methods are comparable to logistic regression techniques in predicting severe walking limitation following total knee arthroplasty. *Knee Surgery, Sports Traumatology, Arthroscopy*. Вип. 28, № 10. С. 3207–3216. DOI:10.1007/s00167-019-05822-7.

63. RDKit Fingerprint. *NodePit*. URL: https://nodepit.com/node/org.rdkit.knime.nodes.rdkfingerprint.RDKitFingerprintNodeFactory (дата звернення: 04.02.2025).

64. Rezvani S., Wang X. A broad review on class imbalance learning techniques. *Applied Soft Computing*. Вип. 143, 08.2023. С. 110415. DOI:10.1016/j.asoc.2023.110415.

65. Richardson E., Trevizani R., Greenbaum J. A. та ін. The receiver operating characteristic curve accurately assesses imbalanced datasets. *Patterns*. Вип. 5, № 6. С. 100994. DOI:10.1016/j.patter.2024.100994.

66. Rodríguez-Pérez R., Bajorath J. Evolution of Support Vector Machine and Regression Modeling in Chemoinformatics and Drug Discovery. *Journal of Computer-Aided Molecular Design*. Вип. 36, № 5. С. 355–362. DOI:10.1007/s10822-022-00442-9.

67. Rogers D., Hahn M. Extended-Connectivity Fingerprints. *Journal of Chemical Information and Modeling*. Вип. 50, № 5. С. 742–754. DOI:10.1021/ci100050t.

68. Rueda-Zubiaurre A., Yahiya S., Fischer O. J. та ін. Structure–Activity Relationship Studies of a Novel Class of Transmission Blocking Antimalarials Targeting Male Gametes. *Journal of Medicinal Chemistry*. Вип. 63, № 5. С. 2240–2262. DOI:10.1021/acs.jmedchem.9b00898.

69. Rydzewski N. R., Peterson E., Lang J. M. та ін. Predicting cancer drug TARGETS - TreAtment Response Generalized Elastic-neT Signatures. *npj Genomic Medicine*. Вип. 6, № 1. С. 76. DOI:10.1038/s41525-021-00239-z.

70. Rymarczyk T., Kozłowski E., Kłosowski G. та ін. Logistic Regression for

Machine Learning in Process Tomography. *Sensors*. Вип. 19, № 15. С. 3400. DOI:10.3390/s19153400.

71. Saggio G., Cavallo P., Ricci M. та ін. Sign Language Recognition Using Wearable Electronics: Implementing k-Nearest Neighbors with Dynamic Time Warping and Convolutional Neural Network Algorithms. *Sensors*. Вип. 20, № 14. С. 3879. DOI:10.3390/s20143879.

72. Schetz J. A. Structure-Activity Relationships: Theory, Uses and Limitations☆. *Reference Module in Biomedical Sciences*. Elsevier, 2015. С. B9780128012383053290. DOI:10.1016/B978-0-12-801238-3.05329-0.

73. Sikander R., Ghulam A., Ali F. XGB-DrugPred: computational prediction of druggable proteins using eXtreme gradient boosting and optimized features set. *Scientific Reports*. Вип. 12, № 1. С. 5505. DOI:10.1038/s41598-022-09484-3.

74. Singh M., Wasim Bhatt M., Bedi H. S. та ін. Performance of bernoulli's naive bayes classifier in the detection of fake news. *Materials Today: Proceedings*. 12.2020. С. S2214785320385333. DOI:10.1016/j.matpr.2020.10.896.

75. Sippl W., Ntie-Kang F. Editorial to Special Issue—"Structure-Activity Relationships (SAR) of Natural Products". *Molecules*. Вип. 26, № 2. С. 250. DOI:10.3390/molecules26020250.

76. Smolen H. A statistical comparison of logistic regression versus machine learning tree-based classification methods in predicting coronary heart disease using real-world patient-level data. *Value in Health*. Вип. 22, 11.2019. С. S563–S564. DOI:10.1016/j.jval.2019.09.844.

77. Stylianou N., Akbarov A., Kontopantelis E. та ін. Mortality risk prediction in burn injury: Comparison of logistic regression with machine learning approaches. *Burns*. Вип. 41, № 5. С. 925–934. DOI:10.1016/j.burns.2015.03.016.

78. Sun Y., Wong A. K. C., Kamel M. S. CLASSIFICATION OF IMBALANCED DATA: A REVIEW. *International Journal of Pattern Recognition and Artificial Intelligence*. Вип. 23, № 04. С. 687–719. DOI:10.1142/S0218001409007326.

79. T. Billones L., C. Gonzaga A. Multiple Logistic Regression Modeling of Compound Class as Active or Inactive Against COX-2 and Prediction on Designed

Coxib Derivatives and Similar Compounds. *Chem-Bio Informatics Journal*. Вип. 22, № 0. С. 63–87. DOI:10.1273/cbij.22.63.

80. Tian Y., Zhang Y., Zhang H. Recent Advances in Stochastic Gradient Descent in Deep Learning. *Mathematics*. Вип. 11, № 3. С. 682. DOI:10.3390/math11030682.

81. Toussi C. A., Haddadnia J., Matta C. F. Drug design by machine-trained elastic networks: predicting Ser/Thr-protein kinase inhibitors' activities. *Molecular Diversity*. Вип. 25, № 2. С. 899–909. DOI:10.1007/s11030-020-10074-6.

82. Twilton J., Christensen M., DiRocco D. A. та ін. Selective Hydrogen Atom Abstraction through Induced Bond Polarization: Direct α -Ary through Photoredox, HAT, and Nickel Catalysis. *Angewandte Chemie International Edition*. Вип. 57, № 19. С. 5369–5373. DOI:10.1002/anie.201800749.

83. Venkatasubramaniam A., Wolfson J., Mitchell N. та ін. Decision trees in epidemiological research. *Emerging Themes in Epidemiology*. Вип. 14, № 1. С. 11. DOI:10.1186/s12982-017-0064-4.

84. Wang J., Li X., Li J. та ін. NGCU: A New RNN Model for Time-Series Data Prediction. *Big Data Research*. Вип. 27, 02.2022. С. 100296. DOI:10.1016/j.bdr.2021.100296.

85. Wharton S. B., Parikh C., Matthews F. E. та ін. Epidemiological pathology of Aβ deposition in the ageing brain in CFAS: addition of multiple Aβ-derived measures does not improve dementia assessment using logistic regression and machine learning approaches. *Acta Neuropathologica Communications*. Вип. 7, № 1. С. 198. DOI:10.1186/s40478-019-0858-4.

86. Willighagen E. L., Mayfield J. W., Alvarsson J. та ін. The Chemistry Development Kit (CDK) v2.0: atom typing, depiction, molecular formulas, and substructure searching. *Journal of Cheminformatics*. Вип. 9, № 1. С. 33. DOI:10.1186/s13321-017-0220-4.

87. Xie Y., Jiang B., Gong E. та ін. Use of Gradient Boosting Machine Learning to Predict Patient Outcome in Acute Ischemic Stroke on the Basis of Imaging, Demographic, and Clinical Information. *American Journal of Roentgenology*. Вип. 212, № 1. С. 44–51. DOI:10.2214/AJR.18.20260.

88. Yap C. W. PaDEL -des
descriptors and fingerprints. *Journal of Computational Chemistry*. Вип. 32, № 7. С. 1466–1474. DOI:10.1002/jcc.21707.

89. Zhang H., Cheng N., Zhang Y. та ін. Label flipping attacks against Naive Bayes on spam filtering systems. *Applied Intelligence*. Вип. 51, № 7. С. 4503–4514. DOI:10.1007/s10489-020-02086-4.

90. Zhang H., Mao J., Qi H.-Z. та ін. Developing novel computational prediction models for assessing chemical-induced neurotoxicity using naïve Bayes classifier technique. *Food and Chemical Toxicology*. Вип. 143, 09.2020. С. 111513. DOI:10.1016/j.fct.2020.111513.

91. Zhang Y., Saxe A. M., Advani M. S. та ін. Energy–entropy competition and the effectiveness of stochastic gradient descent in machine learning. *Molecular Physics*. Вип. 116, № 21–22. С. 3214–3223. DOI:10.1080/00268976.2018.1483535.

92. Управління великими обсягами даних: роль та ефективність систем ETL для бізнесу. *dev.ua*. URL: https://dev.ua/blogs/posts/yalantis-blog-1704811522 (accessed 05/01/2025).

93. Damarta R., Hidayat A., Abdullah A. S. The application of k-nearest neighbors classifier for sentiment analysis of PT PLN (Persero) twitter account service quality. *Journal of Physics: Conference Series*. Vol. 1722, Issue 1. P. 012002. DOI:10.1088/1742-6596/1722/1/012002.

94. Dineshkumar R., Sunil Kumar V., Reddy R. A. et al. Multi-objective de novo Drug Design using Bidirectional Recurrent Neural Networks and Nondominated Sorting. *2024 Second International Conference on Data Science and Information System (ICDSIS)2024 Second International Conference on Data Science and Information System (ICDSIS)*. (Hassan, India, 17.05.2024). Hassan, India : IEEE, 2024. DOI:10.1109/ICDSIS61070.2024.10594595. P. 1–4.

95. Fernandez A., Garcia S., Herrera F. et al. SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary. *Journal of Artificial Intelligence Research*. Vol. 61, 20.04.2018. P. 863–905. DOI:10.1613/jair.1.11192.

96. Ganai A. F., Khursheed F. Predicting next Word using RNN and LSTM cells: Stastical Language Modeling. *2019 Fifth International Conference on Image Information Processing (ICIIP)2019 Fifth International Conference on Image Information Processing (ICIIP)*. (Shimla, India, 11.2019). Shimla, India : IEEE, 2019. DOI:10.1109/ICIIP47207.2019.8985885. P. 469–474.

97. Géron A. Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems. Third edition. Beijing : O'Reilly Media, Inc, 2023. 834 p. [QA76.73.P98 G45 2023]. ISBN 978-1-0981-2597-4.

98. Guha R. On Exploring Structure–Activity Relationships. *In Silico Models for Drug Discovery*. ed. S. Kortagere. Totowa, NJ : Humana Press, 2013. P. 81–94. DOI:10.1007/978-1-62703-342-8_6.

99. Haibo He, Yang Bai, Garcia E. A. et al. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)2008 IEEE International Joint Conference on Neural Networks (IJCNN 2008 - Hong Kong)*. (Hong Kong, China, 06.2008). Hong Kong, China : IEEE, 2008. DOI:10.1109/IJCNN.2008.4633969. P. 1322–1328.

100. Hotzy F., Theodoridou A., Hoff P. et al. Machine Learning: An Approach in Identifying Risk Factors for Coercion Compared to Binary Logistic Regression. *Frontiers in Psychiatry*. Vol. 9, 12.06.2018. P. 258. DOI:10.3389/fpsyt.2018.00258.

101. Kanter J. M., Veeramachaneni K. Deep feature synthesis: Towards automating data science endeavors. *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. (Campus des Cordeliers, Paris, France, 10.2015). Campus des Cordeliers, Paris, France : IEEE, 2015. DOI:10.1109/DSAA.2015.7344858. P. 1–10.

102. Li D., Sun J., Yang H. et al. An Enhanced Naive Bayes Model for Dissolved Oxygen Forecasting in Shellfish Aquaculture. *IEEE Access*. Vol. 8, 2020. P. 217917–217927. DOI:10.1109/ACCESS.2020.3042180.

103. Machine Learning in Python documentation. *scikit-learn*. 2025. URL: https://scikit-learn/stable/api/sklearn.linear_model.html (accessed 03/01/2025).

104. Mandal L., Jana N. D. A Comparative Study of Naive Bayes and k-NN Algorithm for Multi-class Drug Molecule Classification. *2019 IEEE 16th India Council International Conference (INDICON)2019 IEEE 16th India Council International Conference (INDICON)*. (Rajkot, India, 12.2019). Rajkot, India : IEEE, 2019. DOI:10.1109/INDICON47234.2019.9029095. P. 1–4.

105. Manu Garg, Muskan Gupta, Parveen et al. Email Spam Detection Using Logistic Regression. Vol. 13, Issue 10. P. 2111–2118.

106. Nargis T., Nisar Shaha S. A., Hazareena A. An Approach in Prediction of Medicine Side Effects Using RNN Networks. *2024 IEEE International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER)2024 IEEE International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER)*. (Mangalore, India, 18.10.2024). Mangalore, India : IEEE, 2024. DOI:10.1109/DISCOVER62353.2024.10750658. P. 416–421.

107. Patil A., Framewala A., Kazi F. Explainability of SMOTE Based Oversampling for Imbalanced Dataset Problems. *2020 3rd International Conference on Information and Computer Technologies (ICICT)2020 3rd International Conference on Information and Computer Technologies (ICICT)*. (San Jose, CA, USA, 03.2020). San Jose, CA, USA : IEEE, 2020. DOI:10.1109/ICICT50521.2020.00015. P. 41–45.

108. Pawar K., Jalem R. S., Tiwari V. Stock Market Price Prediction Using LSTM RNN. *Emerging Trends in Expert Applications and Security*. ed. V. S. Rathore., M. Worring., D. K. Mishra. et al. Singapore : Springer Singapore, 2019. P. 493–503. DOI:10.1007/978-981-13-2285-3_58.

109. Pradipta G. A., Wardoyo R., Musdholifah A. et al. SMOTE for Handling Imbalanced Data Problem : A Review. *2021 Sixth International Conference on Informatics and Computing (ICIC)2021 Sixth International Conference on Informatics and Computing (ICIC)*. (Jakarta, Indonesia, 03.11.2021). Jakarta, Indonesia : IEEE, 2021. DOI:10.1109/ICIC54025.2021.9632912. P. 1–8.

110. Puspitasari D., Ramanda K., Supriyatna A. et al. Comparison of Data Mining Algorithms Using Artificial Neural Networks (ANN) and Naive Bayes for Preterm Birth Prediction. *Journal of Physics: Conference Series*. Vol. 1641, Issue 1. P. 012068. DOI:10.1088/1742-6596/1641/1/012068.

111. Putri T. E., Subagio R. T., Kusnadi et al. Classification System Of Toddler Nutrition Status using Naïve Bayes Classifier Based on Z- Score Value and Anthropometry Index. *Journal of Physics: Conference Series*. Vol. 1641, Issue 1. P. 012005. DOI:10.1088/1742-6596/1641/1/012005.

112. Requests: HTTP for Humans — Requests 2.0.0 documentation. URL: https://docs.python-requests.org/en/v2.0.0/ (accessed 03/01/2025).

113. Richardson L. Beautiful Soup Documentation. 2025. URL: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

114. Sanket B. Bhatshankar A. S. T. Quantitative structure-activity relationship and group-based quantitative structure-activity relationship: a review. *International Journal of Pharmaceutical Sciences and Research*. Vol. 14, Issue 3.

115. Silahudin D., Henderi, Holidin A. Model Expert System for Diagnosis of Covid-19 Using Naïve Bayes Classifier. *IOP Conference Series: Materials Science and Engineering*. Vol. 1007, Issue 1. P. 012067. DOI:10.1088/1757-899X/1007/1/012067.

116. The Python Language Reference. 2025. URL: https://docs.python.org/3.8/reference/ (accessed 03/01/2025).

117. Universitas Syiah Kuala, Noviandy T. R., Idroes G. M. et al. Machine Learning Approach to Predict AXL Kinase Inhibitor Activity for Cancer Drug Discovery Using Bayesian Optimization-XGBoost. *Journal of Soft Computing and Data Mining*. Vol. 5, Issue 1. DOI:10.30880/jscdm.2024.05.01.004.

118. Haghighi M. H. Z. Analyzing Astronomical Data with Machine Learning Techniques. arXiv, 2023. DOI:10.48550/ARXIV.2302.11573.

119. Lopez Perez K., Avellaneda Tamayo J., Chen L. et al. Molecular Similarity: Theory, Applications, and Perspectives. Chemistry, 2023. DOI:10.26434/chemrxiv-2023-cs3wb.

ДОДАТКИ

Додаток А

Парсер бази даних DrugBank

Import libraries:

```
from bs4 import BeautifulSoup
from pprint import pprint
import requests
from IPython.display import display, HTML
import sqlite3
import pandas as pd
import re
import sys
```

Make a request with requests module via a URL(Uniform Resource Locator)

```
def make_req(url:str, timeout=50)-> object:
    """Return new Request object using the library Request() from URL
    Argument:
    url : is a unique identifier used to locate a resource on the Internet
    """
    headers = {
        'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:102.0)
Gecko/20100101 Firefox/102.0',}
    resp = requests.get(url, headers=headers, timeout=timeout)
    return resp
```

Data Extraction and Cleaning

```
class ScrapingDragbank():
    """the class object takes the bs4 object as input
    and returns a dictionary with data from go.drugbank.com"""

    def __init__(self, soup: object, _id:str)->None:
        self.soup = soup
        self.id = _id
        self.base_sel = self.get_base_sel()
        self.drug = self.ForeignKey()
        print(self.drug)

    def get_base_sel(self)->str:
        """Return a key tag that contains the required data"""
        return 'div.drug-card div.card-content dl'

    def get_simple_data(self, _id:str)->dict:
        """Return dictionary wich contain data with some id"""
        for i in self.soup.select(f'{self.base_sel}'):
            for dt in i.find_all_next('dt'):
                if dt.get('id') == _id:
                    return {dt.text : dt.find_next('dd').text}

    def get_href_data(self, _id:str)->dict:
        """Get href from MSDS tag"""
        for i in self.soup.select(f'{self.base_sel}'):
            for dt in i.find_all_next('dt'):
                if dt.get('id') == _id:
                    return {dt.text : dt.find_next('a').get('href')}
```

Додаток А (продовження)

```
    def ForeignKey(self)->str:
        """Return accession number into database drugbank """
        if self.get_simple_data('drugbank-accession-number') == None:
            return self.id
        else:
            return self.get_simple_data('drugbank-accession-number').get('DrugBank
Accession Number')

    def get_experimental_properties(self)->dict:
        """Return dictionary with some physicochemical properties of drug"""
        properties = {}
        for i in self.soup.select(f'{self.base_sel}'):
            for dt in i.find_all_next('dt'):
                if dt.get('id') == 'experimental-properties':
                    table = dt.find_next('table')
                    if table != None:
                        headers = [header.text for header in table.find_all('th')]
                        res = [{headers[i]: cell.text for i, cell in
enumerate(row.find_all('td'))} for row in table.find_all('tr')]
                        for i in res:
                            properties[i.get('Property')] = i.get('Value')
        if not properties:
            return {'id':self.drug}
        else:
            properties.pop(None)
            properties.update({'id':self.drug})
            return properties

    def det_drug_categories(self)-> dict:
        """Return dictionary with data categories of drug"""
        for i in self.soup.select(f'{self.base_sel}'):
            for dt in i.find_all_next('dt'):
                if dt.get('id') == 'drug-categories':
                    ulli = set()
                    for ul in dt.find_next('ul'):
                        ulli.add(ul.find_next('li').text)
                    return {self.drug :{dt.text : ulli}}

    def get_atc(self)->dict:
        """return dictionary with data Anatomical Therapeutic Chemical Classification
System from database drugbank"""
        atc = {}
        for i in self.soup.select(f'{self.base_sel}'):
            for dt in i.find_all_next('dt'):
                if dt.get('id') == 'atc-codes':
                    for li in dt.find_next('ul', {'class':'atc-drug-tree'}):
                        atc.update({li.text.split('--')[0].replace(' ', ''):1})
        atc.update({'id':self.drug})
        return atc

    @staticmethod
    def remove_character_str(s:str)->str:
        """Return the corrected taxonomy value"""
        if s[-1] == 's':
            return s[:len(s)-1]
        else:
            return s
```

Додаток А (продовження)

```
    def get_chemical_taxonomy(self)->object:
        """
        Return generator with dictionary with classifying compounds by their functional
groups:
        group key in dict - is Names functional groups
        link  - has link to is a web-based application for automated structural
classification of chemical entities.
        drug  - is accession number into database drugbank
        """
        for i in self.soup.select(f'{self.base_sel}'):
            for dt in i.find_all_next('dt'):
                if dt.get('id') == 'chemical-taxonomy':
                    for span in dt.find_all_next('span', {'class':'separated-list-
item'}):
                        href = span.find_next('a', {'class':'classyfire-taxnode'})
                        yield {'id':self.remove_character_str(span.text), self.drug:1}
                        if href != None:
                            group = self.remove_character_str(href.text)
                            link = href.get('href')
                            description = href.get('data-content')
                            yield {'id':group, self.drug:link[-8:-1]}


    def get_simple_table(self)->dict:
        """Return dictionary with fields:
'generic-name' - is a medication with the exact same active ingredient as the brand-
name drug, is taken the same way and offers the same effect,
'type'         - about a molecular size some drugs,
'cas-number'   - is a unique identification number assigned by the Chemical Abstracts
Service,
'iupac-name'   - the most 'official' rules for names of chemical compounds are
promulgated by the International Union of Pure and Applied Chemistry (IUPAC),
'smiles'       - is a line notation for describing chemical structures using short
ASCII strings,
'msds'         - link to a Material Safety Data Sheet (MSDS) is a document that
contains information on the potential hazards"""

        general_data = {}
        for _id in ['generic-name', 'type', 'cas-number', 'iupac-name',
                    'smiles', 'msds']:
            if self.get_simple_data(_id) == None:
                continue
            else:
                general_data.update(self.get_simple_data(_id))
            if self.get_href_data('msds') == None:
                continue
            else:
                general_data.update(self.get_href_data('msds'))
        general_data.update({'id':self.drug})
        return general_data
```

Додаток А (продовження)

```
    def get_weight(self)->dict:
        """Return dictionary with weight values of drugs"""
        weight = {}
        weight.update({'id':self.drug})
        if self.get_simple_data('weight') == None:
            return weight
        else:
            s = self.get_simple_data('weight')['Weight']
            for j, i in enumerate(s.split(' ')):
                if i.find(':') == -1:
                    continue
                else:
                    weight.update({'Weight_' + s.split(' ')[j].replace(':', '') :
float(s.split(' ')[j+1])})
            return weight
```

Design a database and recording collected data

```
class FillDataBase():
    """Class wich recording collected data"""
    def __init__(self, Dict:dict, table:str, DB:object)->None:
        """Take arguments for writing data into database
        Arguments:
                    Dict : dictionary with some data, mast have key with id
                    table : name of table in database
                    DB : object database sqlite3
        """
        self.Dict = self.new_dict(Dict)
        self.table_name_DB = table
        self.DB = DB
        self.columns = self.get_columns()
        self.placeholder = self.get_placeholders()
        self.con = sqlite3.connect(self.DB)

    @staticmethod
    def clean_name(s:str)-> str:
        """Return new name of field wich filtered from unwanted characters"""
        reg = re.compile('[^a-zA-Z0-9--_]')
        s = s.replace('-','--').replace(' ', '_')
        return reg.sub('', s)

    def new_dict(self, Dict:dict)->dict:
        """Return dictionary with filered keys"""
        return {self.clean_name(k):v for k,v in Dict.items()}


    def get_columns(self)->str:
        """Return columns string for SQL query"""
        return str(list(self.Dict.keys())).replace('[',
'').replace(']','').replace("'", '')

    def get_placeholders(self)->str:
        """Return columns string for SQL query"""
        return ':'+', :'.join(self.Dict.keys())
    def CREATE_TABLE(self)->None:
        """Create table into database"""
        cursor = self.con.cursor()
        query = f"CREATE TABLE IF NOT EXISTS {self.table_name_DB} (id VARCHAR(17)
PRIMARY KEY);"
        cursor.execute(query)
```

Додаток А (продовження)

```
    def ALTER_TABLE(self, column:str)-> None:
        """Add some columns into table of database"""
        cursor = self.con.cursor()
        cursor.execute('ALTER TABLE "%s" ADD "%s" ;' % (self.table_name_DB, column))

    def UPDATE(self, value_id:[str,int,float], column_value:str, column:str)->None:
        """"
        Update values into some table of database
        Arguments:
                    value_id: new value wich mast update
                    column_value: name of column of table
                    column: name column wich mast have replase data
        See: https://pynative.com/python-sqlite-update-table/"""
        try:
            sqliteConnection = sqlite3.connect(self.DB)
            cursor = sqliteConnection.cursor()
            # print("Connected to SQLite")

            sql_update_query = f"""UPDATE {self.table_name_DB} SET {column} = ? where
id = ?"""
            #sql_update_query = f"""UPDATEUpdate {taxonomy} set {column} = ? where id =
?"""
            data = (column_value, value_id)
            cursor.execute(sql_update_query, data)
            sqliteConnection.commit()
            # print("Record Updated successfully")
            cursor.close()

        except sqlite3.Error as error:
            print("Failed to update sqlite table", error)
        finally:
            if sqliteConnection:
                sqliteConnection.close()
                # print("The sqlite connection is closed")


    def INSERT_INTO(self)-> None:
        """Insert data and columns into table of database"""
        cursor = self.con.cursor()
        query = 'INSERT OR IGNORE INTO %s (%s) VALUES (%s);' % (self.table_name_DB,
self.columns, self.placeholder)
        # query = 'INSERT INTO %s (%s) VALUES (%s);' % (self.table_name_DB,
self.columns, self.placeholder)
        # https://stackoverflow.com/questions/36518628/sqlite3-integrityerror-unique-
constraint-failed-when-inserting-a-value
        try:
            cursor.execute(query, self.Dict)
            self.con.commit()
        except sqlite3.Error as e:
            if e.args[0][0:26+len(self.table_name_DB)] == f'table {self.table_name_DB}
has no column named':
                for column in self.Dict.keys():
                    try:
                        self.ALTER_TABLE(column)
                    except sqlite3.Error as E:
                        if E.args[0][0:22] == f'duplicate column name:':
                            continue
                        else:
```

```
                                print('Some ERROR FROM  def INSERT_INTO')
                                print(e.args)
                                print(E.args)
                                sys.exit()
                    elif e.args[0][:24] == 'UNIQUE constraint failed':
                        pass
                    else:
                        print(e.args, 'FROM def INSERT_INTO')
                    self.INSERT_INTO()
```

Scraping Script

```
DB = 'FromDrugBankData.db'

for _key in range(1,10):
    DB_key = str(_key).zfill(5)
    url = f'https://go.drugbank.com/drugs/DB{DB_key}'
    resp = make_req(url)
    soup = BeautifulSoup(resp.text, 'html.parser')
    obj = ScrapingDragbank(soup, str('DB' + DB_key))
    table_DATA = {
    'atc': obj.get_atc(),
    'describe': obj.get_simple_table(),
    'weight': obj.get_weight(),
    'properties': obj.get_experimental_properties(),
    # 'taxonomy' : obj.get_chemical_taxonomy()
        }
    for table in table_DATA.keys():
        objF = FillDataBase(table_DATA[table], table, DB)
        objF.CREATE_TABLE()
        objF.INSERT_INTO()

    FK = obj.ForeignKey()
    taxonomy = obj.get_chemical_taxonomy()
    for d in taxonomy:
        objF = FillDataBase(d, 'taxonomy', DB)
        objF.CREATE_TABLE()
        objF.INSERT_INTO()
        for KEY in d:
            value_id = d['id']
            if KEY != 'id':
                objF.UPDATE(value_id, d[FK], FK)
```

Viewing Information About Database

```
def Query(SQL_query, DB, return_=False):
    try:
        with sqlite3.connect(DB) as con: # Create DB or connection
            df = pd.read_sql_query(f"{SQL_query}", con)
            if return_ == True:
                return df
            else:
                display(HTML(df.to_html()))
    except sqlite3.Error as e:
        print(e)
```

Додаток Б

Препроцесінг та формування необхідного набору даних

```
import sqlite3
import pandas as pd
from IPython.display import display, HTML
from pathlib import Path
from typing import Optional
import logging

# Configure logging
logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s'
)
logger = logging.getLogger(__name__)

# Constants
EXCLUDED_DRUG_IDS = {
    'DB01098', 'DB00641', 'DB01382', 'DB00030', 'DB00046',
    'DB00047', 'DB00071', 'DB01306', 'DB01307', 'DB01309',
    'DB09456', 'DB09564', 'DB00641'
}

def query_database(db_path: str, sql_query: str, return_df: bool = False) ->
Optional[pd.DataFrame]:
    """
    Execute SQL query and return results.

    Args:
        db_path: Path to the database
        sql_query: SQL query string
        return_df: If True, return DataFrame; if False, display HTML
    """
    try:
        with sqlite3.connect(db_path) as conn:
            df = pd.read_sql_query(sql_query, conn)

        if return_df:
            return df
        else:
            display(HTML(df.to_html()))

    except sqlite3.Error as e:
        logger.error(f"Database error: {e}")
        raise
```

Додаток В

Підготовка даних для ML навчання з засобами які не володіють протидіабетичною дією згідно АТХ класифікатора

```
def get_filtered_drugs(db_path: str) -> pd.DataFrame:
    """
    Get filtered drug data excluding antidiabetic drugs and meeting specific criteria.

    Args:        db_path: Path to the database
    Returns:         pd.DataFrame: Filtered drugs data
    """
    # Validate database path
    if not Path(db_path).exists():
        raise FileNotFoundError(f"Database not found at {db_path}")
    # Get base data
    describe_df = query_database(db_path, 'SELECT * FROM describe', return_df=True)
    atc_df = query_database(db_path, 'SELECT * FROM ATC', return_df=True)

    # Filter ATC codes to only 3 characters
    atc_df = atc_df[atc_df['ATC'].str.len() == 3]

    # Get antidiabetic drugs IDs (A10 code)
    antidiabetic_ids = set(atc_df[atc_df['ATC'] == 'A10']['id'].unique())

    # Get all IDs that have codes other than A10
    drugs_with_other_codes = set(
        atc_df[
            (atc_df['id'].isin(antidiabetic_ids)) &
            (atc_df['ATC'] != 'A10')]['id'].unique())

    # Get antidiabetic drugs to exclude their IDs
    antidiabetic_df = get_antidiabetic_drugs(db_path)

    # All IDs to exclude
    all_excluded_ids =
EXCLUDED_DRUG_IDS.union(set(antidiabetic_df['id'])).union(drugs_with_other_codes)

    # Create filtered DataFrame
    filtered_df = describe_df[
        # Must have ATC code
        describe_df['id'].isin(atc_df['id'].unique()) &
        # Must have valid SMILES
        describe_df['SMILES'].notna() &
        (describe_df['SMILES'] != "Not Available") &
        # Exclude all specified IDs
        ~describe_df['id'].isin(all_excluded_ids)
    ][['id', 'Generic_Name', 'SMILES']]
    # Add ATC codes to the filtered DataFrame
    filtered_df = filtered_df.merge(
        atc_df[['id', 'ATC']],
        on='id',
        how='left'
    )
    # Remove duplicates, keeping the first occurrence
    filtered_df = filtered_df.drop_duplicates(subset=['id'], keep='first')
    logger.info(f"Total filtered drugs found: {len(filtered_df)}")
    logger.info(f"Unique drug IDs: {filtered_df['id'].nunique()}")
    logger.info(f"Unique ATC codes: {filtered_df['ATC'].nunique()}")

    return filtered_df
```

Додаток Г

Отримання хімічної класифікації сполуки через API ClassyFire

```
from typing import Optional, Dict, Any
import requests
import json
import time
from requests.exceptions import RequestException

def get_classyfire(smiles: str, query_id: Optional[str] = None, max_retries: int = 3) -
> Optional[str]:
    """
    Gets chemical classification of a compound via ClassyFire API.
    Args:
        smiles: SMILES notation of the chemical compound
        query_id: Optional query identifier
        max_retries: Maximum number of retry attempts when exceeding request limits
    Returns:
        str: JSON response from API as a string or None in case of error
    Raises:
        RequestException: For network interaction errors
        ValueError: For invalid API responses
    """
    BASE_URL = "http://classyfire.wishartlab.com/queries"
    payload: Dict[str, str] = {
        "label": query_id or "MyQueryLabel",
        "query_input": smiles,
        "query_type": "STRUCTURE"
    }
    headers: Dict[str, str] = {
        "Content-Type": "application/json",
        "Accept": "application/json"
    }

    def make_request(retry_count: int = 0) -> Optional[str]:
        try:
            # Send initial POST request
            initial_response = requests.post(
                BASE_URL,
                data=json.dumps(payload),
                headers=headers,
                timeout=30
            )
            initial_response.raise_for_status()
            data: Dict[str, Any] = initial_response.json()
            if 'error' in data:
                if data['error'] == 'Limit exceeded' and retry_count < max_retries:
                    print(f"Request limit exceeded. Waiting 61 seconds... (attempt
{retry_count + 1}/{max_retries})")
                    time.sleep(61)
                    return make_request(retry_count + 1)
                raise ValueError(f"API returned error: {data['error']}")
            if 'id' not in data:
                raise ValueError("Missing ID in API response")

            # Get results via GET request
            result_url = f"{BASE_URL}/{data['id']}.json"
            result_response = requests.get(result_url, timeout=30)
            result_response.raise_for_status()
            return result_response.text
```

Додаток Г (продовження)

```
        except RequestException as e:
            print(f"Network interaction error: {e}")
        except ValueError as e:
            print(f"Data processing error: {e}")
        except json.JSONDecodeError as e:
            print(f"JSON decoding error: {e}")

        return None

    return make_request()
```

Додаток Д

Парсинг ідентифікаторів та дескрипторів з відповіді API ClassyFire

```python
import json
from typing import Dict, Any, List

def parse_classyfire_ids(response_text: str) -> Dict[str, str]:
    """
    Extracts identifiers and names from the response of the ClassyFire API.

    Args:
        response_text: A JSON string from the ClassyFire API response

    Returns:
        Dict[str, str]: A dictionary where keys are identifiers and values are names

    Raises:
        json.JSONDecodeError: if the JSON is malformed
        KeyError: if required fields are missing in the data
    """

    result: Dict[str, str] = {}

    try:
        data = json.loads(response_text)

        # Get the first entity from the list
        if not data.get('entities') or not data['entities'][0]:
            return result

        entity = data['entities'][0]

        # Processing predicted_chebi_terms
        chebi_terms: List[str] = entity.get('predicted_chebi_terms', [])
        for term in chebi_terms:
            # Extract CHEBI ID and name from a string in the format "name (CHEBI:id)"
            if '(' in term and ')' in term:
                name, chebi_id = term.split(' (')
                chebi_id = chebi_id.rstrip(')')
                result[chebi_id] = name

        # Processing all nodes with CHEMONTID
        nodes_to_check = [
            entity.get('kingdom', {}),
            entity.get('superclass', {}),
            entity.get('class', {}),
            entity.get('subclass', {}),
            entity.get('direct_parent', {})
        ]
        nodes_to_check.extend(entity.get('alternative_parents', []))
        nodes_to_check.extend(entity.get('intermediate_nodes', []))

        for node in nodes_to_check:
            if node and 'chemont_id' in node and 'name' in node:
                result[node['chemont_id']] = node['name']

    except (json.JSONDecodeError, KeyError) as e:
        print(f"Error processing data: {e}")

    return result
```

Додаток Е

Розрахунки 3D структури для SMILES представлення досліджуваних сполук

```
import os
import subprocess
import pandas as pd
import logging
from colorama import Fore, Style

# Configure logging
logging.basicConfig(level=logging.INFO, format="%(message)s")

def log_success(message):
    """
    Logs a success message in green color.

    Args:
        message (str): The success message to log.
    """
    logging.info(Fore.GREEN + message + Style.RESET_ALL)

def log_error(message):
    """
    Logs an error message in red color.
    Args:
        message (str): The error message to log.
    """
    logging.error(Fore.RED + message + Style.RESET_ALL)

def generate_gjf_file(smiles: str, file_path: str):
    """
    Generates a .gjf file from a SMILES string using Open Babel.
    Args:
        smiles (str): The SMILES string representing the molecule.
        file_path (str): The path where the .gjf file will be saved.

    Returns:
        None
    """
    command = f'obabel -:"{smiles}" -O {file_path} --gen3d'
    process = subprocess.run(command, shell=True, stdout=subprocess.PIPE,
stderr=subprocess.PIPE)
    if process.returncode != 0:
        log_error(f"Error generating .gjf file ({file_path}):
{process.stderr.decode()}")
    else:
        log_success(f".gjf file generated successfully: {file_path}")

def run_gaussian(file_path: str, gauss_exe_dir: str):
    """
    Runs a Gaussian calculation on a .gjf file.

    Args:
        file_path (str): The path to the .gjf file.
        gauss_exe_dir (str): The directory containing the Gaussian executable.

    Returns:
        None
    """
```

Додаток Е (продовження)

```
    command = f"{gauss_exe_dir}/g16 {file_path}"
    process = subprocess.run(command, shell=True, stdout=subprocess.PIPE,
stderr=subprocess.PIPE)
    if process.returncode != 0:
        log_error(f"Error running Gaussian ({file_path}): {process.stderr.decode()}")
    else:
        log_success(f"Gaussian executed successfully: {file_path}")

def convert_to_mol(log_file: str, mol_file: str):
    """
    Converts a Gaussian .log file to a .mol file using Open Babel.

    Args:
        log_file (str): The path to the .log file.
        mol_file (str): The path where the .mol file will be saved.

    Returns:
        None
    """
    command = f"obabel {log_file} -O {mol_file}"
    process = subprocess.run(command, shell=True, stdout=subprocess.PIPE,
stderr=subprocess.PIPE)
    if process.returncode != 0:
        log_error(f"Error converting to .mol ({log_file}): {process.stderr.decode()}")
    else:
        log_success(f"Conversion to .mol completed successfully: {mol_file}")

def process_molecules(df: pd.DataFrame, save_path: str):
    """
    Processes a DataFrame of molecules, generating .gjf files, running Gaussian
calculations,
    and converting the results to .mol files.

    Args:
        df (pd.DataFrame): A DataFrame containing molecule IDs and SMILES strings.
        save_path (str): The directory where output files will be saved.
    Returns:
        None
    """
    os.makedirs(save_path, exist_ok=True)  # Create the directory if it doesn't exist

    gauss_exe_dir = os.environ.get("GAUSS_EXEDIR")
    if not gauss_exe_dir:
        log_error("Environment variable GAUSS_EXEDIR is not set.")
        return

    for index, row in df.iterrows():
        name_file = os.path.join(save_path, f"{row['id']}.gjf")
        log_file = os.path.join(save_path, f"{row['id']}.log")
        mol_file = os.path.join(save_path, f"{row['id']}.mol")
        chk_file = os.path.join(save_path, f"{row['id']}.chk")

        generate_gjf_file(row["SMILES"], name_file)

        if not os.path.exists(name_file):
            log_error(f"File {name_file} not created, skipping processing.")
            continue
```

Додаток Е (продовження)

```
        new_lines = [
            "%NProcShared=6\n",
            "%mem=5GB\n",
            f"%chk={chk_file}\n",
            "# PM6 Opt\n",
            "\n",  # Empty line
            "Title Card\n",
            "\n"
        ]

        with open(name_file, "r") as file:
            lines = file.readlines()

        if len(lines) < 5:
            log_error(f"Error: file {name_file} has too few lines.")
            continue

        lines = new_lines + lines[5:]

        with open(name_file, "w") as file:
            file.writelines(lines)

        run_gaussian(name_file, gauss_exe_dir)
        convert_to_mol(log_file, mol_file)

log_success("Processing completed!")
```

Додаток Є

Порівняльний аналіз моделей класифікації

```
import pickle
import pandas as pd
from collections import Counter
from imblearn.under_sampling import RandomUnderSampler #imbalanced-learn==0.13.0,
imblearn==0.0
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression, SGDClassifier, RidgeClassifier
from sklearn.naive_bayes import MultinomialNB, BernoulliNB
from sklearn.svm import LinearSVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.preprocessing import MinMaxScaler
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import StratifiedKFold, cross_validate
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
roc_auc_score
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LogisticRegression, RidgeClassifier, SGDClassifier
from sklearn.naive_bayes import BernoulliNB, MultinomialNB
from sklearn.svm import LinearSVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import MinMaxScaler
from xgboost import XGBClassifier # xgboost==2.1.4
from sklearn.tree import DecisionTreeClassifier
from sklearn.calibration import CalibratedClassifierCV

with open("DATA_drug_classyfire.pkl", "rb") as f:
    df = pickle.load(f)

with open("models.pkl", "rb") as f:
    models = pickle.load(f)

def calculate_confusion_metrics(y_true, prob_column, threshold=0.5):
    """
    Calculate confusion matrix metrics (TP, FP, TN, FN) for probability predictions

    Parameters:
    -----------
    y_true : array-like
        True binary labels
    prob_column : array-like
        Probability predictions from the model
    threshold : float, default=0.5
        Decision threshold for converting probabilities to binary predictions
```

Додаток Є (продовження)

```
    Returns:
    --------
    dict
        Dictionary containing TP, FP, TN, FN counts
    """
    # Convert probabilities to binary predictions using threshold
    # y_pred = (prob_column >= threshold).astype(int)
    y_pred = np.where(prob_column >= threshold, 1, -1)
    # Calculate confusion matrix
    tn, fp, fn, tp = confusion_matrix(y_true, y_pred).ravel()
        # Calculate additional metrics
    # Compute performance metrics and round to 3 decimal places
    accuracy = round(accuracy_score(y_true, y_pred), 3)
    precision = round(precision_score(y_true, y_pred), 3)
    recall = round(recall_score(y_true, y_pred), 3)
    f1 = round(f1_score(y_true, y_pred), 3)
    roc_auc = round(roc_auc_score(y_true, prob_column), 3)

    return {
        'True Positives (TP)': tp,
        'False Positives (FP)': fp,
        'True Negatives (TN)': tn,
        'False Negatives (FN)': fn,
        'Accuracy': accuracy,
        'Precision': precision,
        'Recall': recall,
        'F1 Score': f1,
        'ROC AUC': roc_auc
    }

def evaluate_models(models, X, y, threshold=0.5):
    """
    Trains models, calibrates where necessary, makes predictions, and calculates
evaluation metrics.

    Parameters:
        models (dict): Dictionary of models with names as keys.
        X (array-like): Feature matrix.
        y (array-like): Target labels.
        threshold (float, optional): Threshold for classification. Defaults to 0.5.

    Returns:
        pd.DataFrame: DataFrame with evaluation metrics for each model.
    """
    predictions = {}

    for model_name, model in models.items():
        if isinstance(model, (LinearSVC, SGDClassifier, RidgeClassifier)):
            calibrated = CalibratedClassifierCV(model)
            calibrated.fit(X, y)
            proba = calibrated.predict_proba(X)
        else:
```

Додаток Є (продовження)

```
            proba = model.predict_proba(X)

        predictions[f'{model_name}_p'] = proba[:, 1]  # Store positive class
probabilities

    # Convert predictions to DataFrame
    P_models = pd.DataFrame.from_dict(predictions)

    # Identify probability columns
    prob_columns = [col for col in P_models.columns if col.endswith('_p')]

    # Calculate metrics for each model
    results = []
    for col in prob_columns:
        metrics = calculate_confusion_metrics(y, P_models[col], threshold=threshold)
        metrics['Model'] = col.replace('_p', '')
        results.append(metrics)

    return pd.DataFrame(results).set_index('Model')

def evaluate_models_with_stats(df, models, n_runs=10, test_size=0.3, threshold=0.5,
random_state=42):
    """
    Performs multiple evaluations of models and calculates statistics for metrics.

    Parameters:
        df (pd.DataFrame): Original dataset
        models (dict): Dictionary of models with names as keys
        n_runs (int): Number of evaluation runs
        test_size (float): Test set proportion
        threshold (float): Classification threshold
        random_state (int): Random state for reproducibility

    Returns:
        tuple: (means_df, stds_df) DataFrames with means and standard deviations of
metrics
    """
    # Initialize lists to store results from each run
    all_results = []

    for run in range(n_runs):
        # Perform undersampling
        rus = RandomUnderSampler(random_state=random_state + run)
        X = df.drop(['Label'], axis=1) # features
        y = df['Label'] # target variable
        X_resampled, y_resampled = rus.fit_resample(X, y)

        # Create balanced dataset
        df_balanced = pd.concat([
            pd.DataFrame(X_resampled, columns=X.columns),
            pd.Series(y_resampled, name='Label')
```

Додаток Є (продовження)

```
        ], axis=1)

        # Split features and target
        X = df_balanced.drop(columns="Label")
        y = df_balanced["Label"]

        # Create train/test split
        X_train, X_test, y_train, y_test = train_test_split(
            X, y, test_size=test_size,
            random_state=random_state + run
        )

        # Evaluate models
        results = evaluate_models(models, X_test, y_test, threshold)
        results['Run'] = run
        all_results.append(results)

    # Combine all results
    combined_results = pd.concat(all_results, axis=0)
    #return combined_results
    # # Calculate means and standard deviations
    means = combined_results.groupby(combined_results.index).mean()
    stds = combined_results.groupby(combined_results.index).std()

    # # Drop the 'Run' column from results
    means = means.drop('Run', axis=1)
    stds = stds.drop('Run', axis=1)

    return means, stds

def format_results_with_stats(means, stds, metrics_to_display=None):
    """
    Formats the results with means and standard deviations.

    Parameters:
        means (pd.DataFrame): DataFrame with mean values
        stds (pd.DataFrame): DataFrame with standard deviations
        metrics_to_display (list): List of metrics to include in output

    Returns:
        pd.DataFrame: Formatted results with means ± standard deviations
    """
    if metrics_to_display is None:
        metrics_to_display = means.columns

    formatted_results = pd.DataFrame(index=means.index)

    for metric in metrics_to_display:
        formatted_results[metric] = means[metric].round(3).astype(str) + ' ± ' + \
                                    stds[metric].round(3).astype(str)

    return formatted_results
```

Додаток Є (продовження)

```
# Run the evaluation
means, stds = evaluate_models_with_stats(
    df=df,  # your original dataframe
    models=models,   # your dictionary of models
    n_runs=100       # number of runs to perform
)

# Format and display results
formatted_results = format_results_with_stats(means, stds)
```

Додаток Ж

Підготовка та навчання класифікаторів для прогнозування протидіабетичних препаратів

```
import pickle
from imblearn.under_sampling import RandomUnderSampler #imbalanced-learn==0.13.0,
imblearn==0.0
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression, SGDClassifier, RidgeClassifier
from sklearn.naive_bayes import MultinomialNB, BernoulliNB
from sklearn.svm import LinearSVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.preprocessing import MinMaxScaler
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import StratifiedKFold, cross_validate
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
roc_auc_score
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LogisticRegression, RidgeClassifier, SGDClassifier
from sklearn.naive_bayes import BernoulliNB, MultinomialNB
from sklearn.svm import LinearSVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import MinMaxScaler
from xgboost import XGBClassifier # xgboost==2.1.4
from sklearn.tree import DecisionTreeClassifier

with open("classyfire_dict_data_non_diabetic.pkl", "rb") as f:
    non_diabetic_drug_classes = pickle.load(f)
with open("classyfire_dict_data_diabetic.pkl", "rb") as f:
    diabetic_drug_classes = pickle.load(f)
# Collect all unique features (CHEBI, CHEMONTID, etc.)
unique_features = set()
for drug_classes in (non_diabetic_drug_classes, diabetic_drug_classes):
    for drug_id, features in drug_classes.items():
        unique_features.update(features.keys())
# Create DataFrame
data_rows = []
for drug_classes, label in zip((non_diabetic_drug_classes, diabetic_drug_classes), (-1,
1)):
    for drug_id, features in drug_classes.items():
        row = {feature: 1 if feature in features else -1 for feature in
unique_features}
        row['Drug_ID'] = drug_id  # Drug identifier
        row['Label'] = label  # Class label
        data_rows.append(row)
df = pd.DataFrame(data_rows).set_index('Drug_ID')
# Create and apply the sampler
rus = RandomUnderSampler(random_state=42)
X = df.drop(['Label'], axis=1) # features
y = df['Label'] # target variable

# Perform undersampling
X_resampled, y_resampled = rus.fit_resample(X, y)
```

Додаток Ж (продовження)

```
# Collect the data back into the dataframe
df_balanced = pd.concat([pd.DataFrame(X_resampled, columns=X.columns),
                         pd.Series(y_resampled, name='Label')], axis=1)

X = df_balanced.drop(columns="Label")
y = df_balanced["Label"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

models = {
    'Logistic Regression': LogisticRegression(solver='liblinear'),
    'BernoulliNB' : BernoulliNB(),
    'Multinomial Naive Bayes': make_pipeline(MinMaxScaler(), MultinomialNB()),
    'Linear SVM': LinearSVC(),
    'k-NN': KNeighborsClassifier(),
    'Random Forest': RandomForestClassifier(),
    'LDA': LinearDiscriminantAnalysis(),
    'SGD Classifier': SGDClassifier(),
    'Lasso Regression': LogisticRegression(penalty='l1', solver='saga'),
    'Ridge Regression': RidgeClassifier(alpha=1.0, solver='auto'),
    'Elastic Net': LogisticRegression(penalty='elasticnet', l1_ratio=0.5,
solver='saga'),
    'Decision Tree': DecisionTreeClassifier(criterion='gini', max_depth=None),
    'Gradient Boosting': GradientBoostingClassifier(n_estimators=100,
learning_rate=0.1),
}

# Define cross-validation strategy
cv = StratifiedKFold(n_splits=12, shuffle=True, random_state=42)

for name, model in models.items():
    # Cross-validation
    scores = cross_validate(model, X_train, y_train, cv=cv, scoring=scoring, n_jobs=-1)

    # Train the model on the entire training set
    model.fit(X_train, y_train)
```

Додаток 3

Навчання Char-RNN нейромережі в Pytorch

```
from rdkit import Chem
from rdkit.Chem import MolToSmiles
import tqdm
import pickle
# Declaring the model
import numpy as np
import torch
from torch import nn
import torch.nn.functional as F


class CharRNN(nn.Module):

    def __init__(self, tokens, n_hidden=10, n_layers=2,
                               drop_prob=0.2, lr=0.001):# n_hidden=256,
        super().__init__()
        self.drop_prob = drop_prob
        self.n_layers = n_layers
        self.n_hidden = n_hidden
        self.lr = lr

        # creating character dictionaries
        self.chars = tokens
        self.int2char = dict(enumerate(self.chars))
        self.char2int = {ch: ii for ii, ch in self.int2char.items()}

        #define the LSTM
        self.lstm = nn.LSTM(len(self.chars), n_hidden, n_layers,
                            dropout=drop_prob, batch_first=True)

        #define a dropout layer
        self.dropout = nn.Dropout(drop_prob)

        #define the final, fully-connected output layer
        self.fc = nn.Linear(n_hidden, len(self.chars))


    def forward(self, x, hidden):
        ''' Forward pass through the network.
            These inputs are x, and the hidden/cell state `hidden`. '''

        #get the outputs and the new hidden state from the lstm
        r_output, hidden = self.lstm(x, hidden)

        #pass through a dropout layer
        out = self.dropout(r_output)

        # Stack up LSTM outputs using view
        out = out.contiguous().view(-1, self.n_hidden)

        #put x through the fully-connected layer
        out = self.fc(out)

        # return the final output and the hidden state
        return out, hidden
```

Додаток З (продовження)

```
    def init_hidden(self, batch_size):
        ''' Initializes hidden state '''
        # Create two new tensors with sizes n_layers x batch_size x n_hidden,
        # initialized to zero, for hidden state and cell state of LSTM
        weight = next(self.parameters()).data
        # Check if GPU is available
        train_on_gpu = torch.cuda.is_available()
        if (train_on_gpu):
            hidden = (weight.new(self.n_layers, batch_size,
self.n_hidden).zero_().cuda(),
                  weight.new(self.n_layers, batch_size, self.n_hidden).zero_().cuda())
        else:
            hidden = (weight.new(self.n_layers, batch_size, self.n_hidden).zero_(),
                      weight.new(self.n_layers, batch_size, self.n_hidden).zero_())

        return hidden
# Defining method to encode one hot labels
def one_hot_encode(arr, n_labels):

    # Initialize the the encoded array
    one_hot = np.zeros((np.multiply(*arr.shape), n_labels), dtype=np.float32)

    # Fill the appropriate elements with ones
    one_hot[np.arange(one_hot.shape[0]), arr.flatten()] = 1.

    # Finally reshape it to get back to the original array
    one_hot = one_hot.reshape((*arr.shape, n_labels))

    return one_hot
# Defining method to make mini-batches for training
def get_batches(arr, batch_size, seq_length):
    '''Create a generator that returns batches of size
       batch_size x seq_length from arr.

       Arguments
       ---------
       arr: Array you want to make batches from
       batch_size: Batch size, the number of sequences per batch
       seq_length: Number of encoded chars in a sequence
    '''
    batch_size_total = batch_size * seq_length
    # total number of batches we can make
    n_batches = len(arr)//batch_size_total

    # Keep only enough characters to make full batches
    arr = arr[:n_batches * batch_size_total]
    # Reshape into batch_size rows
    arr = arr.reshape((batch_size, -1))

    # iterate through the array, one sequence at a time
    for n in range(0, arr.shape[1], seq_length):
        # The features
        x = arr[:, n:n+seq_length]
        # The targets, shifted by one
        y = np.zeros_like(x)
        try:
            y[:, :-1], y[:, -1] = x[:, 1:], arr[:, n+seq_length]
```

Додаток З (продовження)

```
        except IndexError:
            y[:, :-1], y[:, -1] = x[:, 1:], arr[:, 0]
        yield x, y

# Declaring the train method
def train(net, data, epochs=1, batch_size=32, seq_length=100, lr=0.001, clip=5,
val_frac=0.1, print_every=10000):
    ''' Training a network

        Arguments
        ---------

        net: CharRNN network
        data: text data to train the network
        epochs: Number of epochs to train
        batch_size: Number of mini-sequences per mini-batch, aka batch size
        seq_length: Number of character steps per mini-batch
        lr: learning rate
        clip: gradient clipping
        val_frac: Fraction of data to hold out for validation
        print_every: Number of steps for printing training and validation loss

    '''
    net.train()

    opt = torch.optim.Adam(net.parameters(), lr=lr)
    criterion = nn.CrossEntropyLoss()

    # create training and validation data
    val_idx = int(len(data)*(1-val_frac))
    data, val_data = data[:val_idx], data[val_idx:]
    # Check if GPU is available
    train_on_gpu = torch.cuda.is_available()
    #if(train_on_gpu):
        #print('Training on GPU!')
    #else:
        #print('No GPU available, training on CPU; consider making n_epochs very
small.')
    if(train_on_gpu):
        net.cuda()
    v_l = 3
    counter = 0
    n_chars = len(net.chars)
    for e in tqdm.tqdm(range(epochs)):
        # initialize hidden state
        h = net.init_hidden(batch_size)

        for x, y in get_batches(data, batch_size, seq_length):
            counter += 1

            # One-hot encode our data and make them Torch tensors
            x = one_hot_encode(x, n_chars)
            inputs, targets = torch.from_numpy(x), torch.from_numpy(y)

            if(train_on_gpu):
                inputs, targets = inputs.cuda(), targets.cuda()
            # Creating new variables for the hidden state, otherwise
            # we'd backprop through the entire training history
            h = tuple([each.data for each in h])
```

Додаток З (продовження)

```
            # zero accumulated gradients
            net.zero_grad()

            # get the output from the model
            output, h = net(inputs, h)

            # calculate the loss and perform backprop
            loss = criterion(output, targets.view(batch_size*seq_length).long())
            loss.backward()
            # `clip_grad_norm` helps prevent the exploding gradient problem in RNNs /
LSTMs.
            nn.utils.clip_grad_norm_(net.parameters(), clip)
            opt.step()

            # loss stats
            if counter % print_every == 0:
                # Get validation loss
                val_h = net.init_hidden(batch_size)
                val_losses = []
                net.eval()
                for x, y in get_batches(val_data, batch_size, seq_length):
                    # One-hot encode our data and make them Torch tensors
                    x = one_hot_encode(x, n_chars)
                    x, y = torch.from_numpy(x), torch.from_numpy(y)

                    # Creating new variables for the hidden state, otherwise
                    # we'd backprop through the entire training history
                    val_h = tuple([each.data for each in val_h])

                    inputs, targets = x, y
                    if(train_on_gpu):
                        inputs, targets = inputs.cuda(), targets.cuda()

                    output, val_h = net(inputs, val_h)
                    val_loss = criterion(output,
targets.view(batch_size*seq_length).long())

                    val_losses.append(val_loss.item())

                net.train() # reset to train mode after iterationg through validation
data
                Loss = loss.item()
                Val_Loss = np.mean(val_losses)
                print("Epoch: {}/{}...".format(e+1, epochs),
                      "Step: {}...".format(counter),
                      "Loss: {:.4f}...".format(Loss),
                      "Val Loss: {:.4f}".format(Val_Loss))

                if (v_l > Val_Loss) & (counter>0):
                     v_l = Val_Loss
                     # Saving the model
                     checkpoint = {'n_hidden': net.n_hidden,
                                   'n_layers': net.n_layers,
                                   'state_dict': net.state_dict(),
                                   'tokens': net.chars}
                     torch.save(net, '{}_model_{}_{}.pt'.format(counter, Loss,
Val_Loss))
```

Додаток З (продовження)

```
def generate_smiles_variants(smiles_str):
    mol = Chem.MolFromSmiles(smiles_str)
    smiles_variants = set()
    for i in range(10000):  # Number of generation attempts
        smiles_variants.add(MolToSmiles(mol, canonical=False, doRandom=True))
    return smiles_variants

with open("antidiabetic_df.pkl", "rb") as f:
   df = pickle.load(f)

# Generate SMILES variants and write to file
with open('ANTIDIABETIC_smiles_variants.txt', 'w') as file:
    for index, row in tqdm.tqdm(df.iterrows(), total=len(df)):
        variants = generate_smiles_variants(row['SMILES'])
        for variant in variants:
            file.write(variant + '\n')

with open('ANTICANCER_smiles_variants.txt','r') as f:#smiles_variants.txt
    text = f.read()
# Create two dictionaries:
# 1. int2char, which maps integers to characters
# 2. char2int, which maps characters to integers
chars = tuple(set(text))
int2char = dict(enumerate(chars))
char2int = {ch: ii for ii, ch in int2char.items()}

# Encode the text
encoded = np.array([char2int[ch] for ch in text])

# Check if GPU is available
train_on_gpu = torch.cuda.is_available()
if(train_on_gpu):
    print('Training on GPU!')
else:
    print('No GPU available, training on CPU; consider making n_epochs very small.')


n_hidden=100
n_layers=5

net = CharRNN(chars, n_hidden, n_layers)

if __name__ == "__main__":
# Declaring the hyperparameters
    batch_size = 32
    seq_length = 100
    n_epochs = 4 # start smaller if you are just testing initial behavior
# DATA is encoded
# train the model
    train(net, encoded, epochs=n_epochs, batch_size=batch_size, seq_length=seq_length,
lr=0.001, print_every=1000)

# Saving the model
    checkpoint = {'n_hidden': net.n_hidden,
              'n_layers': net.n_layers,
              'state_dict': net.state_dict(),
              'tokens': net.chars}
    torch.save(net,'model.pt')
```

Генерування молекул за допомогою Char-RNN в Pytorch

```
import sys
import os
from rdkit import Chem
from IPython.display import display


def predict(net, char, h=None, top_k=None):
    ''' Given a character, predict the next character.
        Returns the predicted character and the hidden state.
    '''
    # tensor inputs
    x = np.array([[net.char2int[char]]])
    x = one_hot_encode(x, len(net.chars))
    inputs = torch.from_numpy(x)
    train_on_gpu = torch.cuda.is_available()
    if (train_on_gpu):
         inputs = inputs.cuda()
    # detach hidden state from history
    h = tuple([each.data for each in h])
    # get the output of the model
    out, h = net(inputs, h)
    # get the character probabilities
    p = F.softmax(out, dim=1).data
    # train_on_gpu = torch.cuda.is_available()
    if (train_on_gpu):
        p = p.cpu()  # move to cpu

    # get top characters
    if top_k is None:
        top_ch = np.arange(len(net.chars))
    else:
        p, top_ch = p.topk(top_k)
        top_ch = top_ch.numpy().squeeze()

    # select the likely next character with some element of randomness
    p = p.numpy().squeeze()
    char = np.random.choice(top_ch, p=p / p.sum())

    # return the encoded value of the predicted char and the hidden state
    return net.int2char[char], h


def sample(net, size, prime='B', top_k=None):
    # Check if GPU is available
    train_on_gpu = torch.cuda.is_available()

    if(train_on_gpu):
        net.cuda()
    else:
        net.cpu()

    net.eval() # eval mode

    # First off, run through the prime characters
    chars = [ch for ch in prime]
    h = net.init_hidden(1)
    for ch in prime:
        char, h = predict(net, ch, h, top_k=top_k)
```

Додаток І (продовження)

```
        chars.append(char)

        # Now pass in the previous character and get a new one
        for ii in range(size):
            char, h = predict(net, chars[-1], h, top_k=top_k)
            chars.append(char)

        return ''.join(chars)

# Save the current output state
old_stdout = sys.stdout
old_stderr = sys.stderr

# Redirect the output to devnull
sys.stdout = open(os.devnull, 'w')
sys.stderr = open(os.devnull, 'w')

net=torch.load('model.pt', weights_only=False)
# Generating new text (NEW SMILES)
t = sample(net, 10000, prime='C', top_k=5)
t = t.split('\n')

valid_smiles = []
for s in t:
    try:
        mol = Chem.MolFromSmiles(s)
        if mol:  # check if the parsing was successful
            display(mol)
            valid_smiles.append(s)
        else:
            pass
    except Exception as e:
        print(f"Error parsing SMILES '{s}': {e}")
```