Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

Галузь знань 12 – Інформаційні технології

Спеціальність 123 –Комп'ютерна інженерія

на тему «Універсальна система автоматизованого тренування та моніторингу моделей машинного навчання для SAAS платформи in silico розробки ліків. On-premise деплоймент на інфраструктурі фармакологічних компаній за допомогою оркестратора Airflow»

КвРКІП. 2202145.22.02.30 ПЗ

Виконав: студент 2 курсу, група КІ2м-22-2 Возняк В.З.
Підпис Ініціали, прізвище

Керівник доктор філософії, доцент Павлова О.О.
Науковий ступінь, вчене звання Підпис Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КІІС, д.т.н., проф.
Т.О. Говорущенко
21 05 2024 р.

Хмельницький, 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Комп'ютерної інженерії та інформаційних систем

Освітній рівень магістр

Галузь знань 12 Інформаційні технології

Спеціальність 123 Комп'ютерна інженерія

Освітня програма освітньо-наукова програма «Комп'ютерна інженерія та програмування»

ЗАТВЕРДЖУЮ

Зав. кафедри Т.О.Говорущенко

" 01 " 09 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

Возняку Володимиру Зіновійовичу
Прізвище, ім'я, по батькові студента

. Тема проекту (роботи) Універсальна система автоматизованого тренування та моніторингу моделей машинного навчання для SAAS платформи in silico розробки ліків. деплоймент на інфраструктурі фармакологічних компаній за допомогою оркестратора

Керівник проекту (роботи) Павлова О.О., д.ф., доцент
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 01.01.2024 р. № 1

2. Строк подання студентом проекту (роботи) на кафедру 01.05.2024 р.

3. Вихідні дані до проекту (роботи) Завдання на дипломне проектування

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Аналіз відомих методів для побудови SAAS платформи in silico розробки ліків

Розробка On-Premise версії автоматизованого пайплайну

Впровадження пайплайну в SAAS платформу для розробки ліків

Тренування моделей для передбачення ADME-Tox параметрів молекул

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Презентація

6. Консультанти розділів кваліфікаційної роботи магістра

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Лисенко С.М., професор кафедри КІІС		
Антиплагіат	Нічепорук А.О., доцент кафедри КІІС		

7. Дата видачі завдання «_01_»_____09_____2023р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) кваліфікаційної роботи магістра	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики КвРМ з керівником	01.09.2023	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.10.2023	виконано
3	Робота над розділом 1 – аналіз відомих моделей, методів за темою; постановка задачі	01.11.2023	виконано
4	Робота над розділом 2 – розробка моделей для вирішення поставленої задачі	01.12.2023	виконано
5	Робота над науковою статтею	01.02.204	виконано
6	Робота над розділом 3 – розробка методів для вирішення поставленої задачі	15.02.2024	виконано
7	Робота над розділом 4 – проектування та розробка ПЗ для вирішення поставленої задачі, експериментальна частина	01.04.204	виконано
8	Оформлення пояснювальної записки згідно вимог	18.04.2024	виконано
9	Попередній захист ДРМ	29.04.2024	виконано
10	Захист ДРМ на засіданні ЕК	До 15.05.2024	

Студент _____ Підпис Возняк В.З. Ініціали, прізвище

Керівник роботи _____ Підпис Павлова О.О. Ініціали, прізвище

РЕФЕРАТ

Тема кваліфікаційної роботи магістра: Універсальна система автоматизованого тренування та моніторингу моделей машинного навчання для SAAS платформи in silico розробки ліків. On-premise деплоймент на інфраструктурі фармакологічних компаній за допомогою оркестратора Airflow.

Автор роботи: Возняк Володимир Зіновійович.

Керівник роботи: Павлова Ольга Олександрівна.

Пояснювальна записка: 81 с., 35 рис., 9 табл., 3 дод., 61 джерело.

AUTOML, DRUG DISCOVERY, SAAS, MACHINE LEARNING, PIPELINES, ON-PREMISE

Об’єктом дослідження є автоматизовані пайплайни машинного навчання.

Предметом дослідження є автоматизовані пайплайни машинного навчання для біологічних задач з розробки нових ліків для On-Premise інфраструктур.

Метою кваліфікаційної роботи магістра є створення автоматизованого пайплайну для тренування моделей машинного навчання на біологічних даних, який володіє властивостями модульності, детального налаштування конфігурації, паралелізації, швидкого масштабування та деплойменту у двох сценаріях – On-Cloud та On-Premise. Розроблений пайплайн повинен бути інтегрований в якості сервісу в існуючу SAAS платформу in silico розробки ліків.

Для розв’язання поставлених задач використовувалися методи машинного навчання, глибокого навчання та системного проєктування.

Наукова новизна отриманих результатів:

− розроблено On-Premise версію автоматизованого пайплайну повного циклу для тренування моделей машинного навчання на біологічних даних;

− розроблений пайплайн впроваджено в SAAS платформу для in silico розробки ліків;

− за допомогою створеного пайплайну натреновано моделі для передбачення ADME-Tox параметрів молекул та інтегровано їх у SAAS платформу.

На основі проведених досліджень розроблена архітектура і компоненти програмного забезпечення, яке інтегровано в існуючу SAAS платформу з розробки ліків.

Практична значимість отриманих результатів полягає у можливості автоматичного тренування моделей машинного навчання на молекулярних датасетах і подальшого їх використання у SAAS платформі.

У першому розділі проведено аналіз існуючих рішень у галузі автоматизації пайплайнів машинного навчання. Розглянуто основні компоненти пайплайнів, застосування пайплайну для тренування моделей машинного навчання на біологічних даних та проведено огляд літератури.

У другому розділі описано математичну модель основних компонентів пайплайну. Розглянуто принцип роботи таких алгоритмів класичного машинного навчання, як лінійна регресія, логістична регресія, випадковий ліс, нейронні мережі тощо. Описано техніки вибору найкращих ознак та принципів оптимізації моделей.

У третьому розділі детально описано реалізацію версії пайплайну для On-Premise деплойменту на інфраструктурі фармакологічних компаній за допомогою оркестратора Airflow. Також наводиться порівняння різних технологій для реалізації поставленої задачі і обґрунтовується фінальне рішення.

У четвертому розділі міститься опис інтеграції пайплайну в SAAS платформу для розробки ліків. Також наведено результати натренованих моделей за допомогою автоматизованого пайплайну на датасетах ADME-Tox.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ML – Machine Learning (машинне навчання)

AutoML – Automated Machine Learning (автоматизоване машинне навчання)

AWS – Amazon Web Services (Амазон веб сервіси)

GCP – Google Cloud Platform (Гугл Хмарна Платформа)

CPU – Central Processing Unit (центральний процесор)

GPU – Graphics Processing Unit (графічний процесор)

SVM – Support Vector Machine (машина опорних векторів)

MLP – Multi-Layer Perceptron (багатошаровий перцептрон)

SAAS – Software as a Service (програмне забезпечення як сервіс)

On-Premise – Он-преміс (локальне розміщення)

RDS – Relational Database Service (сервіс реляційних баз даних)

VPC – Virtual Private Cloud (віртуальна приватна хмара)

JSON – JavaScript Object Notation (формат об'єктної нотації JavaScript)

SQL – Structured Query Language (мова структурованих запитів)

URI – Uniform Resource Identifier (уніфікований ідентифікатор ресурсів)

CSV – Comma-Separated Values (значення, розділені комами)

ВСТУП

Сучасний світ науки та технологій переживає безпрецедентний розвиток інновацій, що прискорюються завдяки стрімкому прогресу в області інформаційних технологій. Особливе місце в цій революції займає машинне навчання (ML), яке є однією з найбільш впливових областей штучного інтелекту (AI). Машинне навчання здатне трансформувати різні галузі людської діяльності, пропонуючи нові підходи до аналізу даних, прийняття рішень та автоматизації процесів.

Ця дипломна робота зосереджена на розробці та інтеграції автоматизованих пайплайнів машинного навчання для SAAS платформи, спрямованої на in silico розробку лікарських засобів. Інноваційність дослідження полягає у впровадженні новітніх технологій AutoML, хмарних обчислень та передових методик оркестрації пайплайнів, що дозволяють оптимізувати процеси розробки та тестування нових фармацевтичних препаратів.

Завданнями роботи є:

- аналіз існуючих технологій оркестрації контейнерів та автоматизації процесів машинного навчання;
- розробка масштабованого та гнучкого пайплайну для тренування і оцінки моделей ML, який може бути інтегрований у SAAS платформу;
- порівняння провайдерів хмарних послуг для визначення оптимального середовища для деплойменту пайплайнів.

Ця робота спрямована на поглиблення розуміння та практичне застосування сучасних технологій машинного навчання у фармацевтичній індустрії, сприяючи швидшому та ефективнішому розвитку нових лікарських препаратів.

Метою кваліфікаційної роботи магістра є створення автоматизованого пайплайну для тренування моделей машинного навчання на біологічних даних, який володіє властивостями модульності, детального налаштування конфігурації, паралелізації, швидкого масштабування та деплойменту у двох сценаріях – On-

Cloud та On-Premise. Розроблений пайплайн повинен бути інтегрований в якості сервісу в існуючу SAAS платформу in silico розробки ліків.

Для розв’язання поставлених задач використовувалися методи машинного навчання, глибокого навчання та системного проєктування.

Наукова новизна отриманих результатів:

− розроблено On-Premise версію автоматизованого пайплайну повного циклу для тренування моделей машинного навчання на біологічних даних;

− розроблений пайплайн впроваджено в SAAS платформу для in silico розробки ліків;

− за допомогою створеного пайплайну натреновано моделі для передбачення ADME-Tox параметрів молекул та інтегровано їх у SAAS платформу.

На основі проведених досліджень розроблена архітектура і компоненти програмного забезпечення, яке інтегровано в існуючу SaaS платформу з розробки ліків.

Практична значимість отриманих результатів полягає у можливості автоматичного тренування моделей машинного навчання на молекулярних датасетах і подальшого їх використання у SAAS платформі.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ У ГАЛУЗІ АВТОМАТИЗАЦІЇ ПАЙПЛАЙНІВ МАШИННОГО НАВЧАННЯ

1.1 Основні складові пайплайну автоматичного тренування моделей машинного навчання

Автоматизоване машинне навчання (AutoML) є ефективним засобом для спрощення процесу впровадження машинного навчання у різноманітні задачі, від початкової стадії до її завершення. Головна задача AutoML – це зменшення або повне усунення необхідності у фахівцях, що значно прискорює розробку та впровадження моделей машинного навчання. Пайплайн AutoML включає ряд послідовних кроків, кожен із яких має свої специфічні завдання в рамках процесу машинного навчання. Основні компоненти пайплайну AutoML охоплюють попередню обробку даних, інженерію ознак, вибір моделі та налаштування її гіперпараметрів. У даному розділі детально розглядаються ключові елементи пайплайну AutoML та їх роль у спрощенні процесу машинного навчання.

Першою фазою AutoML є попередня обробка даних. Ця фаза має велике значення, адже вона готує вихідні дані для подальшого аналізу. До цього етапу належать завдання, як-от очищення даних, оброблення випадків відсутності даних, їх трансформація та нормалізація. Основна ціль полягає в забезпеченні високоякісних даних для наступних етапів, щоб уникнути помилок, які можуть погіршити ефективність моделей.

Другою фазою AutoML є очищення даних. Це процедура виявлення та коригування (або вилучення) помилок та несумісностей у даних для підвищення їхньої якості. Обробка відсутніх значень: методики для вирішення проблем з відсутніми даними, включаючи імпутацію, видалення та інші алгоритмічні підходи. Трансформація даних: проведення таких операцій, як масштабування, нормалізація, та кодування категоріальних змінних. Вибір та зменшення кількості ознак: відбір найефективніших ознак або зменшення об'єму даних.

Третьою фазою є інженерія ознак. Інженерія ознак – це процес створення нових ознак з наявних даних, що допомагає підвищити ефективність моделей. У

рамках пайплайну AutoML цей процес автоматизовано, що дозволяє швидко генерувати нові ознаки. Цей етап включає такі аспекти:

- екстракція ознак – видобуток нових ознак з існуючих даних;
- відбір ознак – вибір найбільш інформативних ознак із загального набору;
- кодування ознак – перетворення категоріальних ознак у числовий формат
- масштабування ознак – забезпечення однакового масштабу ознак для уникнення упередженості моделі.

Наступною фазою AutoML є налаштування гіперпараметрів. Налаштування гіперпараметрів є ключовим процесом у вдосконаленні продуктивності обраної моделі. У контексті AutoML, цей процес здійснюється автоматично, використовуючи передові методи оптимізації, такі як баєсівська оптимізація, грід-пошук або випадковий пошук.

Методи оптимізації гіперпараметрів включають:

- грід-пошук – систематичний пошук у передбачуваній сітці гіперпараметрів;
- випадковий пошук – вибірка гіперпараметрів з визначеного діапазону випадковим чином;
- баєсівська оптимізація – метод оптимізації, що базується на ймовірнісному підході;
- еволюційні алгоритми – алгоритми, що використовують принципи природного відбору для оптимізації.

Четвертою фазою є візуалізація даних. Візуалізація даних є незамінною частиною пайплайну AutoML, що допомагає у наочному представленні даних і продуктивності моделі. Вона включає створення графічних представлень даних, від простих графіків до складних багатовимірних візуалізацій. За допомогою візуалізацій можна краще зрозуміти дані, інтерпретувати результати моделі та приймати обґрунтовані рішення.

Візуалізація даних охоплює:

– експлораторний аналіз даних (EDA) – використання візуалізацій для дослідження та розуміння структури даних;

– візуалізація ефективності моделі – графічне представлення показників продуктивності моделі;

– візуалізація важливості ознак – демонстрація значення різних ознак у прогнозних моделях;

– аналіз помилок – графічне відображення помилок моделі для ідентифікації шляхів вдосконалення.

Наступною фазою є моніторинг продуктивності. Моніторинг продуктивності є критичним аспектом, що забезпечує стабільність і точність роботи розгорнутих моделей протягом часу. Це включає постійне спостереження за продуктивністю моделі, виявлення будь-якого погіршення та ініціацію процесу повторного навчання чи налаштування, за потреби. Цей елемент є важливим для підтримки надійності та ефективності машинного навчання у виробничому середовищі.

Моніторинг продуктивності включає:

– метрики продуктивності моделі – використання таких метрик, як точність, специфічність, чутливість тощо;

– виявлення зміщення – ідентифікація змін у розподілі даних, які можуть впливати на ефективність моделі;

– збір відгуків – аналіз реакції на прогнози моделі для її поліпшення;

– сповіщення та звітність – налаштування систем сповіщення про зниження точності моделі або інших проблем.

Останньою фазою AutoML є деплоймент моделі. Фінальний етап деплоймента моделі означає її готовність до передбачень на нових даних, переходячи від стадії розробки до впровадження. Цей етап вимагає забезпечення доступності, масштабування та ефективної обробки даних в реальному часі або в пакетному режимі, відповідно до потреб застосування.

До процесу деплойменту належать такі аспекти:

– платформи для деплойменту – використання платформ та фреймворків для розгортання моделі, таких як хмарні сервіси чи локальні сервери;

– контроль версій – управління різними версіями моделей для забезпечення їх відтворення та відстежуваності;

– масштабування – адаптація рішення для обробки зростаючих обсягів даних та запитів;

– моніторинг та обслуговування – неперервний контроль за роботою розгорнутої моделі та здійснення необхідного технічного обслуговування для забезпечення її продуктивності.

У таблиці 1.1 наведено відносне значення та вплив кожного аспекту в AutoML пайплайні, а також ступінь можливої автоматизації цих компонентів. Зрозуміло, що кожен елемент відіграє ключову роль у забезпеченні успіху в процесі AutoML, а високий рівень автоматизації сприяє швидшому та більш ефективному розгортанню моделей машинного навчання.

Таблиця 1.1 – Відносне значення та вплив кожного аспекту в AutoML пайплайні

Складова	Важливість	Вплив на якість метрик	Рівень автоматизації
Обробка даних	Високий	Високий	Високий
Інженерія ознак	Високий	Високий	Високий
Вибір моделі	Високий	Високий	Високий
Тюнінг гіперпараметрів	Від середнього до високого	Значний	Високий
Візуалізація даних	Помірний	Помірний	Помірний
Моніторинг метрик	Високий	Високий	Високий

Кінець таблиці 1.1 – Відносне значення та вплив кожного аспекту в AutoML пайплайні

Деплоймент моделі	Високий	Відсутній	Високий

1.2 Огляд наукових публікацій у галузі хмарних систем машинного навчання для галузі біотеху

Проведемо аналіз поточних рішень у галузі розробки нових ліків. Зокрема у [1] автори зазначають, що машинне навчання може підвищити ефективність виявлення та прийняття рішень у складних процесах розробки ліків. Це дослідження підкреслює застосування ML на всіх етапах розробки ліків, включаючи валідацію біологічних мішеней, ідентифікацію прогностичних біомаркерів та аналіз цифрової патології. Однак вказується на виклики, пов'язані з інтерпретацією та повторюваністю результатів, що може обмежувати їхнє застосування.

У [2] описується дослідження ZairaChem, інструменту на базі штучного інтелекту та машинного навчання для моделювання QSAR/QSPR, який вимагає мінімальних обчислювальних ресурсів. Він показує, як обчислювальний профайлінг сполук перед синтезом та тестуванням може інформувати про просування провідних сполук.

Pharm-AutoML – автоматизований пакет машинного навчання для прогнозування клінічних результатів, описаний у [3] - це відкрите програмне забезпечення на Python, що дозволяє автоматизувати створення моделей ML та прогнозувати результати пізніх етапів розробки ліків, а саме – клінічних досліджень. Він спрощує кроки ML, такі як попередня обробка даних, налаштування моделей, аналіз результатів та інтерпретація моделей. Автори стверджують, що Pharm-AutoML перевершує інші фреймворки ML за точністю прогнозування, і використовує методи інтерпретації моделей для пояснення

налаштованих ML-пайплайнів користувачам. На даний момент існує обмеження на тип задач, для якого може бути використана програма, а саме задача мульти-класифікації.

Chemistry42 [4] є програмною платформою для де-ново дизайну та оптимізації малих молекул, інтегруючи техніки штучного інтелекту з методологіями обчислювальної та лікарської хімії. Вона генерує нові молекулярні структури з оптимізованими властивостями, перевіреними у in vitro та in vivo дослідженнях. Автори демонструють, як платформа може бути використана для пошуку нових молекулярних структур проти DDR1 та CDK20.

З огляду на проблему приватності і збереження конфіденційності, дослідження [5] аналізує використання хмарних обчислень у біотехнологічних лабораторіях для біоінформатичного застосунку. Особливу увагу приділено гібридним хмарам, які поєднують масштабованість публічних хмар з більшим контролем та індивідуальними налаштуваннями приватних хмар. Вони підкреслюють роль хмарного брокера як посередника між користувачами та публічними постачальниками.

У [6] автори обговорюють використання ML та DL у прогнозуванні нових малих молекулярних біоактивностей для деконволюції мішеней та оптимізації від хіту до ліду в дослідженнях з відкриття ліків. Вони розглядають сучасні оновлення в AI-інструментах як застосування хемоінформатики в медичній хімії для прийняття рішень, заснованих на даних, у відкритті ліків. Особливу увагу приділяється викликам, пов'язаним із забезпеченням якості даних у фармацевтичній промисловості, а також підвищенню ефективності малих молекул та їхніх властивостей.

1.3 Застосування пайплайну для тренування моделей машинного навчання на біологічних даних

Ключовим аспектом у сферах, як-от медична хімія, фармацевтика та біоінформатика, є конвертація молекулярних структур у формати, придатні для

обробки алгоритмами машинного навчання. Різноманітні методи представлення молекул мають свої специфічні переваги та обмеження. Основні методи включають: SMILES, графові представлення, відбитки та тривимірні структурні представлення.

SMILES є методом представлення структури молекули через короткий ASCII рядок.

Переваги:

- ефективність у плані компактності та зберігання;
- широке використання в хімінформатичних програмах.

Недоліки:

- не враховує деякі просторові аспекти молекули;
- можливість неоднозначності представлення.

У графовому представленні молекули відображаються як графи, де атоми – це вузли, а хімічні зв'язки – ребра.

Переваги:

- зберігають просторову конфігурацію молекул;
- підходять для застосування алгоритмів графових нейронних мереж.

Недоліки:

- підвищена складність обчислень;
- потребує більших ресурсів для зберігання та аналізу.

Відбитки (Fingerprints) – це вектори, що репрезентують наявність чи відсутність певних хімічних структур у молекулі.

Переваги:

- ефективність у плані компактності представлення;
- зручність при порівнянні молекул.

Недоліки:

- втрата інформації про тривимірну структуру молекул;
- ризик колізій при збігу відбитків різних молекул.

Тривимірні представлення задають точні координати атомів молекули у просторі.

Переваги:

– найвища точність у відображенні просторової структури молекул;

– ключове значення для аналізу молекулярних взаємодій.

Недоліки:

– велика обчислювальна складність;

– значний обсяг даних для зберігання.

На рисунку 1.1 зображено основні формати представлення молекул для машинного навчання.

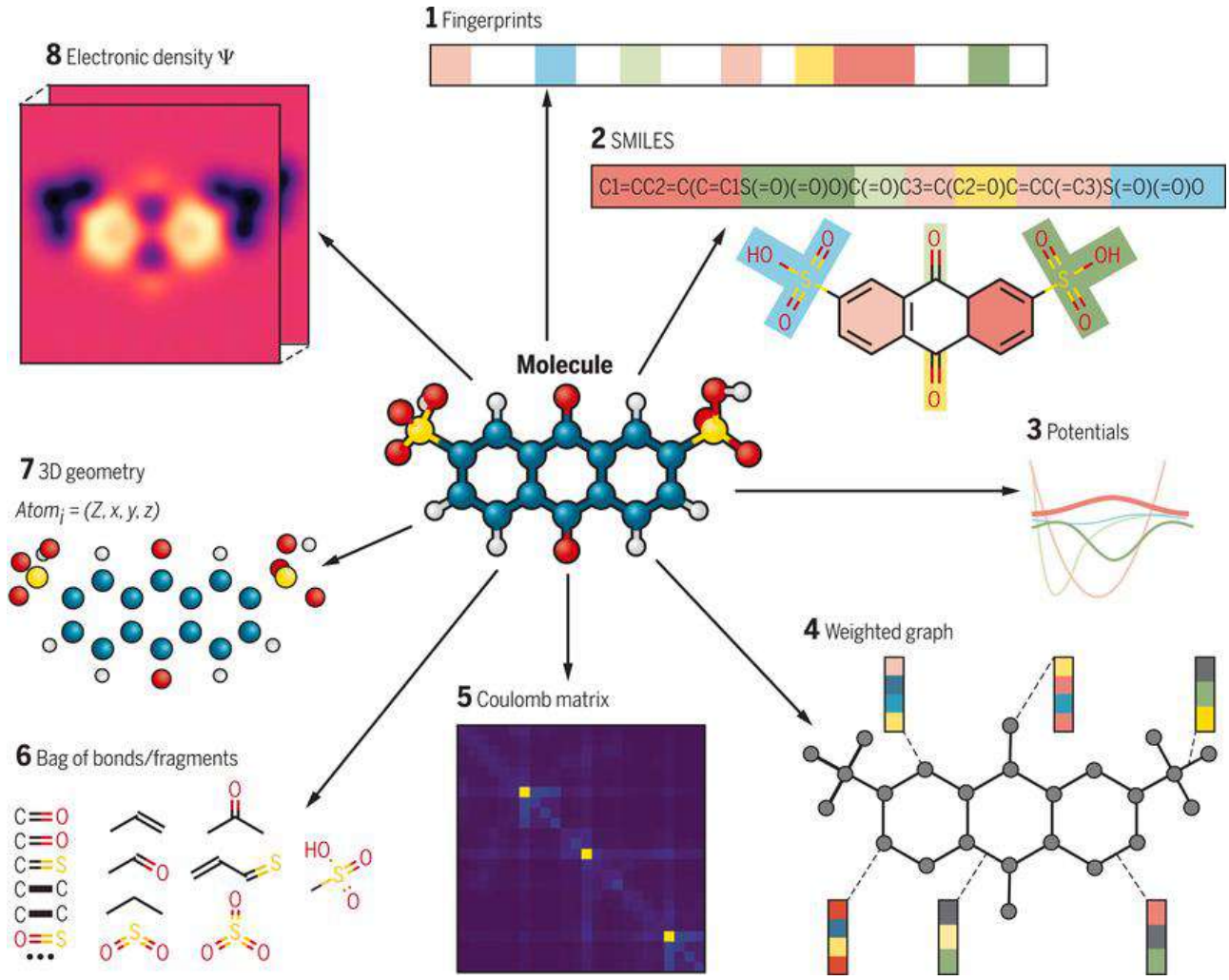


Рисунок 1.1 – Основні формати представлення молекул для машинного навчання [17]

У таблиці 1.2 наведено порівняння основних форматів представлення молекул.

Таблиця 1.2 – Порівняння основних форматів представлення молекул

Тип представлення	Опис	Переваги	Недоліки
SMILES	ASCII рядок, що описує структуру молекули.	Компактність, легкість зберігання, широка підтримка в хімінформатичних інструментах.	Втрата просторових характеристик, можлива неоднозначність.
Графові представлення	Молекули представлені як графи з атомами як вершинами і зв'язками як ребрами.	Збереження просторової структури, підтримка алгоритмами графових нейронних мереж.	Більша складність обчислень, вимоги до ресурсів.
Відбитки (Fingerprints)	Бінарні або числові вектори, що відображають характеристики хімічних структур.	Компактність, легкість порівняння молекул.	Втрата інформації про просторову структуру, ризик колізій.
3D Структурні представлення	Точні тривимірні координати атомів у молекулі.	Висока точність представлення, важливість для аналізу молекулярних взаємодій.	Велика обчислювальна складність, об'єм даних для зберігання.

Вибір методу представлення молекул у машинному навчанні опирається на цілі дослідження та вимоги до результатів.

Методи SMILES та відбитків ефективно застосовуються для швидкого аналізу та порівняння обширної кількості молекул, однак вони можуть ігнорувати деякі елементи просторової структури.

Графові представлення ідеально підходять для задач, де критично важливі характеристики зв'язків та структури молекул.

Тривимірні структурні представлення надають найвищу точність і є ключовими для аналізу просторових інтеракцій, наприклад, у вивченні зв'язування білків.

В остаточному підсумку, вибір підходу диктується специфікою завдань, наявністю даних та комп'ютерними ресурсами.

У сфері обробки молекулярних даних, AutoML пайплайни можуть суттєво полегшити та прискорити дослідження у відкритті нових медичних препаратів, прогнозуванні характеристик молекул та інших біоінформатичних дослідженнях. Наприклад:

- прогнозування біологічної активності;
- автоматизація ідентифікації молекул, потенційно активних проти певних біологічних цілей;
- оптимізація лікарських засобів;
- дослідження та прогнозування характеристик фармацевтичних продуктів, таких як розчинність, токсичність, біодоступність;
- дослідження молекулярних інтеракцій;
- розробка моделей для аналізу взаємодій між різними молекулами та білками.

1.4 Постановка задачі та вибір технологій для реалізації

У даній роботі поставлено завдання створення всебічної автоматизованої системи для тренування та моніторингу моделей машинного навчання у рамках

SAAS платформи для розробки ліків in silico. Головною метою є розробка пайплайну, призначеного для роботи з біологічними даними, який би забезпечував гнучкість, модульність, легке налаштування, паралельну обробку даних та здатність до швидкого масштабування та гнучкого деплоювання у двох режимах: On-Cloud та On-Premise. Цей пайплайн має бути інтегрований як сервіс у існуючу SAAS платформу, яка спеціалізується на in silico розробці ліків.

Використання Kubernetes як інструменту для оркестрації контейнерів надає високу доступність, масштабованість та ефективне управління ресурсами. Kubernetes виглядає ідеальним варіантом для On-Cloud деплойменту, забезпечуючи підтримку більшістю хмарних провайдерів, таких як AWS та GCP, з їх власними сервісами для Kubernetes. Це рішення пропонує гнучкість, потужність та забезпечує простоту масштабування.

Kubeflow надає можливість автоматизації процесів тренування, оцінки та розгортання моделей машинного навчання. AutoML пайплайн має включати етапи попередньої обробки даних, інженерії ознак, вибору моделі та налаштування гіперпараметрів.

Використання відбитків (Fingerprints) для представлення молекул у поєднанні з класичними моделями машинного навчання є вдалим вибором для AutoML пайплайну з урахуванням типових розмірів навчальних вибірок та автоматизації вибору алгоритмів. Для генерації відбитків і їх подальшу конвертацію у признаки для моделей найкраще підійде Python та бібліотека scikit-learn.

Для обробки біологічних даних буде застосована спеціалізована бібліотека Python - RDKit

Для контейнеризації всіх процесів найкращим вибором є Docker. Він надасть наступні переваги:

− стандартизація додатків та їх залежностей у контейнери, що спрощує деплоймент і масштабування;

– ізоляція додатків, гарантуючи їх стабільну роботу в різних середовищах, що є ключовим для відтворюваності наукових експериментів та машинного навчання.

– контейнери можуть легко бути оркестровані через Kubernetes, забезпечуючи автоматизоване управління додатками.

– уніфікація робочих процесів, мінімізуючи ризики проблем "це працює на моєму комп'ютері".

– ефективність розділення та ізоляції ресурсів, надаючи високий рівень безпеки для обробки конфіденційних біологічних даних

1.5 Висновки

У даному розділі було проведено детальний огляд ключових складових, необхідних для AutoML пайплайну. Проаналізовано та порівняно різноманітні технології, що можуть бути використані для реалізації кожного елемента пайплайну. Окрім того, розглянуто різні методи представлення біологічних даних, придатних для використання у машинному навчанні. Розділ завершується формулюванням завдання та вибором відповідних технологій, що обґрунтовано на основі проведеного аналізу.

2 МАТЕМАТИЧНА МОДЕЛЬ ОСНОВНИХ КОМПОНЕНТ ПАЙПЛАЙНУ ДЛЯ АВТОМАТИЧНОГО ТРЕНУВАННЯ МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ

2.1 Математична модель алгоритму вибору найкращих ознак

Процес визначення та відбору ключових ознак з датасету у машинному навчанні є важливим кроком, що дозволяє ідентифікувати найбільш значущі ознаки для задачі. Існує декілька основних методів відбору ознак, кожен з яких характеризується своїми особливостями, перевагами та обмеженнями.

2.1.1 Фільтраційні методи

Фільтраційні методи можна описати як відбір ознак на основі статистичних показників. Переваги: простота в обчисленнях; не потребує тренування моделей. Обмеження: не враховує взаємодію між ознаками.

Приклади фільтраційних методів:

- кореляція Пірсона – оцінка лінійної залежності між двома змінними;
- хі-квадрат (chi-squared) тест – аналіз незалежності між категоріальними змінними;
- взаємна інформація (mutual information) – міра взаємної залежності між змінними, вказує на кількість інформації, яку одна змінна містить про іншу;
- аналіз головних компонент (PCA) – зменшення розмірності даних зі збереженням максимальної варіативності;
- ANOVA F-тест – визначення статистично значущих різниць між середніми значеннями двох або більше груп.

Фільтраційні методи слугують сильним засобом для початкового аналізу даних та вибору ознак. Їх перевагою є зниження розмірності даних при збереженні важливої інформації. Проте, оскільки ці методи не враховують взаємодію між ознаками та алгоритмом машинного навчання, їх застосовують у комбінації з іншими для досягнення найкращого ефекту.

2.1.2 Обгорткові методи

Ці методи передбачають використання моделей машинного навчання для оцінки значимості ознак. Суть полягає в тому, що алгоритм навчання використовується як чорна скринька для оцінки ефективності підмножин ознак у вирішенні конкретної задачі.

Математичне формулювання: як приклад – випадковий ліс, де важливість ознаки A може бути оцінена через зниження точності моделі при видаленні цієї ознаки. Важливість ознаки визначається як різниця між точністю моделі з усіма ознаками та точністю моделі без даної ознаки.

Ці методи враховують взаємодію між ознаками, що часто призводить до кращих результатів у порівнянні з фільтраційними методами. Вони можуть виявити найбільш значущі ознаки для конкретної моделі.

Основним недоліком є висока обчислювальна складність, оскільки потрібно оцінювати багато різних комбінацій ознак. Це може призвести до ризику перенавчання, особливо при великій кількості ознак.

2.1.3 Вбудовані методи

Вбудовані методи інтегрують процес відбору ознак безпосередньо в алгоритм навчання моделі. Це означає, що відбір ознак і тренування моделі відбуваються одночасно, і важливість ознак оцінюється в рамках процесу навчання.

Математичне формулювання: прикладом може слугувати регуляризація LASSO, де цільова функція моделі штрафує за величину коефіцієнтів: це допомагає зменшити кількість ознак, включених до моделі, залишаючи лише найважливіші.

Вбудовані методи дозволяють ефективно визначити важливі ознаки, одночасно знижуючи шанс перенавчання. Вони можуть бути більш ефективними з точки зору обчислень, ніж обгорткові методи.

Вбудовані методи залежать від вибору моделі, що може обмежити їх застосування. Крім того, вони можуть бути складними у налаштуванні, оскільки потребують тонкої регуляризації параметрів.

2.1.4 Висновки

На рисунку 2.1 зображено методи вибору найкращих ознак. Кожна стратегія вибору ознак має свої специфічні переваги та сфери використання. Фільтраційні підходи характеризуються простотою та оперативністю, але можуть ігнорувати значущі взаємодії між ознаками. Обгорткові стратегії гарантують більш точне відбирання ознак, однак потребують великих обчислювальних витрат. Вбудовані підходи включають процес відбору ознак безпосередньо у процедуру навчання, надаючи гармонійний метод. Селекція певної стратегії визначається конкретною задачею, наявністю даних та комп'ютерними ресурсами.

2.2 Математичне формулювання алгоритмів машинного навчання

Алгоритми машинного навчання можуть бути представлені через три фундаментальні елементи: функцію апроксимації, критерій якості та метод оптимізації.

Функція апроксимації, іноді звана моделлю, є математичною структурою, яка прагне відтворити залежності між вхідними та вихідними даними у навчальному наборі. Математично це можна виразити як:

$$\hat{y} = f(x; \theta), \tag{2.1}$$

де $\hat{y}$ – прогнозоване вихідне значення;

x – вектор вхідних атрибутів;

θ – параметри моделі (наприклад, ваги у нейронній мережі);

f – сама функція апроксимації.

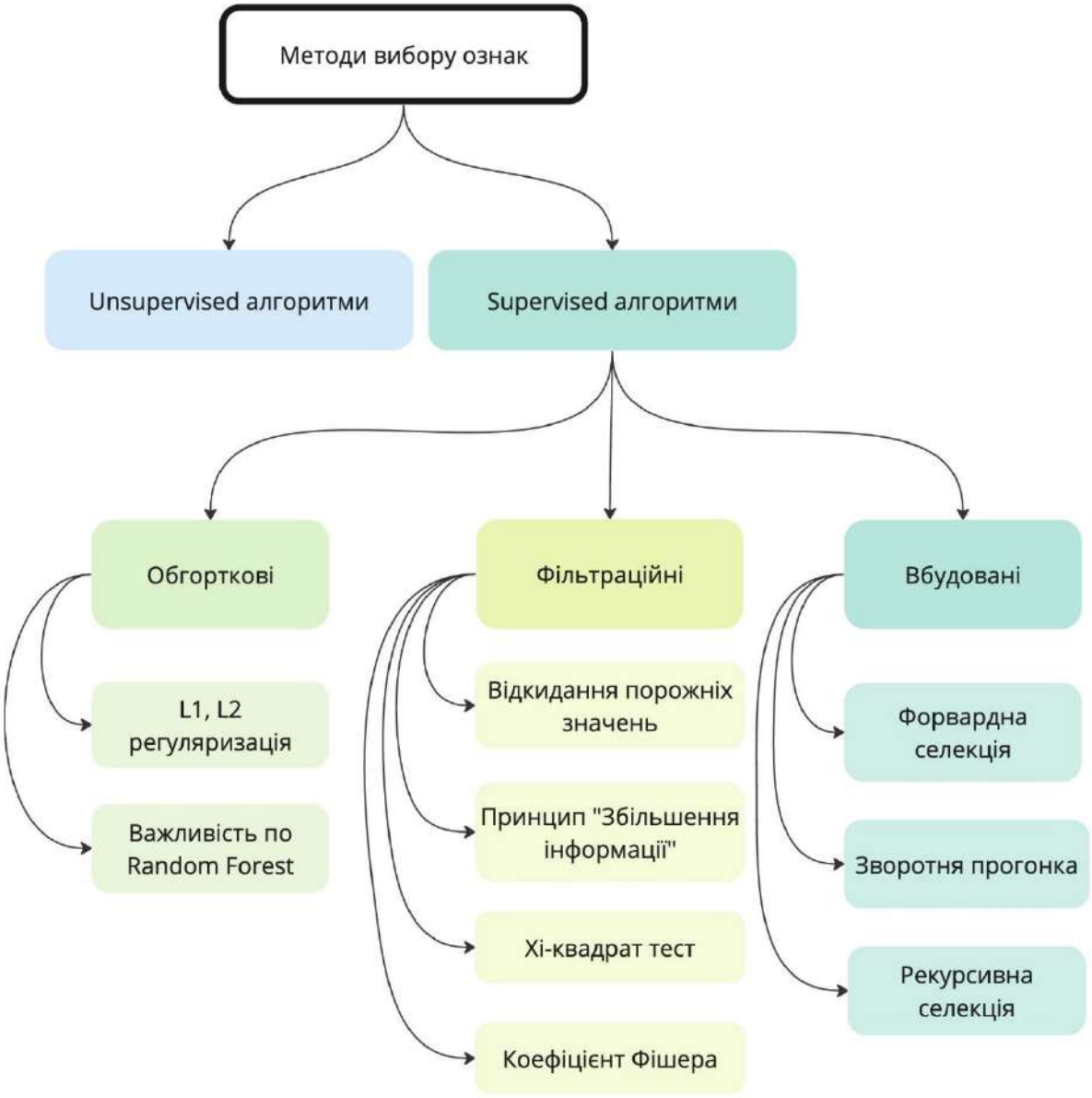


Рисунок 2.1 – Методи вибору найкращих ознак

Критерій якості, також званий функцією втрат, оцінює якість моделі у відтворенні реальних даних. Він вимірює розбіжність між прогнозованими величинами $\hat{y}$ та реальними величинами y. Ціль навчання полягає в мінімізації цього критерію. Наприклад, у випадку регресії часто застосовують середньоквадратичну помилку (MSE):

$$L(\theta) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2\,, \tag{2.2}$$

де $L(\theta)$ – значення критерію якості;

n – кількість спостережень у навчальному наборі;

$\hat{y}_i$ – прогнозоване значення для і-го спостереження;

y_i – реальне значення для і-го спостереження.

Метод оптимізації використовується для знаходження оптимальних значень параметрів моделі θ, які мінімізують критерій якості. Досягнення цього можливе за допомогою різноманітних технік, включаючи градієнтний спуск, стохастичний градієнтний спуск (SGD) та інші методи. Наприклад, у стохастичному градієнтному спуску параметри моделі оновлюються на кожному кроці за допомогою наступної формули:

$$\theta_{t+1} = \theta_t - a\nabla_\theta L(\theta_t), \tag{2.3}$$

де θ_t – значення параметрів на кроці t;

a – швидкість навчання;

$\nabla_\theta L(\theta_t)$ – градієнт критерію якості за параметрами моделі на кроці t.

Загалом, математична модель алгоритму машинного навчання включає взаємодію між функцією апроксимації, критерієм якості та методом оптимізації. Ці елементи разом визначають процес навчання моделі з даних та її адаптацію до розв'язання конкретної задачі. Вибір певних формулювань для кожного з цих елементів залежить від задачі, типу даних та цілей дослідника або інженера. Нижче наведено математичні описи для деяких з найпопулярніших алгоритмів машинного навчання.

2.2.1 Лінійна регресія

Застосування: лінійна регресія застосовується для прогнозування значень неперервних змінних, таких як вартість нерухомості або обсяг продажів.

Концепція: цей метод прагне встановити лінійну залежність між незалежними та залежними змінними, представляючи їх у вигляді прямої лінії.

Механізм дії: лінійна регресія полягає у мінімізації суми квадратів різниць між фактичними значеннями та тими, що прогнозуються моделлю.

Плюси: модель проста у розумінні та швидка в обчисленнях.

Мінуси: обмежена лінійними зв'язками та чутлива до викидів у даних.

Чутливість до викидів: висока, оскільки вони можуть суттєво вплинути на параметри регресії.

Обмеження: не придатна для аналізу не лінійних залежностей.

Швидкість обчислень: висока завдяки простоті моделі.

Математична модель: модель описується рівнянням лінії. Цільова функція полягає у мінімізації суми квадратів відхилень між фактичними та передбаченими значеннями.

2.2.2 Логістична регресія

Застосування: логістична регресія використовується для бінарної класифікації, наприклад, для оцінки імовірності події або для прогнозування відтоку клієнтів.

Концепція: метод оцінює ймовірність приналежності спостереження до одного з двох класів.

Механізм дії: логістична регресія мінімізує логарифмічну функцію втрат, яка вимірює розбіжність між прогнозованими ймовірностями та фактичними мітками класів.

Плюси: модель здатна надавати ймовірності приналежності до класів та легко інтерпретується.

Мінуси: припускає лінійний розподіл даних та обмежена лише бінарною класифікацією.

Чутливість до викидів: середня, оскільки логістична регресія менш чутлива до викидів, ніж лінійна регресія.

Обмеження: непридатна для аналізу нелінійних відносин і обмежена двома класами.

Швидкість обчислень: висока завдяки ефективним алгоритмам оптимізації.

Математична модель: модель використовує логістичну функцію для перетворення лінійної комбінації вхідних змінних у ймовірності. Цільова функція полягає у мінімізації логарифмічної втрати, яка вимірює відхилення прогнозованих ймовірностей від фактичних міток класів.

2.2.3 Дерево рішень

Застосування: дерево рішень використовується для класифікації та регресії, наприклад, для ідентифікації видів рослин або для оцінки вартості житла.

Концепт: дерево рішень приймає рішення, просуваючись від кореня до листя, вибираючи напрямки на основі атрибутів.

Механізм: дерево рішень використовує рекурсивний поділ даних, застосовуючи критерії поділу на кожному етапі.

Переваги: простота у розумінні та здатність моделювати не лінійні зв'язки.

Недоліки: схильність до перенавчання та чутливість до змін у даних.

Чутливість до викидів: висока, оскільки викиди можуть сильно вплинути на структуру дерева.

Обмеження: може утворювати складні дерева, які важко інтерпретувати.

Швидкість: залежить від глибини дерева та кількості атрибутів.

Математична модель: рекурсивний поділ простору атрибутів на підпростори відповідно до критерію оптимального поділу (наприклад, індекс Джині або ентропія).

Цільова функція: мінімізація критерію неоднорідності в кожному листку дерева.

2.2.4 Випадковий ліс

Застосування: випадковий ліс застосовується для класифікації та регресії у різних областях, включаючи медицину, фінанси та екологію.

Концепт: випадковий ліс є ансамблем дерев рішень, кожне з яких навчається на випадковій підмножині даних, а їхні прогнози комбінуються.

Механізм: кожне дерево в ансамблі навчається на випадковій підмножині атрибутів та прикладів, а потім усі дерева "голосують" за кінцевий прогноз.

Переваги: зниження ризику перенавчання порівняно з одиночним деревом рішень, висока точність прогнозування.

Недоліки: великі вимоги до обчислювальних ресурсів, складність у інтерпретації.

Чутливість до викидів: низька, оскільки ансамбль дерев стійкий до окремих викидів.

Обмеження: може бути неефективним при великій кількості шуму в даних.

Швидкість: залежить від кількості дерев в ансамблі та глибини кожного дерева.

Математична модель: ансамбль дерев рішень, кожне з яких навчається на випадковій підмножині даних та атрибутів.

Цільова функція: покращення точності за рахунок комбінування прогнозів багатьох дерев.

2.2.5 Одношаровий перцептрон

Застосування: використовується для бінарної класифікації та розпізнавання простих образів.

Концепція: це базова форма нейронної мережі, яка вирішує, чи слід активувати вихід на основі вагової суми входів.

Механізм: нейрон активується, коли сума вагових входів перевищує заданий поріг.

Переваги: простота у реалізації та навчанні.

Недоліки: обмежена здатність розв'язувати лише лінійно роздільні завдання.

Чутливість до викидів: висока, оскільки викиди можуть сильно вплинути на рішення перцептрона.

Обмеження: непридатний для розв'язання завдань з не лінійною роздільністю.

Швидкість: висока через простоту моделі.

Математична модель: модель базується на лінійній комбінації вхідних сигналів та їх ваг.

Цільова функція: мінімізація кількості помилок класифікації.

2.2.6 Багатошаровий перцептрон (MLP)

Застосування: застосовується у широкому спектрі завдань, включаючи класифікацію, регресію, розпізнавання образів та обробку природної мови.

Концепція: багатошаровий перцептрон складається з одного або декількох прихованих шарів, що дозволяє моделювати складні не лінійні зв'язки між даними.

Механізм: кожен нейрон у мережі формує ваговану суму своїх входів, яка потім проходить через не лінійну активаційну функцію.

Переваги: здатність до моделювання складних не лінійних залежностей.

Недоліки: схильність до перенавчання, великі обчислювальні вимоги.

Чутливість до викидів: залежить від конкретної архітектури мережі та застосованих методів регуляризації.

Обмеження: потребує ретельного налаштування гіперпараметрів та тривалого навчання.

Швидкість: залежить від кількості шарів та нейронів у мережі.

Математична модель: модель представляє собою ансамбль нейронів, організованих у шари, кожен з яких виконує певні обчислення.

Цільова функція: мінімізація сумарної функції втрат, яка оцінює різницю між прогнозованими та фактичними величинами, з можливим застосуванням регуляризації.

2.2.7 Графова нейронна мережа

Для формулювання математичної моделі графових нейронних мереж, потрібно сформувате визначення “графу” і його властивостей.

Почнемо з графа, який не має зв'язків (ребер), а складається лише з множини вершин υ. Припустимо, що $x_i \in R^k$ – представляє ознаки вершини і. Об'єднання цих ознак у матрицю дає нам матрицю ознак вершин розміром $n\ x\ k$:

$$X = (x_1, \dots, x_n)^T. \tag{2.4}$$

Зазначимо, що зміна нумерації вершин призведе до зміни матриці ознак, тому вихід нейронної мережі повинен бути незалежним від порядку вершин. Для цього введимо поняття інваріантності та еквіваріантності функцій щодо перестановок. Функція $f(X)$ є інварінтною до перестановок, якщо для усіх матриць перестановок P, виконується:

$$f(PX) = f(X). \tag{2.5}$$

Нижче наведено приклад такої моделі:

$$f(X) = \phi\left(\sum_{i\epsilon\upsilon} \psi(x_i)\right), \tag{2.6}$$

де ψ та ϕ – це функції, які оптимізуються градієнтним спуском, сума може бути заміненою на функцію мінімуму, максимуму тощо.

Введемо поняття еквіваріантності функції $f(X)$ відносно перестановок, якщо для всіх матриць перестановок P виконуєтсья рівність:

$$f(PX) = Pf(X). \tag{2.7}$$

Для побудови еквіваріантної функції, яка оброблятиме ознаки кожної вершини та видаватиме нові ознаки, незалежно від порядку вершин у графі, застосовується наступний підхід:

$$h_i = \psi(x_i), \tag{2.8}$$

де x_i – вхідні признаки;

h_i – вихідні нові признаки;

ψ – еквіваріантна функція.

Також можна визначити інваріантну функцію, яка обробляє еквіваріантні функції та повертає вектор чисел, що характеризує увесь граф:

$$f(X) = \phi\left(\sum_{i \epsilon \upsilon} \psi(x_i)\right), \tag{2.9}$$

де ψ – еквіваріантна функція;

ϕ – інваріантна функція.

Розглянемо граф, що складається з множини вершин, а також граней між ними, де $E \subseteq VxV$. Ці грані можна представити, використовуючи матрицю суміжності:

$$a_{ij} = \begin{cases} 1, (i,j)\ \epsilon \text{ E}; \\ 0, \text{інакше}. \end{cases} \tag{2.10}$$

Властивості еквіваріантності, а також інваріантності функцій, зберігаються, але в цьому випадку для граней вони набудуть іншої форми.

Інваріантність:

$$f(PX, PAP^T) = Pf(X, A), \tag{2.11}$$

де P – матриця перестановок;

A – матриця суміжності;

X – матриця признаків.

Важливо враховувати оточення кожної ноди – її сусідів. Для вершини i сусіди на 1-му рівні визначаються як:

$$N_i = \{j: (i,j) \in E \ \vee (j,i) \in E\}. \tag{2.12}$$

З цього отримуємо можливість ввести матрицю ознак усіх суміжних до даної вершин:

$$X_{N_i} = \left\{\{x_j: j \in N_i\}\right\}. \tag{2.13}$$

З отриманого, визначимо еквіваріантну функцію до перестановок $f(X, A)$, через функцію g для усіх суміжних нод:

$$f(X, A) = \begin{bmatrix} - g(x_1, X_{N_1}) - \\ - g(x_2, X_{N_2}) - \\ \vdots \\ - g(x_n, X_{N_n}) - \end{bmatrix}, \tag{2.14}$$

де X_{N_i} – матриця ознак суміжних вершин.

Для того, щоб функція f була еквіваріантною, функція g має бути незалежною до послідовності нод у X_{N_i}, тож g повинна мати властивість інваріантності до перестановок.

Рисунок 2.2 демонструє призначення функції g. Вона модифікує ознаки вершини з урахуванням її сусідів. Це схоже на роботу одного шару будь-якої графової нейронної мережі. Використовуючи функцію f, можна розв'язувати безліч задач для графів, включаючи класифікацію вершин, класифікацію графів, прогнозування існування ребер тощо.

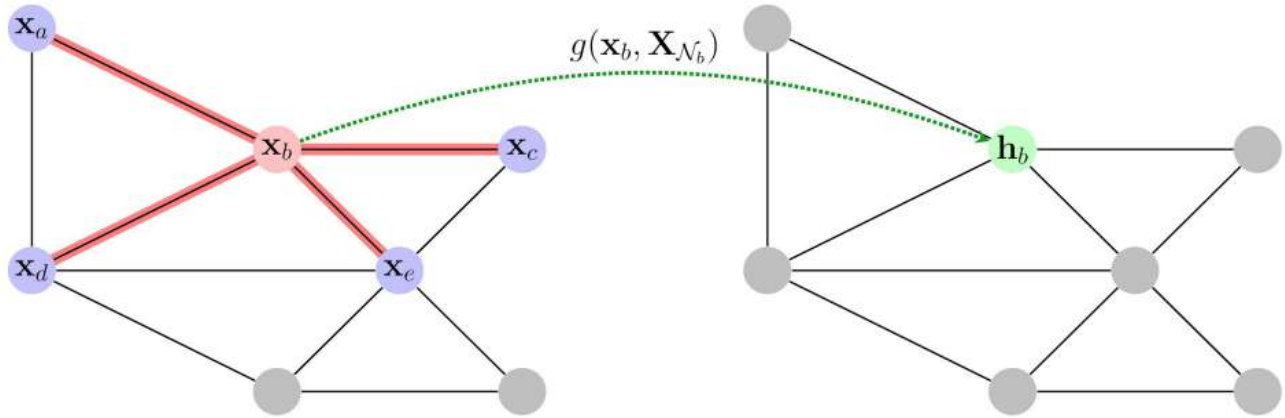


Рисунок 2.2 – Візуалізація роботи функції g

Існує три основних підходи формалізації функції g:

- графові згорткові нейронні мережі (GCN);
- уважні нейронні мережі (GAT);
- нейронні мережі з передачею повідомлень.

Розглянемо ці підходи детальніше.

На рисунку 2.3 зображено формульну візуалізацію цих трьох проблем, пов'язаних із графами.

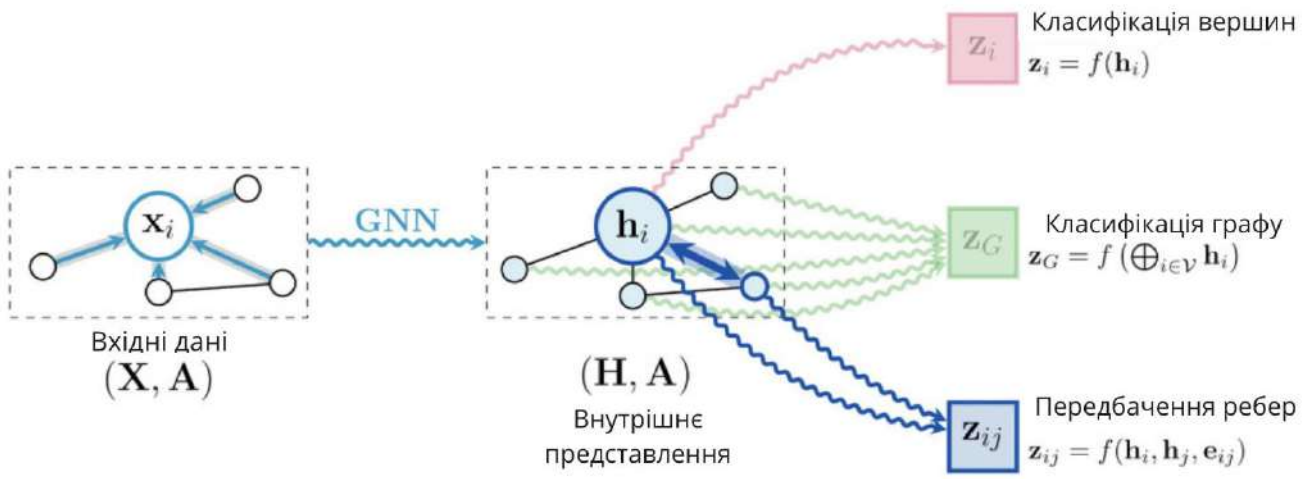


Рисунок 2.3 – Різновидності задач із графами

У графових згорткових нейронних мережах ознаки сусідів вершин агрегуються за допомогою константних ваг c_{ij}:

$$h_i = \phi\left(x_i, \sum_{i \epsilon N_i} c_{ij}\psi(x_j)\right), \tag{2.15}$$

де ϕ – інваріантна функція;

ψ – еквіваріантна функція;

N_i – сусіди вершини i 1-ого рівня.

У стандартній реалізації ці ваги прораховуються, враховуючи весь граф. Цей процес відображає рисунок 2.4.

Ці архітектури доцільно використовувати для гомофілійних графів, де справджується принцип, що подібні вершини з великою ймовірністю з'єднуються. Однією з основних переваг цього підходу є висока швидкість обчислень, оскільки ваги обчислюються один раз і залишаються константними. Однією з найпоширеніших архітектур є GCN. Зміна ознак кожної вершини відбувається за формулою:

$$h_i = \Theta^T \sum_{j \epsilon N_i} \frac{e_{j,i}}{\sqrt{\widehat{d}_j \widehat{d}_i}} x_j, \tag{2.16}$$

де Θ^T – функція, яка навчається;

$\widehat{d}_i = 1 + \sum_{j \epsilon N_i} e_{j,i}$ – зважений степінь вершини, де $e_{j,i}$ означає вагу серцевої вершини j до таргет-вершини i (1 за замовчуванням).

Однією із найпоширеніших архітектур цього підходу є власне GCN, у назві якої якраз є слово convolutional, тобто згортковий. Ця нейронна мережа також є однією із перших у сфері побудови графових нейронних мереж.

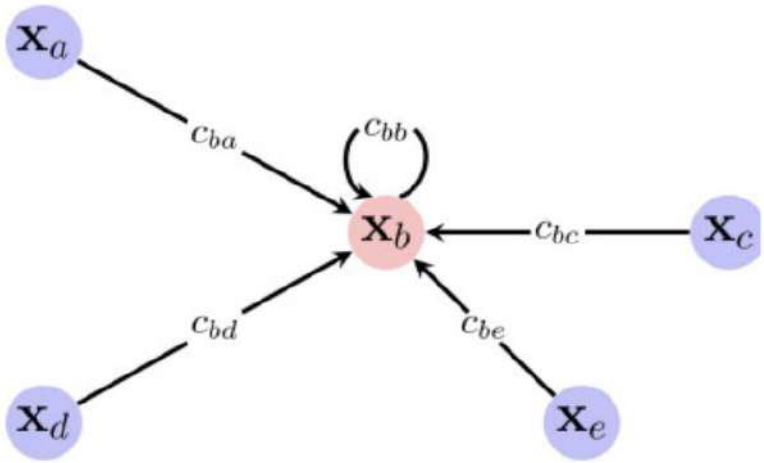


Рисунок 2.4 – Візуалізація згорткового підходу

Для уважних нейронних мереж признаки для сусідів вершин агрегуються за допомогою неявних ваг $a_{ij} = a(x_i, x_j)$:

$$h_i = \phi\left(x_i, \sum_{i \epsilon N_i} a(x_i, x_j)\psi(x_j)\right), \tag{2.17}$$

де ϕ – інваріантна функція;

ψ – еквіваріантна функція;

N_i – сусіди вершини i 1-ого рівня.

На відміну від графових згорткових нейронних мереж, такі мережі вже використовують признаки вершин, проте все ще в неявному вигляді. На рисунку 2.5 зображено візуалізацію такого підходу.

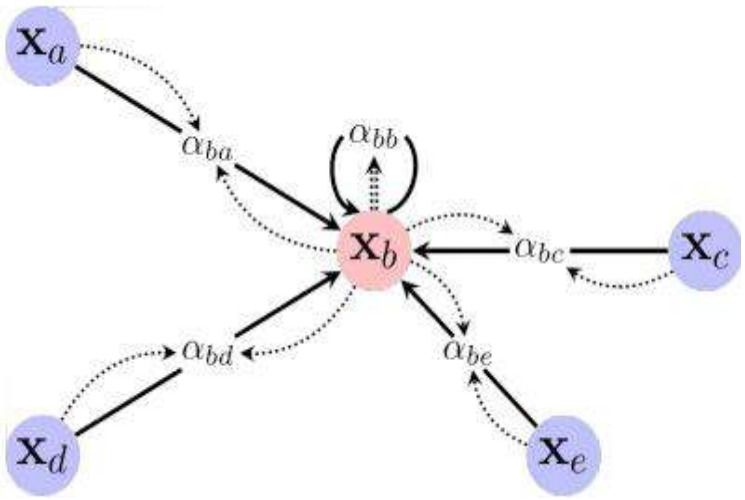


Рисунок 2.5 – Візуалізація уважного підходу

Такі архітектури доцільно використовувати, якщо ребра не обов'язково мають представляти гомофілію, тобто подібність вершин. Також цей підхід все одно підраховує скалярні величини для кожного ребра, що позитивно впливає на обчислювальну швидкість.

Для графових нейронних мереж із передачею повідомлень фічі для сусідів вершин агрегуються за допомогою "повідомлень" $m_{ij} = \psi(x_i, x_j)$:

$$h_i = \phi\left(x_i, \sum_{i \epsilon N_i} \psi(x_i, x_j)\right), \tag{2.18}$$

де ϕ – інваріантна функція;

ψ – еквіваріантна функція;

N_i – сусіди вершини i 1-ого рівня.

Цей підхід використовує всю інформацію про таргет-вершину та вершину-сусіда й динамічно обчислює необхідні ваги. На рисунку 2.6 зображено візуалізацію такого підходу.

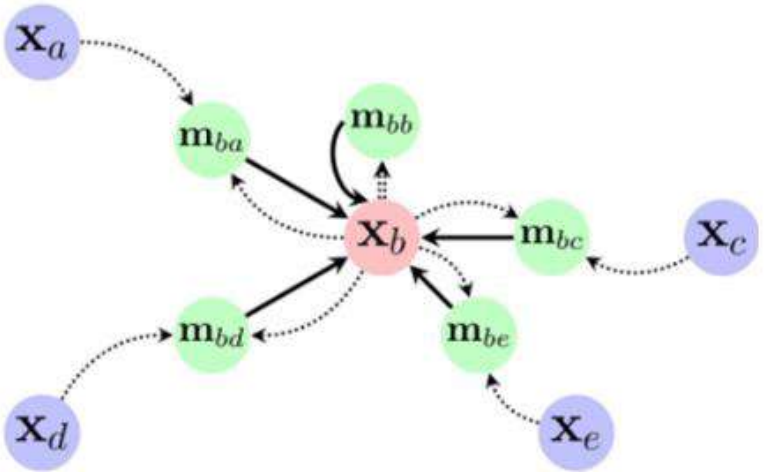


Рисунок 2.6 – Візуалізація підходу із передачею повідомлень

Такі архітектури можуть відчувати проблеми із швидкістю тренування, оскільки динамічно обчислюють інформацію про кожне ребро. Проте саме такі архітектури ідеально використовувати для біолого-хімічних задач (саме така, як у нас), а також для причинних (reasoning) задач та задач симуляції.

Найпопулярнішими й найбільш вживаними архітектурами є GIN та GINE.

У GIN зміна фіч кожної вершини відбувається за наступною формулою:

$$h_i = \theta\left(x_i + \sum_{j \epsilon N_i} x_j\right), \tag{2.19}$$

де θ – нейронна мережа, наприклад, MLP – багатошаровий перцептрон;

N_i – сусіди вершини i 1-ого рівня.

У GINE зміна признаків кожної вершини відбувається за наступною формулою:

$$h_i = \theta\left(x_i + \sum_{j \in N_i} ReLU\left(x_j + e_{j,i}\right)\right), \quad (2.20)$$

де θ – нейронна мережа, наприклад, MLP – багатошаровий перцептрон;

N_i – сусіди вершини i 1-ого рівня;

$e_{j,i}$ – це ребреві фічі конкретного зв'язку між вершинами;

$ReLU$ – функція активації.

2.2.8 Техніка машин опорних векторів (SVM)

Область застосування: використовується для задач класифікації, регресії та визначення викидів.

Опис методу: SVM шукає оптимальну гіперплощину, яка найефективніше розділяє дані на дві категорії.

Методика роботи: спрямована на максимізацію маржі між гіперплощиною та найближчими до неї точками з кожного класу, відомими як опорні вектори.

Переваги: відома своєю високою точністю та здатністю узагальнення на нові дані.

Недоліки: вимагає значних обчислювальних ресурсів і може бути складною у налаштуванні параметрів.

Стійкість до аномалій: має середню чутливість до викидів.

Обмеження: не є оптимальним рішенням для аналізу великих наборів даних через обчислювальну складність.

Продуктивність: залежить від вибору ядра та обсягу даних.

Математична модель: використовує оптимізацію для знаходження гіперплощини, яка найкраще розділяє класи.

Цільова функція: мінімізація відстані між опорними векторами та гіперплощиною.

2.2.9 Підсилення градієнтного бустингу (XGBoost)

Область застосування: використовується для класифікації, регресії, ранжування та виявлення аномалій.

Опис методу: XGBoost є удосконаленою версією градієнтного бустингу, спрямованою на оптимізацію швидкості та точності.

Методика роботи: послідовне побудова дерев, де кожне наступне дерево коригує помилки попередніх.

Переваги: відзначається високою швидкістю та ефективністю, має гнучкість у налаштуванні.

Недоліки: вимагає ретельного підбору параметрів і схильний до перенавчання при неправильному налаштуванні.

Стійкість до аномалій: має низьку чутливість до викидів.

Обмеження: може бути менш ефективним для дуже маленьких датасетів.

Продуктивність: висока завдяки ефективним алгоритмам оптимізації.

Математичний підхід: комбінує декілька дерев рішень, де кожне нове дерево фокусується на виправленні помилок попередніх.

Цільова функція: мінімізація сукупної функції втрат з урахуванням регуляризації для контролю складності моделі.

2.3 Математичний опис процесу баєсівської оптимізації для налаштування гіперпараметрів

Баєсівська оптимізація є високоефективним методом для ідентифікації оптимальних гіперпараметрів у моделях машинного навчання. Використовуючи принципи баєсівської теорії, цей метод дозволяє апроксимувати невідому цільову функцію і здійснювати цілеспрямований пошук у просторі гіперпараметрів. Цей підхід особливо корисний, коли змінні оптимізації є дискретними або категоріальними і коли цільова функція не є диференційованою за цими змінними.

Розглянемо $f(x)$ як невідому цільову функцію, де x – це вектор гіперпараметрів. Метою баєсівської оптимізації є знаходження такого x^*, що мінімізує значення $f(x)$.

Побудова апроксимації: використання гаусівського процесу (GP) для апроксимації невідомої функції $f(x)$, де GP задається наступним чином:

$$f(x) \sim GP\big(m(x), k(x, x')\big), \tag{2.21}$$

де $m(x)$ – це функція середнього значення;
$k(x, x')$ – коваріаційна функція, що визначає зв'язок між різними точками у просторі гіперпараметрів.

Визначення функції виразності: використання спеціальної функції, такої як очікуване поліпшення (EI), верхня границя довіри (UCB) або ймовірність поліпшення (PI), для визначення найбільш перспективних точок для подальшого пошуку.

Оптимізація функції виразності: вибір наступної точки для оцінки шляхом максимізації функції виразності.

Оновлення моделі: після оцінки апроксимація гаусівського процесу оновлюється з урахуванням нових даних, і цикл повторюється.

Завдяки цим крокам, баєсівська оптимізація забезпечує поступове наближення до оптимальних значень гіперпараметрів, враховуючи інформацію про попередні оцінки цільової функції. Це дозволяє ефективно керувати балансом між дослідженням нових областей простору гіперпараметрів і використанням наявної інформації для поліпшення точності моделі.

У практичному застосуванні для реалізації баєсівської оптимізації часто використовується фреймворк Optuna, який надає зручні інструменти для автоматизованого налаштування гіперпараметрів з використанням різних методів оптимізації, включаючи баєсівську оптимізацію з використанням гаусівських процесів. Optuna дозволяє користувачам визначити простір пошуку гіперпараметрів, функцію оцінки продуктивності моделі та критерії зупинки

процесу оптимізації. Під час роботи Optuna автоматично вибирає найбільш обнадійливі комбінації гіперпараметрів для оцінки, використовуючи апроксимацію гаусівського процесу та функцію виразності для керування процесом пошуку. Це дозволяє ефективно знаходити оптимальні значення гіперпараметрів, покращуючи продуктивність моделі машинного навчання з мінімальними обчислювальними витратами.

2.4 Метрики оцінювання моделей машинного навчання

Основними метриками для оцінки задач бінарної класифікації є $accuracy$, $recall$, $precision$ та $F1$.

Формульно $accuracy$ можна подати так:

$$accuracy = \frac{TP + TN}{P + N}, \tag{2.22}$$

де TP – кількість правильно передбачених семплів класу 1;

TN – кількість правильно передбачених семплів класу 0;

P – кількість реальних семплів класу 1;

N – кількість реальних семплів класу 0.

Формульно $recall$ можна подати так:

$$recall = \frac{TP}{P}, \tag{2.23}$$

де TP – кількість правильно передбачених семплів класу 1;

P – кількість реальних семплів класу 1.

Формульно $precision$ можна подати так:

$$precision = \frac{TP}{PP}, \tag{2.24}$$

де TP – кількість правильно передбачених семплів класу 1;

PP – кількість передбачених семплів класу 1.

Формульно $F1$ можна подати так:

$$F1 = \frac{2 * recall * precision}{recall + precision} = \frac{2 * TP}{2 * TP + FP + FN}, \quad (2.25)$$

де TP – кількість правильно передбачених семплів класу 1;

FP – кількість неправильно передбачених семплів класу 1;

FN – кількість неправильно передбачених семплів класу 0.

Основними метриками для оцінки задач регресії є MSE, MAE, $MAPE$ та R^2.

Формульно MSE можна подати так:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \quad (4.5)$$

де n – кількість входжень;

y_i – реальне значення і-того входження;

$\hat{y}_i$ – передбачене значення і-того входження.

Формульно MAE можна подати так:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|, \quad (4.5)$$

де n – кількість входжень;

y_i – реальне значення і-того входження;

$\hat{y}_i$ – передбачене значення і-того входження.

Формульно $MAPE$ можна подати так:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|, \quad (4.5)$$

де n – кількість входжень;

y_i – реальне значення і-того входження;

$\hat{y}_i$ – передбачене значення і-того входження.

Формульно R^2можна подати так:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}, \quad (4.5)$$

де n – кількість входжень;

y_i – реальне значення і-того входження;

$\hat{y}_i$ – передбачене значення і-того входження;

$\bar{y}$ – середнє значення реальних входжень.

2.5 Висновки

У даному розділі було проведено детальний огляд математичної моделі алгоритму вибору найкращих ознак, серед яких можна виділити фільтраційні, обгорткові та вбудовані методи. Також було наведено формулювання основних алгоритмів машинного навчання: лінійної та логістичної регресій, дерева рішень, випадкового лісу, одно- та багатошарових перцептронів, графової нейронної мережі, техніки машин опорних векторів (SVM) та підсилення градієнтного бустингу (XGBoost). Було розглянуто математичний опис процесу баєсівської оптимізації для тюнінгу гіперпараметрів моделі. Розділ завершується описом основних метрик для оцінки моделей бінарної класифікації та регресії.

3 ON-PREMISE СИСТЕМА АВТОМАТИЗОВАНОГО ТРЕНУВАННЯ ТА МОНІТОРИНГУ МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ

3.1 Дослідження і вибір технологій для реалізації компонентів системи автоматизованого тренування та моніторингу моделей машинного навчання

Створення ефективних пайплайнів автоматизованого машинного навчання (AutoML) вимагає інтеграції широкого спектру технологій. Кожна з них відіграє важливу роль у автоматизації та оптимізації різних сегментів процесу машинного навчання. Гармонійне поєднання цих технологій є ключовим для ефективного функціонування AutoML пайплайнів, від етапу збору даних до їх деплойменту та наступного моніторингу. У цьому розділі буде розглянуто основні технологічні інструменти, необхідні для роботи екосистеми AutoML, та їх взаємодію.

3.1.1 Інструмент попередньої обробки даних

Порівняння інструментів попередньої обробки даних є важливим при виборі інструментарію для проєктів машинного навчання, особливо в контексті AutoML. Розглянемо декілька популярних інструментів: Pandas (Python), DataPrep (Python), Talend і Apache NiFi.

Ось основні критерії для порівняння:

- програмування або візуальний інтерфейс – чи інструмент використовує мову програмування або надає візуальний інтерфейс для обробки даних;
- легкість використання – наскільки просто вчитися і використовувати інструмент;
- гнучкість – здатність інструменту пристосовуватися до різних типів даних і завдань обробки.
- масштабованість – здатність обробляти великі обсяги даних;
- підтримка форматів даних – підтримувані формати даних (CSV, JSON, SQL тощо).

– інтеграція з іншими інструментами – легкість інтеграції з іншими інструментами та системами;

– спільнота та підтримка – розмір та активність спільноти, доступність документації та підтримки.

У таблиці 3.1 наведено порівняння інструментів попередньої обробки даних.

Найкращим варіантом для реалізації попередньої обробки даних запланованого AutoML пайплайну є Pandas.

Таблиця 3.1 – Порівняння інструментів попередньої обробки даних

Критерій	Pandas	DataPrep	Talend	Apache NiFi
Легкість використання	Висока	Висока	Помірна	Помірна
Гнучкість	Висока	Висока	Низька	Висока
Масштабованість	Помірна	Помірна	Висока	Помірна
Підтримка форматів даних	Висока	Помірна	Висока	Висока
Інтеграція з іншими інструментами	Висока	Помірна	Висока	Помірна
Спільнота та підтримка	Висока	Висока	Помірна	Висока

3.1.2 Фреймворк для машинного навчання

Фреймворки машинного навчання надають набір інструментів для розробки, тренування та валідації моделей. Для порівняння розглянемо кілька популярних фреймворків: Scikit-learn, TensorFlow, PyTorch, та XGBoost.

Основні критерії для порівняння:

– тип моделей – підтримка різних типів моделей машинного навчання (наприклад, лінійні моделі, нейронні мережі);

– легкість використання – наскільки просто освоїти та використовувати фреймворк.

– гнучкість – можливість налаштування та експериментування з моделями;

– масштабованість – здатність обробляти великі набори даних і складні архітектури моделей;

– спільнота та підтримка – розмір спільноти та доступність ресурсів для навчання та підтримки;

– інтеграція з іншими інструментами – легкість інтеграції з іншими інструментами та платформами.

У таблиці 3.2 наведено порівняння фреймворків машинного навчання. Ця таблиця дозволяє оцінити фреймворки з точки зору їх підходів до моделювання, їх легкості використання, гнучкості, масштабування та підтримки. Наприклад, Scikit-learn є відмінним вибором для традиційних алгоритмів машинного навчання, тоді як TensorFlow і PyTorch пропонують більшу гнучкість та масштабованість для нейронних мереж. XGBoost є лідером для завдань, де використовується ансамблеве навчання, особливо градієнтний бустинг.

Отже, найоптимальнішим варіантом для побудови AutoML пайплайну буде використання Scikit-learn для алгоритмів класичного машинного навчання та PyTorch для нейронних мереж.

Таблиця 3.2 – Порівняння фреймворків машинного навчання

Критерій	Scikit-learn	TensorFlow	PyTorch	XGBoost
Тип моделей	Класичний ML	Нейронні мережі	Нейронні мережі	Ансамблеві моделі (особливо градієнтний бустинг)

Кінець таблиці 3.2 – Порівняння фреймворків машинного навчання

Легкість використання	Висока	Середня	Середня	Висока
Гнучкість	Середня	Висока	Висока	Середня
Масштабованість	Середня	Висока	Висока	Висока
Спільнота та підтримка	Велика	Дуже велика	Дуже велика	Велика
Інтеграція з іншими інструментами	Висока	Висока	Висока	Висока

3.1.3 Інструмент оптимізації гіперпараметрів

Бібліотеки для тюнінгу гіперпараметрів мають велике значення у процесі оптимізації моделей машинного навчання. Для порівняння оберемо кілька популярних бібліотек: GridSearchCV з Scikit-learn, Hyperopt, Optuna, та Bayesian Optimization.

Основні критерії для порівняння:

− методи оптимізації – які методи оптимізації підтримуються (наприклад, пошук по сітці, випадковий пошук, баєсівська оптимізація);

− легкість використання – наскільки інтуїтивно зрозумілим та простим у використанні є інструмент;

− гнучкість – можливість налаштування процесу оптимізації під специфічні вимоги;

− масштабованість – здатність обробляти великі простори гіперпараметрів і великі датасети;

– інтеграція з іншими інструментами – легкість інтеграції з фреймворками машинного навчання;

– спільнота та підтримка – наявність документації, активність спільноти.

У таблиці 3.3 наведено порівняння бібліотек для тюнінгу гіперпараметрів. Ця таблиця дозволяє оцінити бібліотеки тюнінгу гіперпараметрів з точки зору їх методів оптимізації, легкості використання, гнучкості, масштабування та інтеграції. Наприклад, GridSearchCV є простим у використанні, але не завжди підходить для дуже великих просторів гіперпараметрів, тоді як Hyperopt та Optuna пропонують більшу гнучкість та масштабованість за рахунок байєсівської оптимізації.

За рахунок своєї універсальності й легкості у використанні, найоптимальнішим варіантом для тюнінгу гіперпараметрів є використання бібліотеки Optuna.

Таблиця 3.3 – Порівняння бібліотек для тюнінгу гіперпараметрів

Критерій	GridSearchCV (Scikit-learn)	Hyperopt	Optuna	Bayesian Optimization
Методи оптимізації	Пошук по сітці, випадковий пошук	Байєсівська оптимізація, випадковий пошук	Байєсівська оптимізація, випадковий пошук	Байєсівська оптимізація
Легкість використання	Висока	Середня	Висока	Середня
Гнучкість	Обмежена	Висока	Висока	Висока

Кінець таблиці 3.3 – Порівняння бібліотек для тюнінгу гіперпараметрів

Масштабованість	Обмежена (не підходить для дуже великих просторів)	Висока	Висока	Висока
Інтеграція з іншими інструментами	Висока	Висока	Висока	Середня
Спільнота та підтримка	Дуже велика	Велика	Велика	Середня

3.1.4 Сервіс хмарних провайдерів для побудови ML пайплайнів

Платформи хмарних обчислень і деплойменту моделей є ключовими для розгортання та ефективного використання моделей машинного навчання. Для порівняння розглянемо декілька популярних платформ: Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP), і IBM Cloud.

Основні критерії для порівняння:

- можливості машинного навчання – підтримка та інструменти для машинного навчання;
- масштабованість – здатність підтримувати розширення вимог до обчислювальних ресурсів;
- гнучкість деплойменту – можливості налаштування та оптимізації процесу деплойменту;
- інтеграція з іншими сервісами – сумісність та інтеграція з іншими хмарними сервісами та інструментами;
- ціноутворення – модель ціноутворення та вартість використання;

– безпека та надійність – наявність механізмів безпеки та гарантії стабільності.

У таблиці 3.4 наведено порівняння платформ хмарних обчислень. Ця таблиця дозволяє оцінити різні хмарні платформи з точки зору їх можливостей для машинного навчання, масштабованості, гнучкості деплойменту, інтеграції, ціноутворення та безпеки. Наприклад, AWS, Azure і GCP пропонують різноманітні інструменти та сервіси для машинного навчання з високою масштабованістю та гнучкістю.

За рахунок своєї ціни й послуг, які надаються хмарними провайдерами, було прийнято рішення відмовитися від їх використання на даному етапі розробки AutoML пайплайну.

Таблиця 3.4 – Порівняння платформ хмарних обчислень

Критерій	AWS	Microsoft Azure	Google Cloud Platform	IBM Cloud
Можливості машинного навчання	Широкий спектр сервісів	Широкий спектр сервісів	Широкий спектр сервісів	Обмеженіше порівняно з іншими
Масштабованість	Висока	Висока	Висока	Висока
Гнучкість деплойменту	Висока	Висока	Висока	Середня
Інтеграція з іншими сервісами	Висока	Висока	Висока	Висока

Кінець таблиці 3.4 – Порівняння платформ хмарних обчислень

Ціноутворення	Змінна, конкурентоспроможна	Змінна, конкурентоспроможна	Змінна, конкурентоспроможна	Змінна, може бути дорожче
Безпека та надійність	Високий рівень	Високий рівень	Високий рівень	Високий рівень

3.2 Налаштування роботи та взаємозв'язків компонентів системи автоматизованого тренування та моніторингу моделей машинного навчання

Apache Airflow - це платформа для оркестрації робочих процесів, яка дозволяє програмістам планувати, організовувати та моніторити робочі процеси за допомогою програмного коду. В основі її популярності лежить здатність моделювати складні робочі процеси як ациклічні графи (DAGs), де кожен вузол графа відображає завдання, а ребра – залежності між завданнями.

Airflow складається з кількох ключових компонентів, які разом формують потужну систему управління робочими процесами.

Веб-сервер: інтерфейс користувача, реалізований на Flask. Цей компонент дозволяє користувачам переглядати і управляти робочими процесами, стежити за їх виконанням, переглядати логи, конфігурувати налаштування тощо.

Планувальник: основний двигун Airflow, який відповідає за планування, запуск і моніторинг всіх задач, визначених у DAGs. Планувальник періодично перевіряє стан задач і запускає їх відповідно до їх залежностей та часових рамок.

Виконавець: компонент, який виконує завдання. Airflow підтримує кілька типів виконавців, наприклад, LocalExecutor, CeleryExecutor, KubernetesExecutor

тощо, які дозволяють налаштувати виконання завдань відповідно до потреб інфраструктури.

База даних: компонент, який зберігає інформацію про стан виконання робочих процесів, історію задач, конфігураційні налаштування та іншу метаінформацію. Airflow може використовувати різні СУБД, такі як PostgreSQL, MySQL, SQLite для цієї мети.

Журнал подій: записує детальні логи виконання для кожного завдання. Ці логи можуть бути сконфігуровані для зберігання в різних місцях, включаючи локальні файлові системи чи віддалені сховища даних.

Веб-сервер надає графічний інтерфейс користувача, що дозволяє користувачам з легкістю керувати їхніми робочими процесами. Веб-інтерфейс містить інформативні панелі, які відображають стан робочих процесів, що дозволяє користувачам швидко виявляти та вирішувати проблеми.

Планувальник розроблений для ефективного масштабування та забезпечує надійність планування завдань. Він використовує механізми збору задач, які забезпечують, що завдання виконуються вчасно та в заданому порядку залежностей.

Виконавець інтегрується з різними обчислювальними середовищами, забезпечуючи гнучкість у виборі підходящої конфігурації для виконання завдань. Це критично важливо для On-Premise розгортань, де ресурси можуть бути обмежені або специфічні.

База даних є центральним вузлом для управління станами та залежностями в Airflow. Надійність та цілісність даних забезпечується через використання транзакційних СУБД.

Система логування надає детальні дані для аналізу та відладки робочих процесів, що є незамінним для забезпечення високої якості виконання робочих процесів і швидкого виявлення та усунення проблем.

На рисунку 3.1 зображено архітектуру Airflow.

Directed Acyclic Graph (DAG) в Apache Airflow – це модель, яка представляє робочий процес як набір задач і залежностей між ними. Кожен DAG складається з одного або більше завдань (tasks), які виконуються в певній послідовності.

Визначення завдань: кожна задача в DAG описується як окремий елемент, який виконує певну дію, наприклад, виконання скрипту Python, запуск SQL-запиту або зовнішнього процесу.

Встановлення залежностей: залежності між задачами визначаються за допомогою спеціальних операторів, які показують, які задачі повинні виконуватися до або після інших. Наприклад, `task2.set_upstream(task1)` вказує, що `task2` не може початися до завершення `task1`.

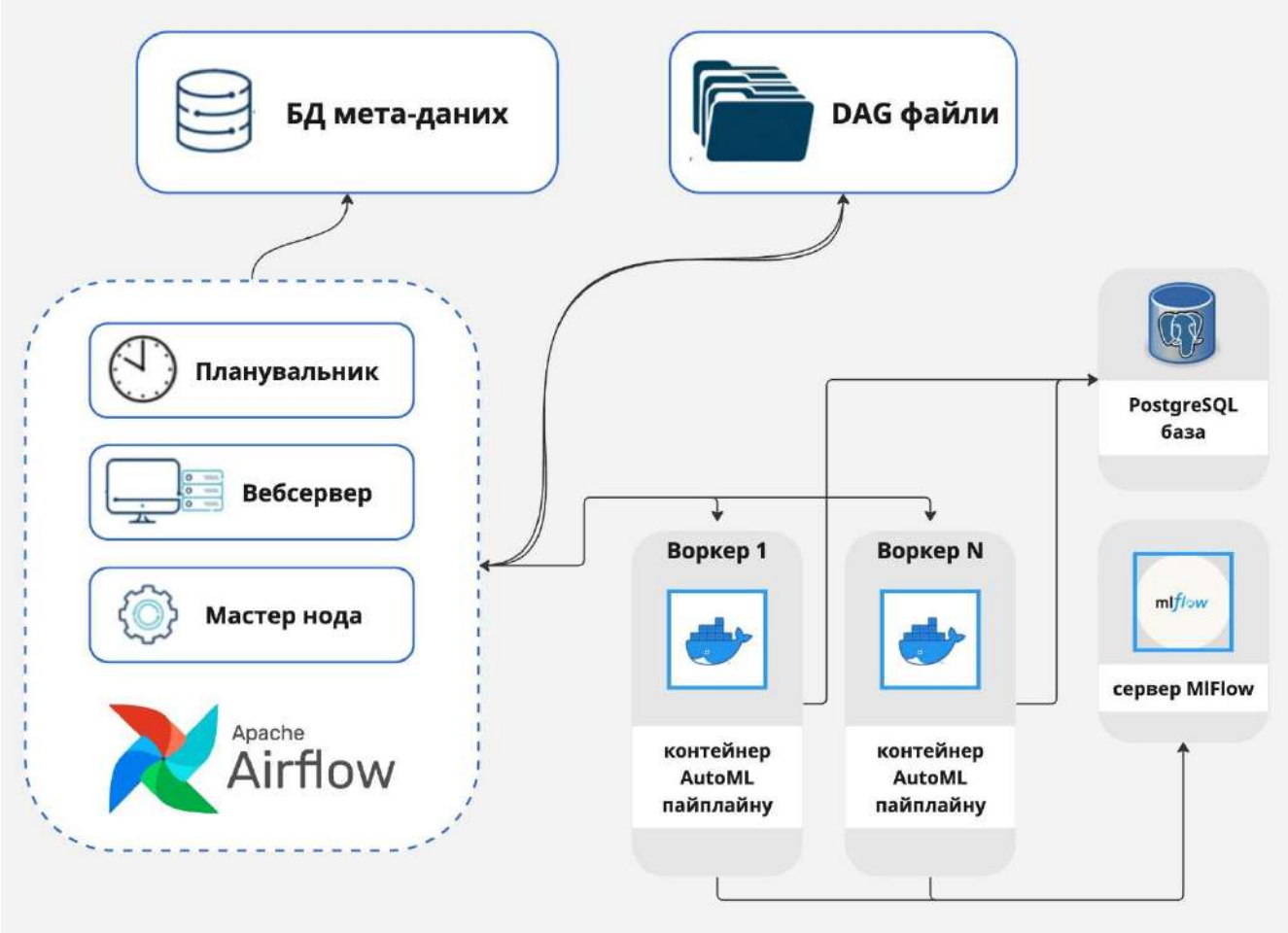


Рисунок 3.1 – Архітектура Airflow

Черговість виконання: Airflow використовує планувальник (Scheduler), щоб забезпечити виконання задач згідно з встановленими залежностями. Планувальник періодично перевіряє стан кожної задачі та розпочинає виконання наступних задач, коли залежні завершені.

Обмін інформацією: задачі можуть обмінюватися даними через механізм XComs у Airflow, який дозволяє задачам "спілкуватися" між собою, передаючи малі обсяги даних (наприклад, ідентифікатори файлів, шляхи до файлів або статуси).

Отже, окремі воркери мають дуже обмежені можливості для передачі інформації між собою. Для передачі великих обсягів інформації, обміну важкими файлами необхідно мати зовнішнє сховище для зберігання проміжних результатів таким чином, щоб містити вихідні дані попереднього кроку, для того щоб наступний крок міг їх скачати у якості вхідних.

Нехай маємо задачу, виконання якої займає тривалий час і багато обчислювальних ресурсів. Тоді, якщо ця задача може бути розпаралелена (наприклад, препроцесинг датасету шляхом його розділення на батчі), тоді цю паралелізацію можна налаштувати і в DAG. На рівні виконавців (executors) паралелізм в Airflow досягається шляхом використання різних типів виконавців, які можуть керувати одночасним виконанням багатьох задач. Наприклад, `LocalExecutor` виконує задачі паралельно на одній машині, тоді як `CeleryExecutor` може використовувати багатомашинний кластер для розподілу задач. На рисунку 3.2 зображено блок-схему об’єднання воркерів в єдиний кластер.

Використання CeleryExecutor дозволяє Airflow управляти кластером робочих процесів (workers), які можуть розміщуватися на різних машинах. Для цього потрібно налаштувати Celery з брокером повідомлень (наприклад, RabbitMQ або Redis) і конфігураційним файлом Airflow, щоб вказати зв'язок між воркерами.

Для налаштування безпеки кластеру зазвичай вдаються до застосування таких технологій:

– аутентифікація та авторизація – використання модулів безпеки Airflow, таких як LDAP або OAuth, для забезпечення контрольованого доступу до веб-інтерфейсу та API;

– шифрування – налаштування шифрування для збереження чутливих даних, використовуючи засоби, як-от Fernet keys, для шифрування зберіганих у базі даних значень.

– журналювання та моніторинг – налаштування журналювання для відстеження всіх дій у системі та використання засобів моніторингу для спостереження за станом виконання задач та здоров'ям кластера.

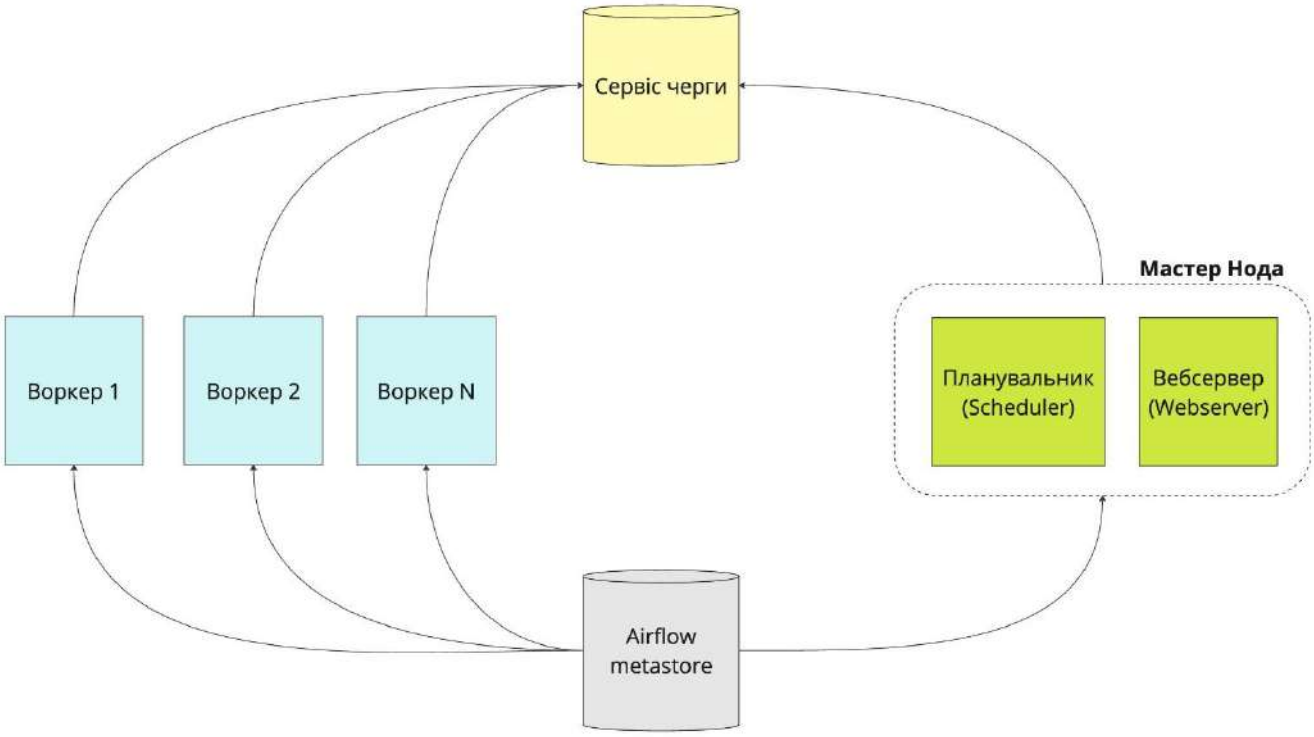


Рисунок 3.2 – Блок-схема об’єднання воркерів в єдиний кластер

3.3 Проведення тестувань On-Premise провайдерів для побудови SAAS платформи

Вибір провайдера ресурсів для розгортання On-Premise інфраструктури є стратегічно важливим кроком в процесі створення та підтримки SAAS платформи для розробки лікарських засобів. Аналізуючи різних провайдерів, які надають в оренду серверні потужності, ми розглянемо їх в контексті надання ресурсів, що дозволяють оптимізувати витрати та підвищити продуктивність платформи.

Розглянемо кілька компаній, які пропонують сервери в оренду за приблизно однакову ціну – близько 40 доларів на місяць. Цей бюджет був обраний для того, щоб установити рівні умови порівняння для кожного з них.

Німецький провайдер Hetzner представляє сервіси з різним спектром ресурсів – від VPS до присвячених серверів. Цінова політика Hetzner є однією з найбільш конкурентоспроможних на ринку, що робить їх вигідним вибором для стартапів та дослідницьких проектів. Hetzner вирізняється найнижчими цінами серед розглянутих опцій і пропонує три різні конфігурації, які аналізувались з точки зору ціни і продуктивності:

- хмарний сервер CPX31 – VPS із 4 спільними vCores, 8 ГБ оперативної пам’яті та 160 ГБ пам’яті NVME; ціна: €12,40/міс;
- хмарний сервер CCX22 – VPS із 4 виділеними vCores, 16 ГБ оперативної пам’яті та 160 ГБ пам’яті NVME; ціна: €34,90/міс;
- виділений сервер AX41-NVME – 6 ядер Ryzen 5 3600 (тобто 12 vCores), 64 ГБ оперативної пам’яті та 2x 512 ГБ пам’яті NVM; ціна: €34,00/міс.

Найбільше враження справила пропозиція виділеного сервера AX41-NVME, яка, за аналогічну вартість, надавала значно більше ресурсів у порівнянні з VPS від інших провайдерів.

Linode, відомий своїми хмарними послугами, пропонує стандартні конфігурації VPS, що конкурують за ціною з DigitalOcean та UpCloud. Однак, згідно з результатами тестування, продуктивність Linode виявилася нижчою від очікувань.

DigitalOcean - один із популярних виборів серед розробників, надає послуги хмарних серверів з балансом ціни та продуктивності. Тестування показало, що їх пропозиції є цілком конкурентними, але за певних обставин вони можуть поступатися іншим провайдерам.

UpCloud позиціонує себе як провайдера з найшвидшими хмарними серверами завдяки своїй технології MaxIOPS. Проте отримані дані свідчать, що їх продуктивність не завжди виправдовує очікування.

Linode, DigitalOcean і UpCloud були оцінені за стандартну конфігурацію VPS з 4 спільними ядрами, 8 ГБ ОЗУ та 160 ГБ SSD сховища за ціною в 40 доларів на місяць. Їхні пропозиції хоч і популярні, але не вразили в контексті продуктивності.

Terrahost надає сервіси з потужними ядрами Ryzen 5950X, що демонструють видатну продуктивність, особливо в області обчислювальних можливостей CPU. Норвезький провайдер Terrahost, хоч і є менш відомим на ринку, приємно здивував високими результатами тестування. Сервери з процесорами Ryzen 5950X показали відмінні результати, що робить їх привабливими для вимогливих задач.

Scaleway відрізняється тим, що пропонує трохи відмінну конфігурацію з 12 ГБ ОЗУ, але з меншим SSD сховищем в 120 ГБ. Останнім часом компанія підвищила ціни, і зараз її пропозиція стає найбільш дорогою серед розглянутих.

Бенчмаркінг проводився за допомогою популярних скриптів yabs.sh та nench.sh. Ці інструменти дозволяють оцінити продуктивність пам’яті та CPU.

3.3.1 Аналіз продуктивності пам’яті

Як ми бачимо із рисунку 3.3 пропускна здатність читання Мб/с найкраща у Terrahost. Різниця з іншими – за винятком виділеного сервера від Hetzner – величезна. Найімовірніше, виділені нам віртуальні сервери опинилися на абсолютно нових хостах із дуже малою кількістю інших клієнтів або зовсім без них, про що свідчать імена хостів (host1, host2, host3) і послідовні ідентифікатори віртуальних серверів. Тож цілком імовірно, що продуктивність може бути іншою із збільшенням навантаження на гіпервізори, але різниця з іншими настільки велика, що продуктивність все одно очікується на високому рівні.

Викликає розчарування UpCloud, оскільки вони стверджують, що мають найшвидші хмарні сервери у світі та найшвидші диски з технологією MaxIOPS. Проте тести показали протилежну ситуацію.

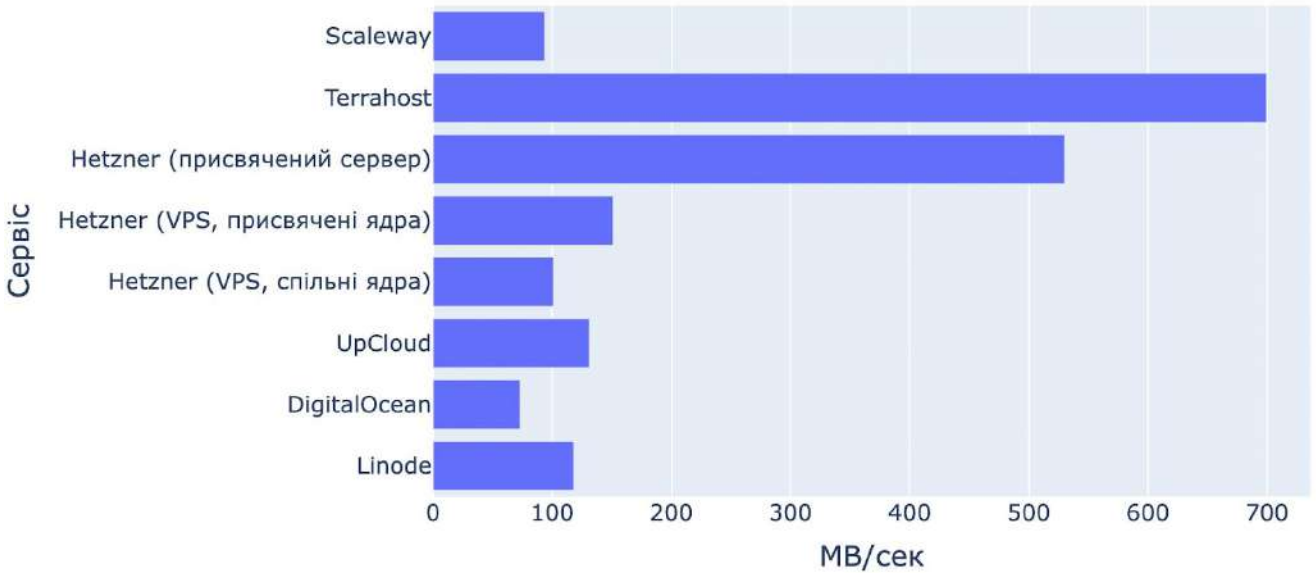


Рисунок 3.3 – FIO диск розміром блоку 4K – пропускна здатність на зчитування Мб/с

На рисунку 3.4 зображено пропускну здатність на зчитування IOPS. Згідно графіку, Terrahost та Hetzner демонструють вражаючі результати. В той же час UpCloud, DigitalOcean і Linode також дуже розчаровують, враховуючи, наскільки вони популярні.

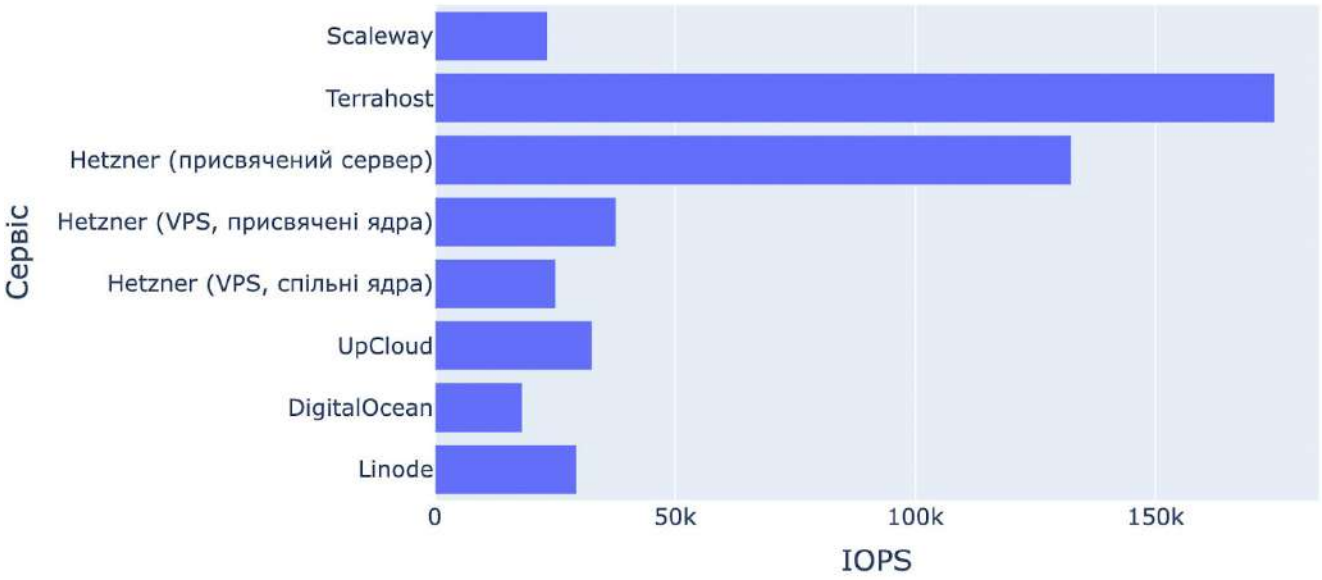


Рисунок 3.4 – FIO диск розміром блоку 4K – пропускна здатність читання IOPS

На рисунку 3.5 зображено пропуску здатність запису Мб/с. Результати показують ту саму картину, що і у попередньому пункті.

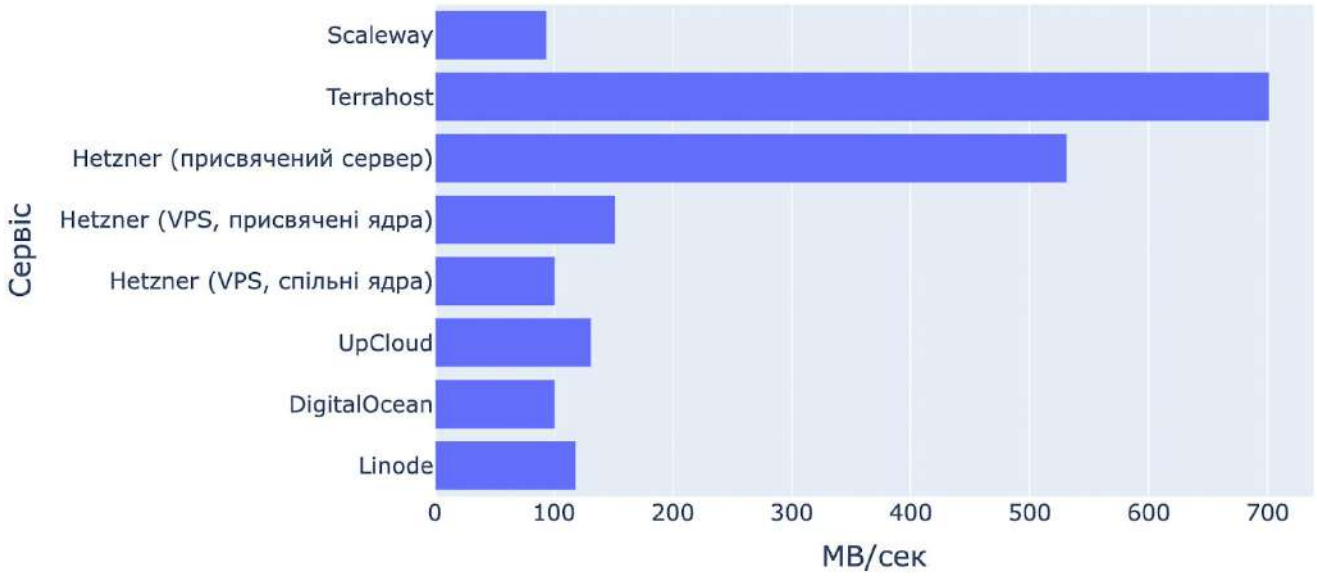


Рисунок 3.5 – FIO диск розміром блоку 4K – пропускна здатність на запис Мб/с

На рисунку 3.6 зображено пропускну здатність запису IOPS. Результати показують ту саму картину, що й на попередніх діаграмах.

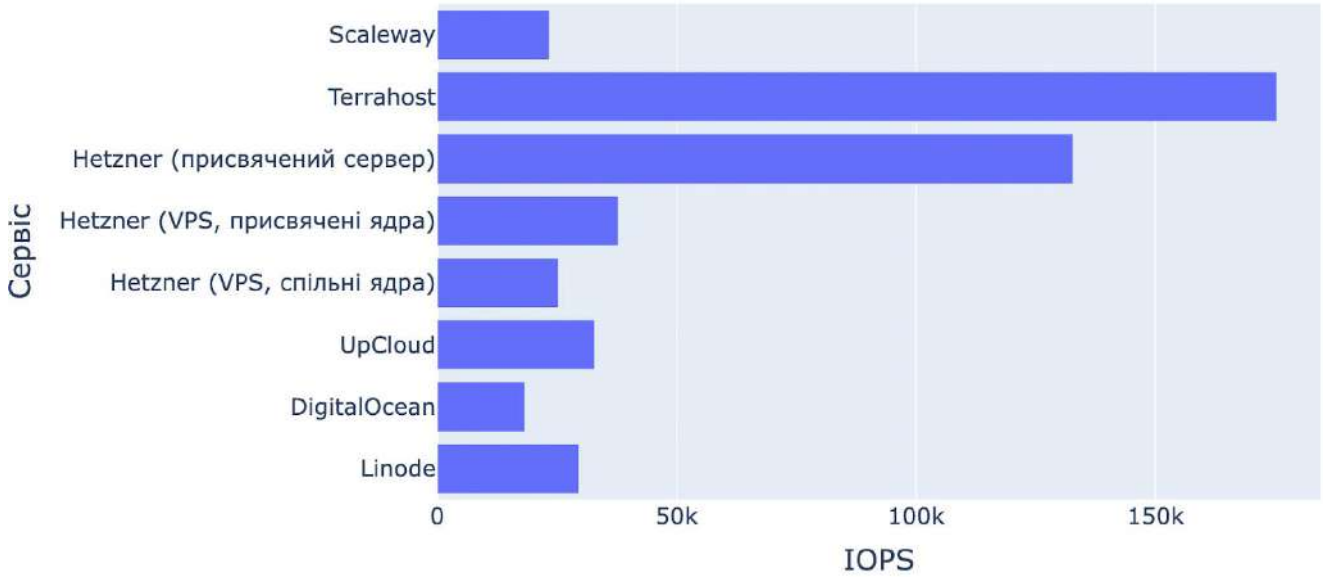


Рисунок 3.6 – FIO диск розміром блоку 4K – пропускна здатність запису IOPS

3.3.2 Аналіз продуктивності CPU

У якості метрики для порівняння було обрано Geekbench 5. Geekbench 5 є одним із найбільш популярних бенчмаркінгових інструментів, який використовується для оцінювання продуктивності процесорів в реальних та синтетичних сценаріях. Цей інструмент дозволяє користувачам вимірювати продуктивність їхніх систем і порівнювати її з іншими системами на базі загальноприйнятих стандартів.

Geekbench 5 проводить ряд тестів, які вимірюють продуктивність процесора за допомогою різних завдань, що відображають реальні сценарії використання. Ці тести включають обчислення, що вимагають великої кількості обробки даних, такі як шифрування, обробка зображень, фізичні симуляції та інші. Відмінною особливістю Geekbench 5 є його здатність навантажувати декілька ядер процесора одночасно, що робить його ідеальним інструментом для оцінки багатоядерних систем.

Geekbench 5 вимірює продуктивність процесора за двома основними метриками. Одноядрова оцінка: оцінює продуктивність одного ядра процесора. Ця метрика критично важлива для додатків, які не оптимізовані для використання декількох потоків або ядер. Вища одноядрова оцінка свідчить про кращу продуктивність процесора в однопоточних задачах. Багатоядрова оцінка: оцінює загальну продуктивність процесора при виконанні завдань, що використовують декілька ядер одночасно. Ця метрика відображає здатність процесора ефективно розподіляти обчислювальне навантаження між ядрами, що є важливим для сучасних багатозадачних або багатокористувацьких систем.

Завдяки цьому Geekbench 5 використовується для оцінювання процесорів у різних областях, включаючи:

− вибір апаратного забезпечення – допомагає користувачам та організаціям вибрати оптимальне апаратне забезпечення відповідно до їх потреб в обчислювальній потужності;

– розробка та тестування програмного забезпечення – розробники використовують Geekbench для тестування оптимізації своїх додатків для різних процесорів;

– наукові дослідження – академічні та промислові дослідницькі групи аналізують продуктивність обчислювальних систем за допомогою Geekbench для вирішення складних наукових задач.

Отже, Geekbench 5 є важливим інструментом у сфері аналізу продуктивності процесорів, який надає об'єктивні та точні дані про здатність процесорів справлятися з сучасними обчислювальними викликами. Його здатність оцінювати як одноядерні, так і багатоядерні аспекти продуктивності робить його незамінним інструментом в руках кожного, хто залучений до вибору або розробки апаратних ресурсів.

На рисунку 3.7 зображено одноядрову оцінку за допомогою Geekbench 5. На даному графіку можна побачити наскільки потужним є Ryzen 5950X на серверах Terrahost. І ця перевага є очевидною, навіть для спільнтх ядер. Продуктивність одного ядра краща, ніж у виділеного сервера Hetzner, який має процесор Ryzen 3600, тому це очікувано, якщо не брати до уваги різницю між спільними та виділеними на один момент, оскільки гіпервізори Terrahost, швидше за все, були новими/порожніми. Хмарні сервери Hetzner також вражають результатами, враховуючи те, що вони випередили UpCloud і коштували набагато дешевше. Дивно, але екземпляр DEV від Scaleway виявився кращим, ніж DigitalOcean і Linode, оскільки програма DynaBlogger Rails здавалася повільнішою на Scaleway під час тестування.

На рисунку 3.8 зображено багатоядрову оцінку за допомогою Geekbench 5. Цілком очікуваним є те, що багатоядерний результат виділеного сервера вищий, ніж у віртуальних серверів Terrahost, оскільки він має 6 ядер проти 4. Проте Terrahost все ще у лідерах. Викликає подив той факт, що хмарний сервер Hetzner із спільними ядрами працює краще, ніж сервер із виділеними ядрами. UpCloud показали кращі результати в цьому тесті, ніж в інших, а DigitalOcean і Linode знову розчарували.

Отже, у розрізі потужності процесорів, Terrahost домінує завдяки потужним ядрам Ryzen 5950X. Проте, враховуючи, що виділені сервери Hetzner продаються за ціною спільних ресурсів, їхні результати є найкращими по співвідношенню результативність / вартість.

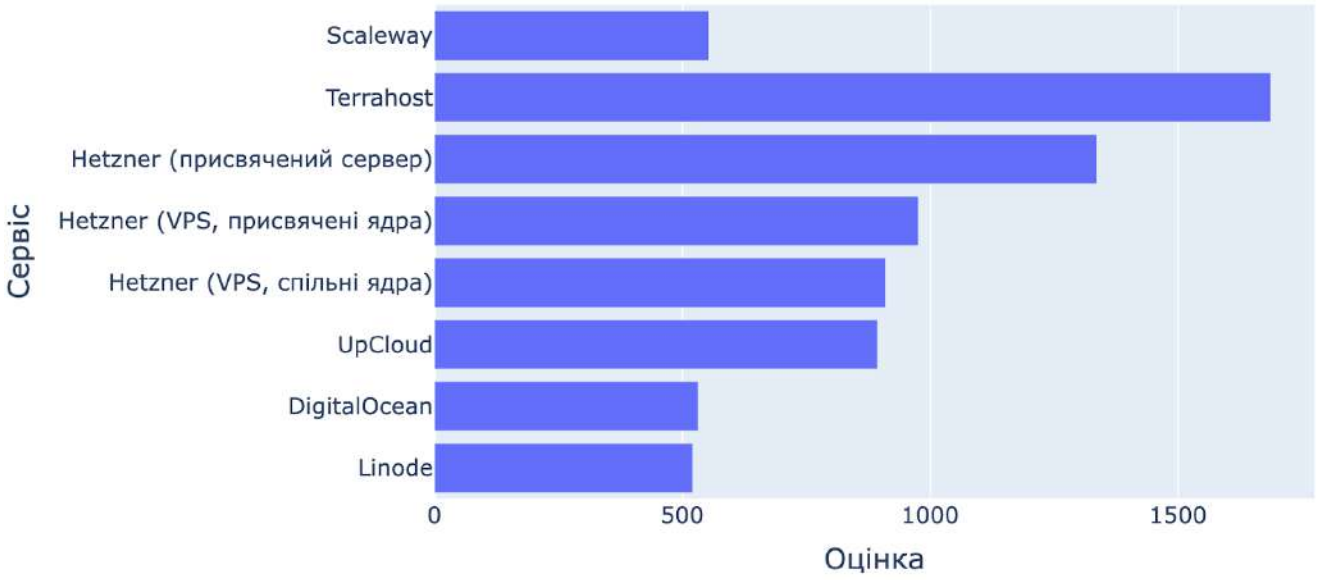


Рисунок 3.7 – Одноядрова оцінка Geekbench 5

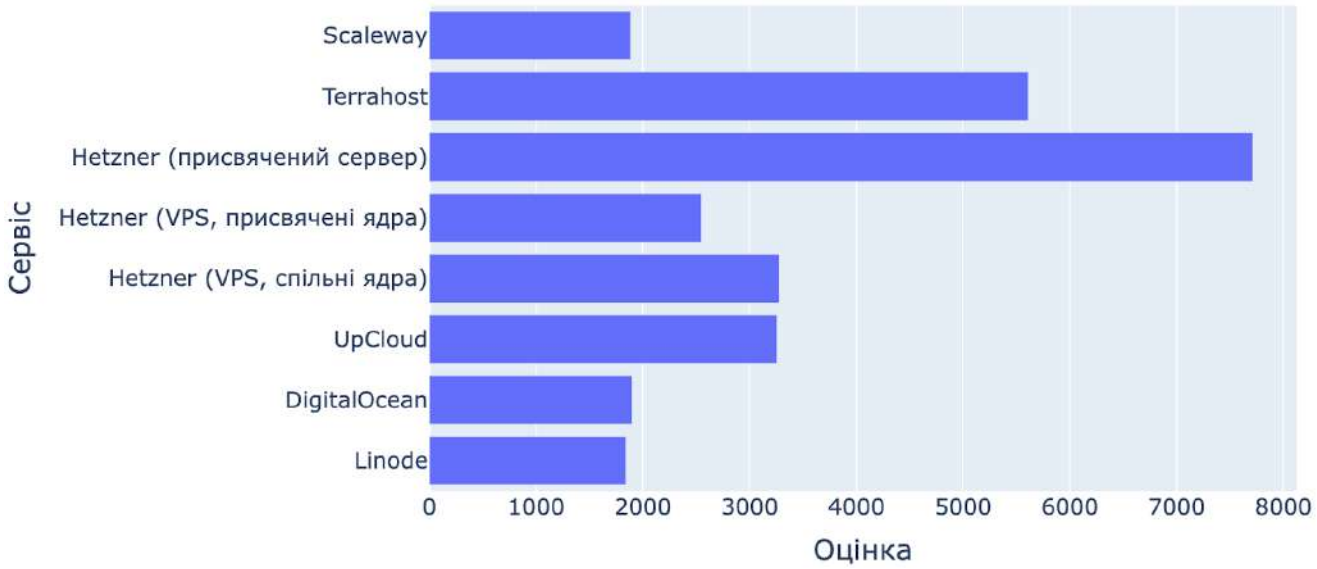


Рисунок 3.8 – Багатоядрова оцінка Geekbench 5

При виборі провайдера важливо враховувати не тільки ціну, але й відповідність продуктивності потребам проекту. Hetzner пропонує найкраще

співвідношення ціна/продуктивність, особливо з виділеними серверами. Якщо вибір падає на класичні VPS, то Terrahost виглядає перспективним варіантом з точки зору продуктивності CPU.

3.4 Опис On-Premise версії AutoML пайплайну системи автоматизованого тренування та моніторингу моделей машинного навчання з оркестрацією AirFlow

Пайплайн реалізовує процес автоматичного тренування стекінгу моделей. В якості платформи для трекінгу ML експериментів було обрано MlFlow. Тому перший етап відповідає за ініціалізацію MlFlow експерименту та відповідного рану. На наступних етапах генеруються молекулярні дескриптори, а далі відбувається відбір найкращих з них. Наступні етапи – тренування моделей ML.

На першому етапі (base) для кожного сімейства моделей з MODEL_FAMILIES підбираються найкращі гіперпараметри. Тюнінг триває TIMEOUT_BASE, після чого з кожного сімейства обираються моделі з найвищими значеннями OPTIMIZATION_METRIC метрики. Серед найкращих моделей на наступний етап відбираються лише ті, у яких значення метрики більше за SCORE_THRESHOLD.

На другому етапі (meta) відбувається процес побудови мета-моделі в стекінговій архітектурі. Стекінг – це ансамблевий метод машинного навчання, який використовує кілька базових моделей, зазвичай різних типів, для генерації передбачень, які потім використовуються як вхідні дані для мета-моделі. Ця мета-модель має завдання використати передбачення базових моделей для кінцевого передбачення. Базові моделі: різні алгоритми машинного навчання, такі як дерева рішень, машини підтримки векторів (SVM) та нейронні мережі, навчаються на тренувальному наборі даних і роблять незалежні передбачення. Збір передбачень: передбачення від кожної базової моделі збираються і використовуються як вхідні дані для мета-моделі. Мета-модель: ця модель навчається використовувати передбачення від базових моделей для вироблення фінального передбачення. Це може бути ще один алгоритм машинного навчання, оптимізований для інтеграції

різних типів передбачень. Стекінг часто покращує передбачувальну точність, об'єднуючи сильні сторони різних моделей і мінімізуючи вплив їх слабких сторін.

Після завершення тюнінгу мета-моделі, відбувається тренування і валідація цілісної стекінгової моделі, де кожна з моделей-компонент є найкращою моделлю з попередніх етапів.

Кожен етап пайплайну виконується у власному Docker контейнері. Це дозволяє ізолювати виконання кожного етапу, забезпечує відтворюваність середовища виконання і легкість масштабування. Також, це забезпечує універсальність пайплайну, адже можна перевикористати ті ж самі компоненти для On-Cloud рішення без необхідності вручну налаштовувати середовище.

На рисунку 3.9 зображено компоненти пайплайну.

Рисунок 3.9 – Компоненти пайплайну

1. init_mlflow_experiment: ініціалізація експерименту в MLflow.
2. generate_descriptors: генерація дескрипторів для датасету.
3. select_descriptors: вибір релевантних дескрипторів.
4. automl_base: тренування базових моделей.

5. automl_meta: тренування мета-моделі.
6. log_automl_to_mlflow: логування результатів AutoML в MLflow.

Кожен із етапів пайплайну є конфігурованим і налаштовується шляхом передавання змінних, описаних у таблиці 3.5.

Таблиця 3.5 – Етапи пайплайну

Назва змінної	Призначення	Етапи пайплайну, де використовується
MLFLOW_DB_URL	URL бази даних для MLflow	init_mlflow_experiment, generate_descriptors, select_descriptors, log_automl_to_mlflow
EXPERIMENT_NAME	Назва експерименту в MLflow	Всі етапи, окрім init_mlflow_experiment
RUN_NAME	Назва запуску експерименту в MLflow	Всі етапи, окрім init_mlflow_experiment
DATASET_FOLDER	Папка з датасетами	generate_descriptors, select_descriptors, automl_base_task_list, automl_meta_task_list, log_automl_to_mlflow
DATASET_FILE	Файл датасету	generate_descriptors, select_descriptors, automl_base_task_list, automl_meta_task_list, log_automl_to_mlflow

Продовження таблиці 3.5 – Етапи пайплайну

DESCRIPTORS_FILE	Файл з дескрипторами	generate_descriptors, select_descriptors, automl_base_task_list, automl_meta_task_list, log_automl_to_mlflow
DESCRIPTORS_TYPES	Типи дескрипторів	generate_descriptors
SELECTED_DESCRIPTORS_FILE	Файл з вибраними дескрипторами	select_descriptors, automl_base_task_list, automl_meta_task_list, log_automl_to_mlflow
TARGET_FILE	Файл з цільовими значеннями	select_descriptors, automl_base_task_list, automl_meta_task_list, log_automl_to_mlflow
PROBLEM_TYPE	Тип задачі (класифікація, регресія тощо)	automl_base_task_list, automl_meta_task_list
MODEL_FAMILIES	Сімейства моделей для використання	automl_base_task_list, automl_meta_task_list
OPTIMIZATION_METRIC	Цільова метрика для оптимізації	automl_base_task_list, automl_meta_task_list
SCORE_THRESHOLD	Мінімальне значення метрики, для прийняття моделі	automl_base_task_list, automl_meta_task_list

Кінець таблиці 3.5 – Етапи пайплайну

TIMEOUT_BASE	Базовий часовий ліміт для AutoML	automl_base_task_list, automl_meta_task_list
TIMEOUT_META	Мета часовий ліміт для AutoML	automl_meta_task_list
OPTUNA_DB_URL	URL бази даних для Optuna	automl_base_task_list, automl_meta_task_list
LEVEL	Рівень обробки в AutoML (base, meta)	automl_base_task_list, automl_meta_task_list

3.5 Висновки

У даному розділі було досліджено вибір технологій для реалізації компонентів системи автоматизованого тренування та моніторингу моделей машинного навчання. Також було налаштовано роботу та взаємозв’язки компонентів системи і проведено тестування On-Premise провайдерів для побудови SAAS платформи. Розділ завершується описом On-Premise версії AutoML пайплайну за допомогою оркестратора AirFlow.

4 РЕАЛІЗАЦІЯ ТА ІНТЕГРАЦІЯ РОЗРОБЛЕНОЇ СИСТЕМИ В SAAS ПЛАТФОРМУ ДЛЯ РОЗРОБКИ ЛІКІВ

4.1 Опис SAAS платформи для розробки ліків

Побудована SAAS платформа є платформою, що застосовує штучний інтелект у пошуку нових лікарських препаратів та пропонує ретельно перевірені стратегії відкриття ліків, які адаптовані до унікальних викликів кожної цільової молекули. Вона використовує комплексний підхід для виявлення потенційних ліків, комбінуючи передові алгоритми штучного інтелекту та фізично обґрунтовані комп'ютерні методи.

Оптимізовані потенційні ліки послідовно перетворюються на стратегічно розроблені похідні сполуки. Потім їх оцінюють за допомогою фармакокінетичних і токсикологічних профілів за допомогою низки моделей штучного інтелекту, які точно налаштовані на основі результатів реальних експериментів.

Поєднання алгоритмів штучного інтелекту з фізично обґрунтованими методами прискорює процес оптимізації лідерських сполук. Завдяки машинному навчанню для аналізу даних і молекулярній динаміці для уточнення структури ми тонко налаштовуємо лідерські сполуки для оптимальної ефективності, безпеки та фармакокінетики.

Забезпечення безпеки, ефективності, селективності, стабільності та біодоступності майбутнього препарату є фінальними кроками на шляху до його виведення на ринок. Наша платформа забезпечує комплексний профіль поліфармакології кандидатів за допомогою знаннєвих графів і підтвердження за допомогою молекулярної динаміки та квантово-хімічних технік.

4.2 Аналіз експериментів засобами MLFlow

MLFlow UI надає інтуїтивно зрозумілий інтерфейс для відстеження та аналізу експериментів машинного навчання. Він дозволяє користувачам організовувати

експерименти, відображати детальну інформацію про кожен запуск і порівнювати результати між різними запусками.

Організація експериментів:

− MLFlow дозволяє створювати окремі експерименти для кожного ADME-Tox параметру, що полегшує керування експериментами та аналіз результатів;

− кожен експеримент об'єднує усі запуски (runs), пов'язані з конкретним параметром, дозволяючи спостерігати за всіма експериментами в одному місці.

Специфікація запусків: у назві кожного запуску міститься важлива інформація, така як тип датасету і цільова метрика. Це спрощує розуміння контексту та цілей кожного запуску.

На рисунку 4.1 зображено користувальницький інтерфейс MLFlow, який відображає список експериментів.

Інтерфейс користувача включає такі можливості:

− відображення результатів – таблиця відображає ключові деталі кожного рану, такі як тривалість, ім'я, користувача, версію коду та моделі, а також інформація організована таким чином, щоб користувачі могли оцінити результати експерименту на одному екрані;

− порівняння експериментів – функціонал порівняння дозволяє відкрити кілька ранів одночасно для аналізу різниці в показниках та гіперпараметрах;

− експорт даних – завантаження результатів у форматі CSV для подальшого аналізу або звітності;

− версіонування моделей – кожен ран може бути пов'язаний з версією моделі, що спрощує управління версіями та розгортання моделей для їх використання в майбутньому.

На рисунках 4.2, 4.3 та 4.4 зображено інтерфейс відображення основних параметрів, метрик та додаткової інформації експериментів у MLFlow.

Рисунок 4.1 – Користувальницький інтерфейс MLflow

Name	Value
base_CatBoost_auto_class_weights	None
base_CatBoost_l2_leaf_reg	0.5198349069607762
base_CatBoost_learning_rate	0.09268596443136451
base_CatBoost_max_depth	8
base_ExtraTrees_ccp_alpha	0.00019242246203737823
base_ExtraTrees_class_weight	None
base_ExtraTrees_max_depth	30
base_ExtraTrees_max_features	None
base_ExtraTrees_min_samples_leaf	9
base_ExtraTrees_min_samples_split	25
base_ExtraTrees_n_estimators	693

Рисунок 4.2 – Відображення параметрів експеримента у MLFlow

base_CatBoost_cv_accuracy_balanced_multiclass	0.696
base_CatBoost_cv_accuracy_multiclass	0.742
base_CatBoost_cv_f1_macro	0.714
base_CatBoost_cv_f1_micro	0.742
base_CatBoost_cv_f1_weighted	0.738
base_CatBoost_cv_log_loss_multiclass	0.608
base_CatBoost_cv_mcc_multiclass	0.58
base_CatBoost_cv_precision_macro	0.752
base_CatBoost_cv_precision_micro	0.742
base_CatBoost_cv_precision_weighted	0.746
base_CatBoost_cv_recall_macro	0.696

Рисунок 4.3 – Відображення метрик експеримента у MLFlow

Name	Value	Actions
BEST_BASE_MODEL_FAMILY	SVMLinearOvO	
BEST_META_MODEL_FAMILY	LinearOvO	
BEST_MODEL_SCORER	auto	
CV_SPLITTER_INSTANCE	{'method': 'StratifiedKFold', 'kwargs': {'n_splits': 5, 'shuffle': True, 'random_state': 47}}	

Рисунок 4.4 – Відображення додаткової інформації експеримента у MLFlow

4.3 Інтеграція пайплайну в інтерфейс SAAS платформи

Інтерфейс був розроблений таким чином, щоб користувач мав змогу детально налаштувати тренування моделей відповідно до власних задач. Також, йому надається можливість детально проаналізувати результати кожного етапу роботи пайплайну.

Отож, воркфлоу запуску AutoML пайплайну наступний. По-перше користувач повинен завантажити власний тренувальний датасет із молекулами. На рисунку 4.5 зображено сторінку завантаження користувачем власного тренувального датасету.

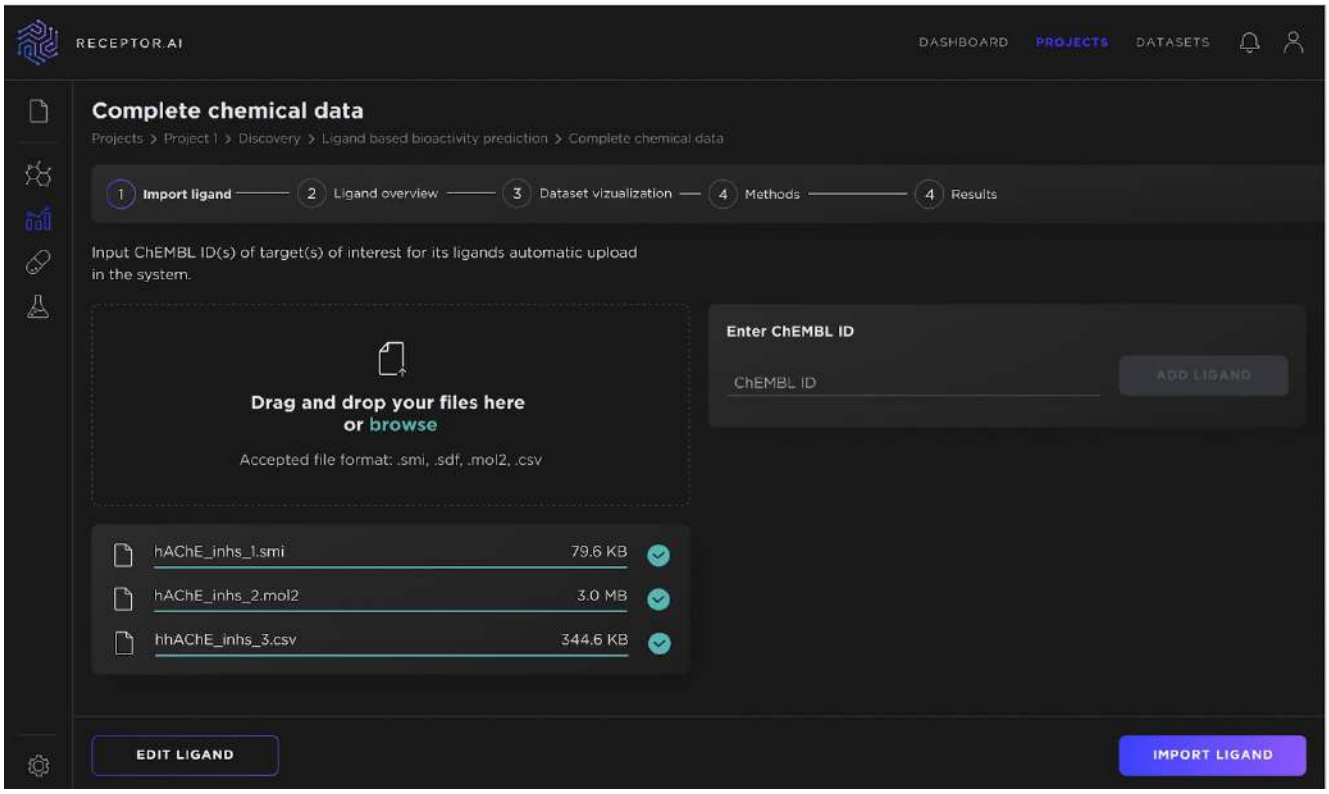


Рисунок 4.5 – Сторінка завантаження користувачем власного тренувального датасету

Після того, як сервер прийняв завантажений користувачем тренувальний датасет, йому надається можливість переглянути молекули, що містяться у ньому. Це важливий етап верифікації користувачем правильності завантажених даних. Тут

користувач повинен обрати колонку, яка буде використана в якості таргет змінної, передбачати значення якої і буде задачею моделей в AutoML пайплайні. На рисунку 4.6 зображено сторінку візуалізації завантаженого користувачем тренувального датасету. Додатково можна звести регресійну задачу до класифікаційної, якщо задати значення катофу, по якому виконувати дискретизацію змінної.

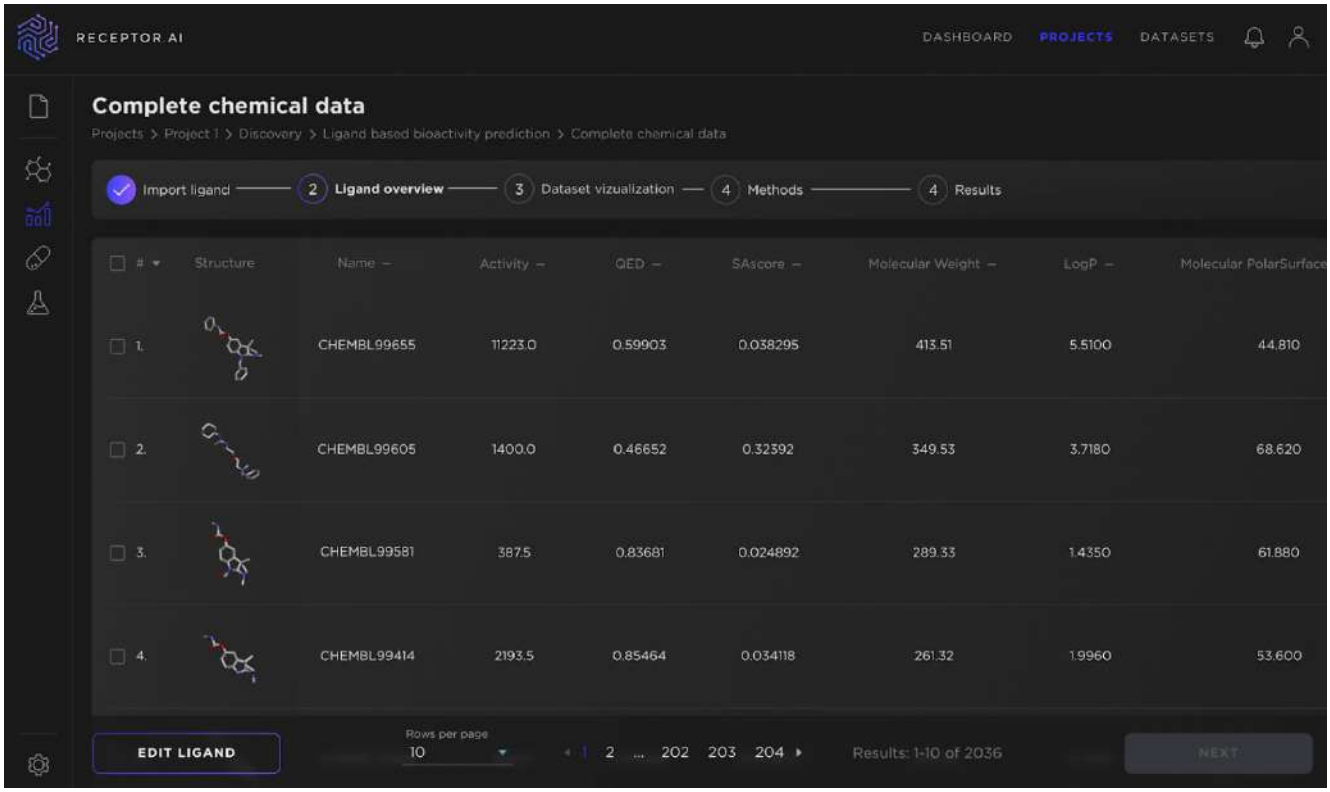


Рисунок 4.6 – Сторінка візуалізації завантаженого користувачем тренувального датасету

Наступний важливий етап – це проведення дослідницького аналізу даних – візуалізація статистик та розподілів різних параметрів у датасеті. На цьому етапі користувач може виявити залежності або проблеми, приховані у датасеті, і отримати важливі висновки для наступного етапу налаштування параметрів запуску пайплайну. На рисунку 4.7 зображено сторінка дослідницького аналізу даних.

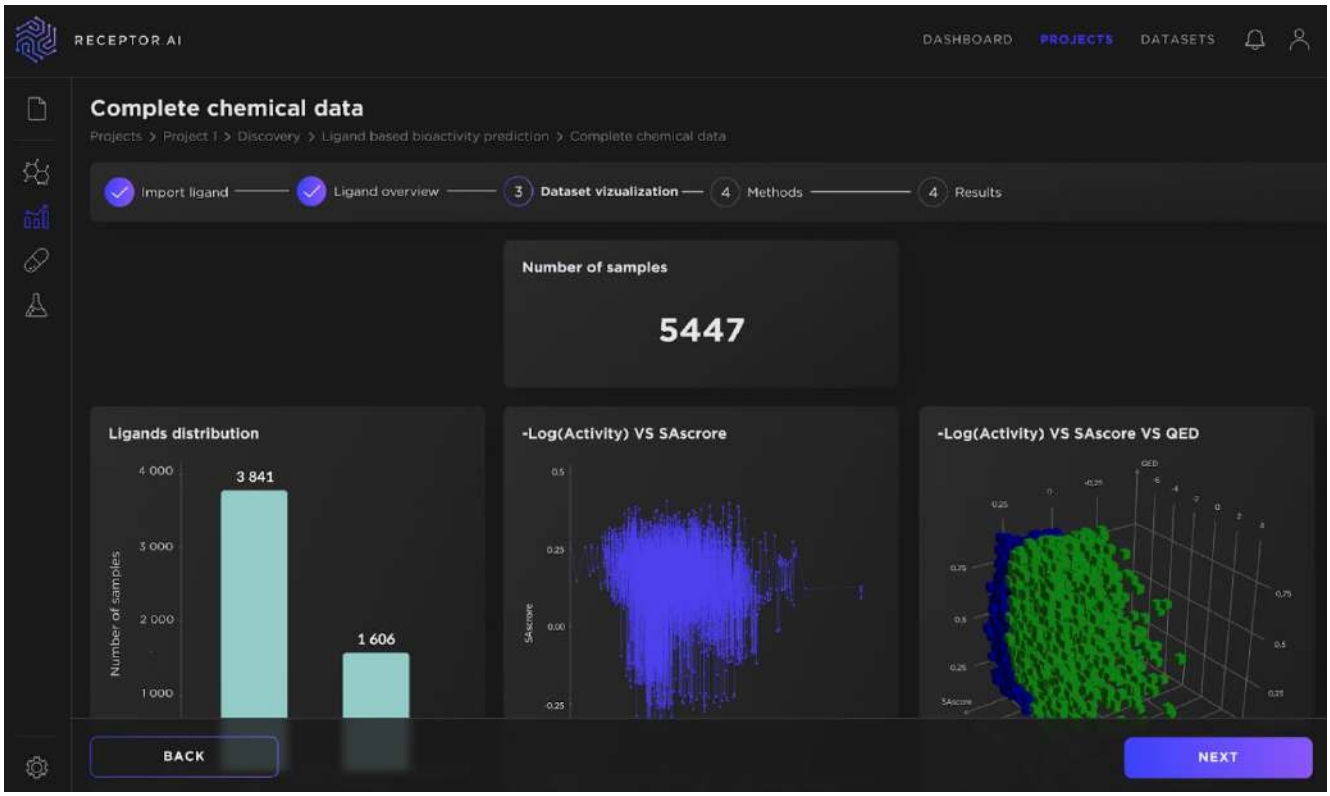


Рисунок 4.7 – Сторінка дослідницького аналізу даних

Після проведення візуалізації датасету, користувач переходить до налаштування параметрів тренування моделей. Інтерфейс передбачає можливість зручно і централізовано налаштувати усі параметри, які були описані у попередніх розділах. На рисунку 4.8 зображено сторінку налаштувань тренувального етапу пайплайну.

Після того, як закінчився процес тренування моделі, система переводить користувача на сторінку аналізу результатів. Тут користувач може детально подивитися на різні метрики, розподіл на тренувальні і валідаційні сети, візуалізацію передбачень для конкретних молекул, графіки зменшення розмірності тощо. На рисунках 4.9 та 4.10 зображено приклади результатів.

4.4 Аналіз продуктивності розробленої системи на датасетах ADME-Tox

Проаналізуємо продуктивність розробленої системи на прикладі моделей ADME-Tox. У таблицях 4.1 та 4.2 наведено результати побудованих моделей бінарної класифікації та регресії відповідно. Колонка Ручне вказує на ручне

тренування, а колонка AutoML – на використання розробленої системи автоматизованого тренування моделей машинного навчання. Як видно із цих таблиць, у середньому система автоматизованого тренування моделей машинного навчання дозволяє побудувати набагато точнішу модель, яку ще й набагато легше проаналізувати за рахунок побудови кастомних графіків.

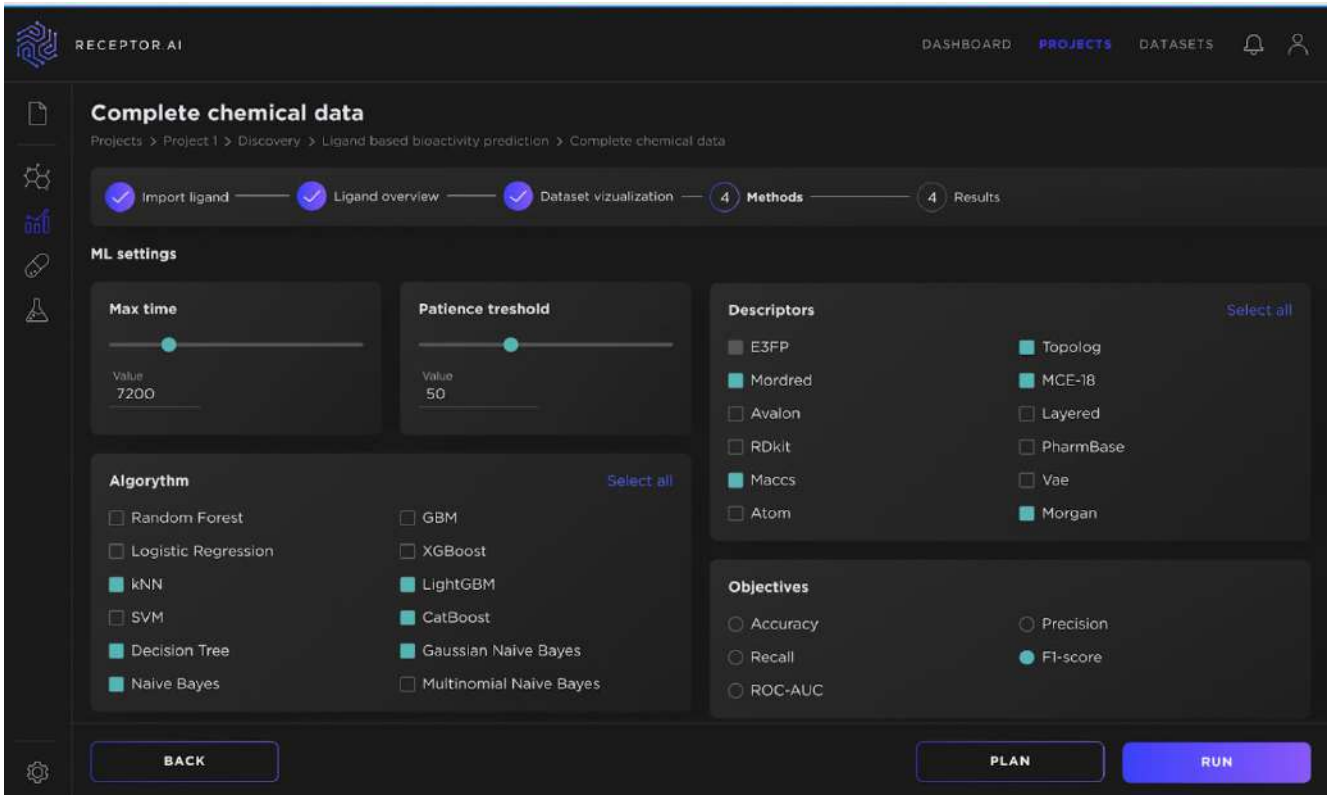


Рисунок 4.8 – Сторінка налаштування тренувального етапу пайплайну

Таблиця 4.1 – Результати тренування ADME-Tox моделей бінарної класифікації

Task	F1		Accuracy		Precision		Recall	
	Ручне	AutoML	Ручне	AutoML	Ручне	AutoML	Ручне	AutoML
Bioavailability	0.685	0.730	0.701	0.698	0.741	0.672	0.638	0.800
HIA	0.970	0.979	0.947	0.963	0.965	0.965	0.975	0.994
P-gp inhibitors	0.892	0.890	0.890	0.887	0.924	0.926	0.863	0.856

Кінець таблиці 4.1 – Результати тренування ADME-Tox моделей бінарної класифікації

P-gp substrates	0.800	0.830	0.800	0.834	0.803	0.855	0.797	0.806
BBB	0.901	0.902	0.911	0.912	0.885	0.893	0.918	0.912
AMES	0.843	0.841	0.826	0.821	0.843	0.827	0.844	0.856
DILI	0.622	0.774	0.626	0.732	0.721	0.733	0.547	0.820
Carcinogenicity ISF	0.627	0.716	0.827	0.868	0.650	0.746	0.605	0.698
Carcinogenicity OSF	0.541	0.697	0.715	0.790	0.625	0.713	0.476	0.683
Androgen antagonist	0.907	0.924	0.942	0.954	0.905	0.946	0.908	0.903
Androgen agonist	0.904	0.916	0.980	0.982	0.966	0.954	0.849	0.880
Androgen binding	0.921	0.936	0.906	0.924	0.932	0.949	0.910	0.924
Estrogen antagonist	0.875	0.898	0.946	0.956	0.867	0.879	0.882	0.918
Estrogen agonist	0.749	0.747	0.944	0.944	0.826	0.814	0.687	0.690
Estrogen binding	0.818	0.882	0.930	0.952	0.969	0.979	0.708	0.803
hERG	0.873	0.884	0.854	0.864	0.881	0.881	0.865	0.887
A549	0.799	0.798	0.818	0.821	0.803	0.822	0.796	0.775
HEK293	0.780	0.774	0.908	0.905	0.806	0.797	0.757	0.752
MCF7	0.778	0.778	0.803	0.808	0.773	0.790	0.784	0.767
Aromatase	0.880	0.891	0.947	0.951	0.875	0.875	0.885	0.907

Таблиця 4.2 – Результати тренування ADME-Tox моделей регресії

Task	R^2		MSE		MAE	
	Ручне	AutoML	Ручне	AutoML	Ручне	AutoML
Caco-2	0.693	0.741	0.199	0.152	0.371	0.322
PPB	0.725	0.662	287.093	352.117	12.307	13.560
VD	0.487	0.495	1.092	1.067	0.809	0.798
Plasma Clearance	0.240	0.330	1.564	1.325	0.938	0.873
Renal Clearance	0.352	0.450	3.982	3.534	1.387	1.241
Acute	0.457	0.525	2.037	1.779	1.068	1.010

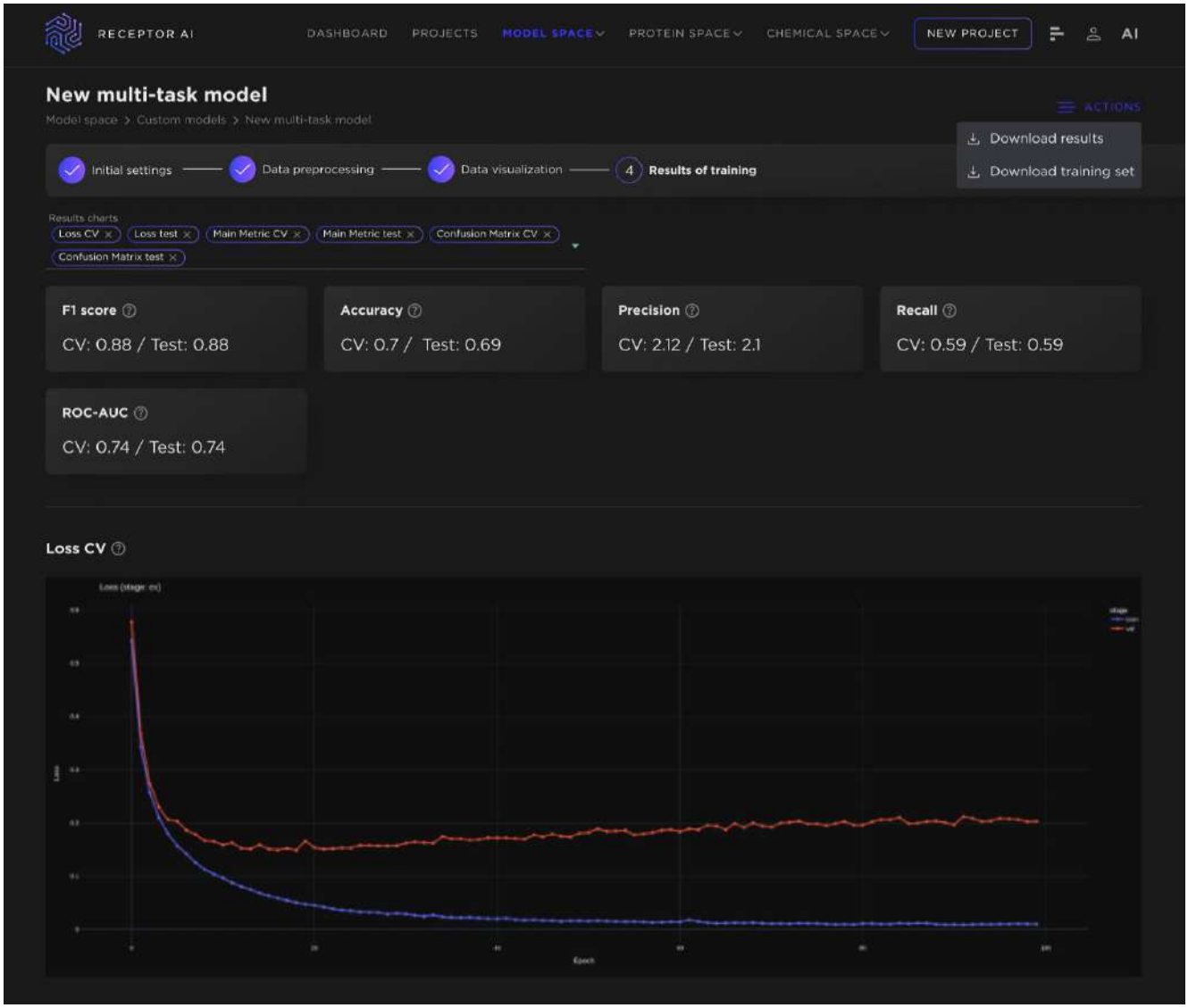


Рисунок 4.9 – Приклад результатів (метрики та графік лосів)

Рисунок 4.10 – Приклад результатів (розподіл цільової змінної та признаків)

4.4.1 Детальний аналіз бінарної класифікації (на прикладі BBB)

Проведемо детальний аналіз однієї моделі бінарної класифікації, а саме – BBB (ГЕБ).

Гематоенцефалічний бар'єр (ГЕБ або BBB) служить як регулятор та захисний механізм для центральної нервової системи (ЦНС), забезпечуючи необхідний гомеостаз та захищаючи її від токсинів, патогенів та інших шкідливих речовин. Однак деякі молекули ліків, за оцінками, можуть проникати через цей бар'єр, щоб досягти своєї мети. Тому оцінка проникності ГЕБ відіграє важливу роль у процесі розробки та відкриття нових лікарських засобів.

Побудована модель бінарної класифікації машинного навчання передбачає ймовірність проникності BBB. На рисунку 4.11 зображено гістограму розподілу таргет (цільової) змінної, значення якої ми прагнемо передбачати за допомогою натренованої моделі.

На рисунку 4.12 зображено зменшення розмірності признаків за допомогою PCA (Principal Component Analysis) методу, який довзоляє найбільш інформативно перевести признаки із N-вимірного у виміри, які можна проаналізувати людським оком. Для МЛ-задачі такі графіки є надзвичайно інформативними, оскільки можуть одразу ж продемонструвати нам лінійну або будь-яку іншу залежність для таргет змінної. І якщо така залежність є, то ми можемо бути впевнені, що можна побудувати модель, яка передбачатиме значення цієї таргет змінної із доволі високою точністю.

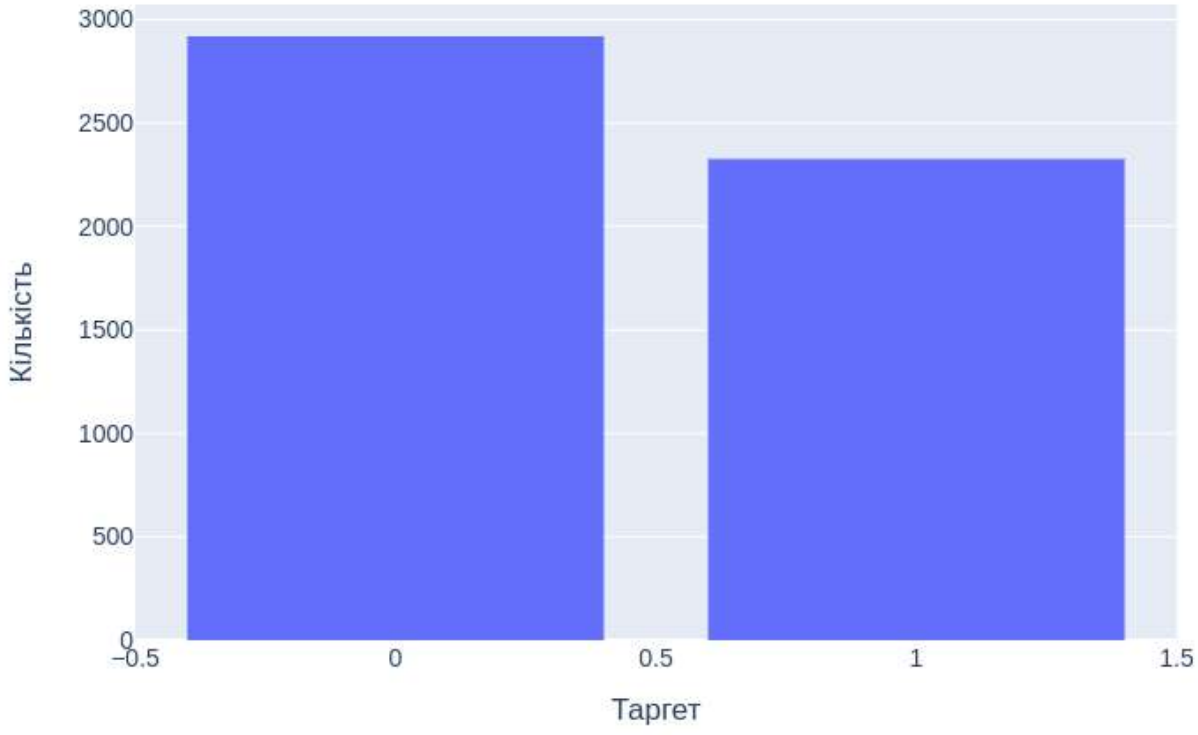


Рисунок 4.11 – Розподіл цільової змінної для BBB

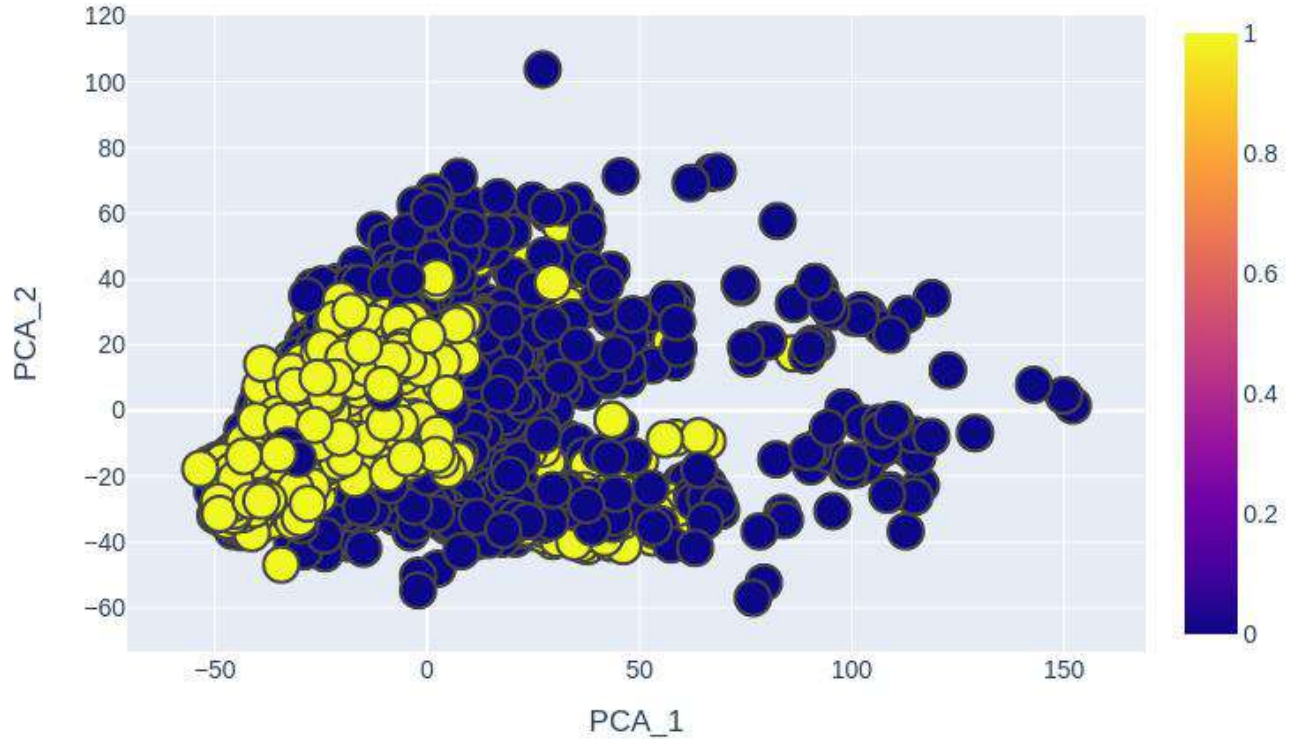


Рисунок 4.12 – Зменшення розмірності признаків у 2D простір для BBB

На рисунку 4.13 зображено графік втрат (або лосів) у залежності від номеру епохи для багатошарового перцептрона. На рисунках 4.14 та 4.15 зображено графіки $accuracy$ та $F1$ у залежності від номеру епохи.

На рисунку 4.16 зображено матрицю невідповідностей. За допомогою цієї матриці можна дуже легко встановити реальний перформанс моделі без додаткових метрик, аналізуючи кількість хибно передбачених семплів. Зокрема, для моделі BBB можна зробити висновок, що модель працює доволі непогано, оскільки к-сть хибно передбачених семплів класів 0 та 1 є невеликою.

4.4.2 Детальний аналіз регресії (на прикладі Caco-2)

Проведемо детальний аналіз однієї моделі регресії, а саме – Caco-2.

Клітинна лінія Caco-2 широко використовується як модель епітелію кишечника людини для оцінки проникності лікарських засобів in vivo, оскільки вона має схожість у морфологічних та функціональних характеристиках з ентероцитами людини.

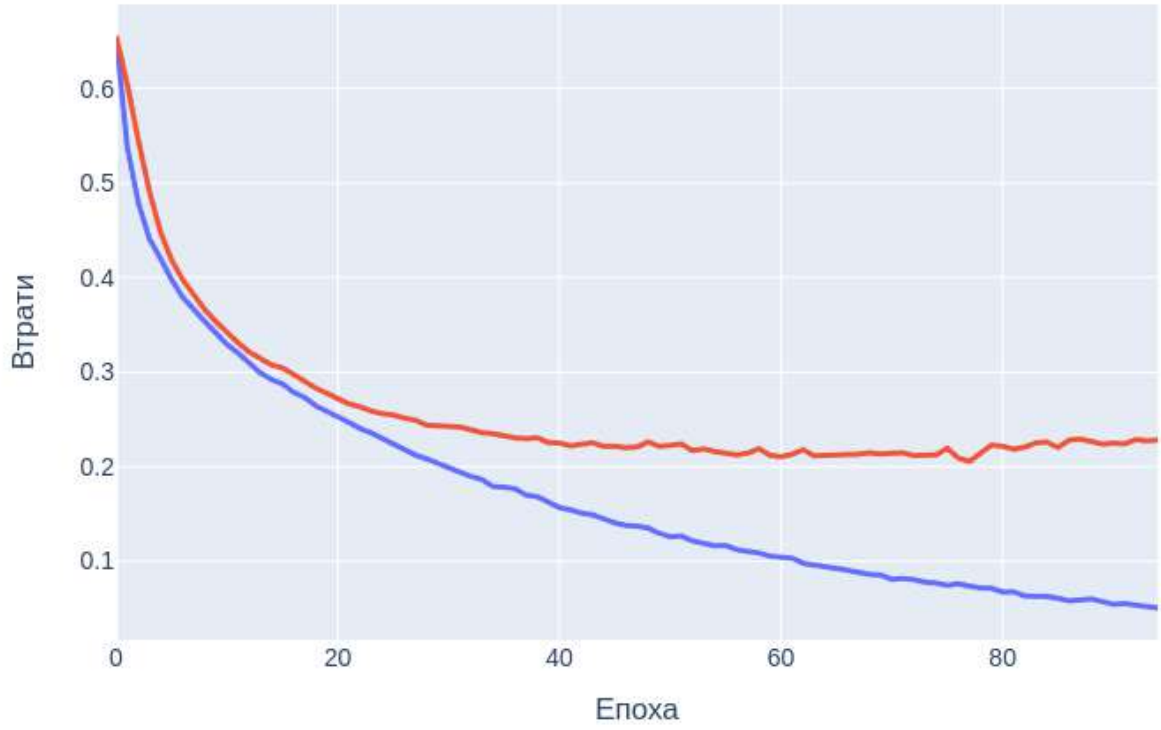


Рисунок 4.13 – Залежність втрат від номеру епохи для BBB. Синім зображено втрати для тренувального набору даних, червоним – для тестового

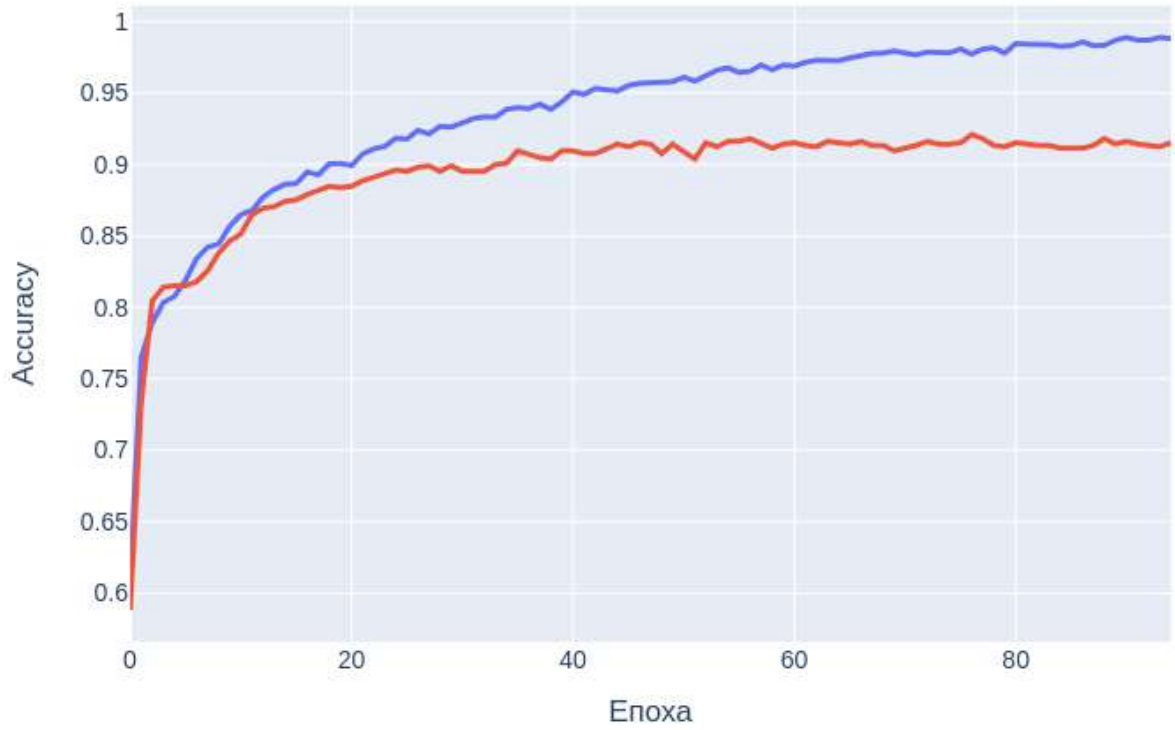


Рисунок 4.14 – Залежність $accuracy$ від номеру епохи для BBB. Синім зображено $accuracy$ для тренувального набору даних, червоним – для тестового

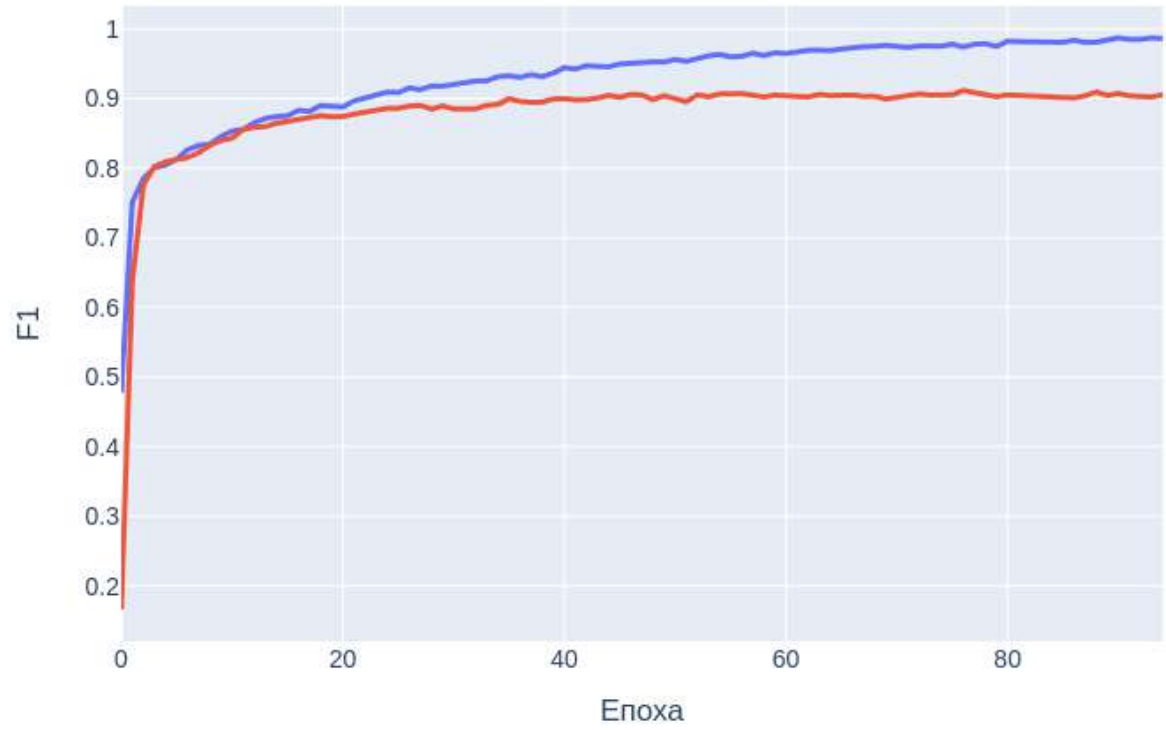


Рисунок 4.15 – Залежність $F1$ від номеру епохи для BBB. Синім зображено $F1$ для тренувального набору даних, червоним – для тестового

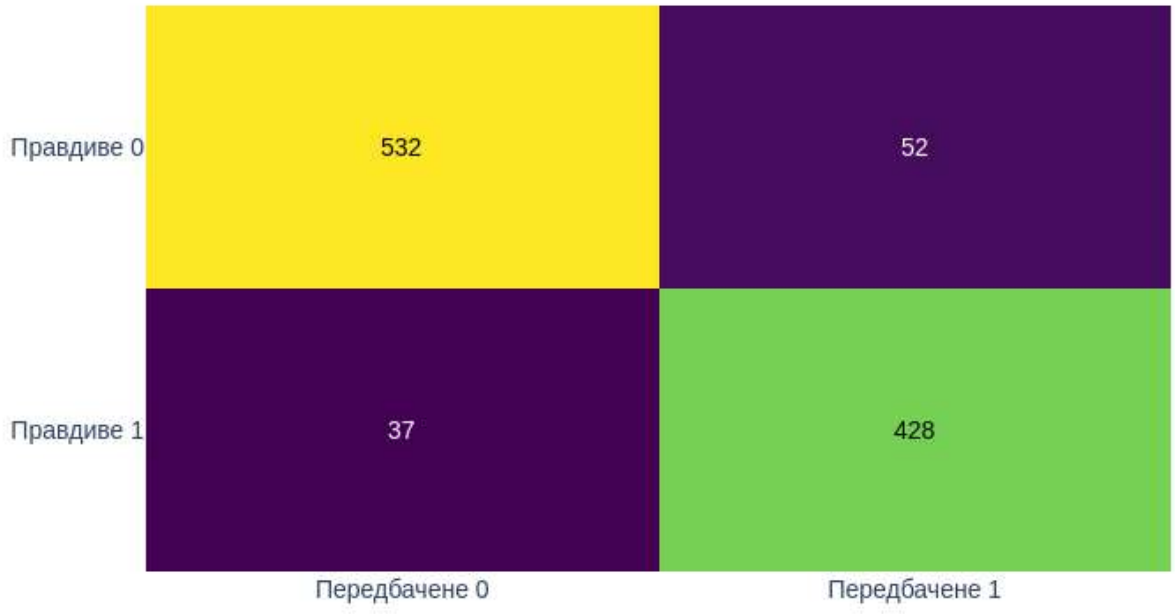


Рисунок 4.16 – Матриця невідповідностей для BBB

Побудована модель регресії машинного навчання передбачає проникність через кишечник людини Caco-2. На рисунку 4.17 зображено гістограму розподілу таргет (цільової) змінної, значення якої ми прагнемо передбачати за допомогою натренованої моделі.

На рисунку 4.18 зображено зменшення розмірності признаків за допомогою PCA методу.

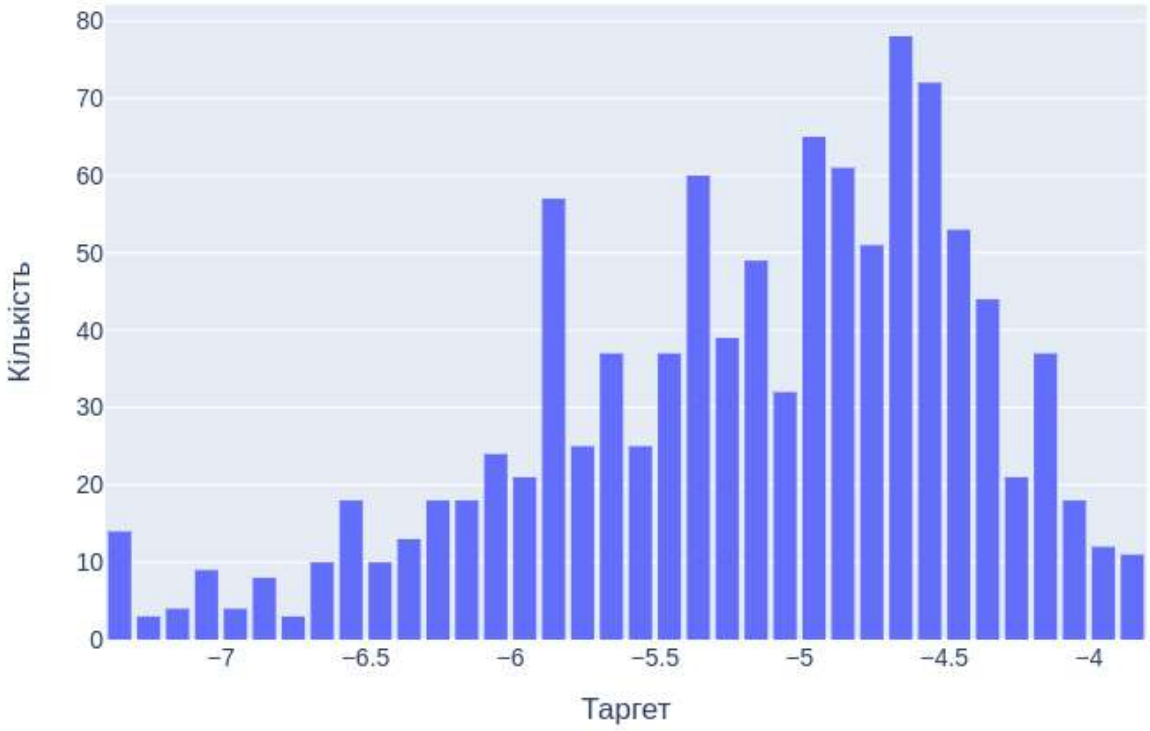


Рисунок 4.17 – Розподіл цільової змінної для Caco-2

На рисунку 4.19 зображено залежність між реальними й передбаченими значеннями. Такий графік дозволяє нам оцінити наскільки точними є передбачення моделі: чим ближча хмарка точок до прямої x=y, тим точніші є передбачення моделі. Проаналізувавши цей графік, можна зробити висновок, що наша модель регресії видає доволі точні передбачення Caco-2.

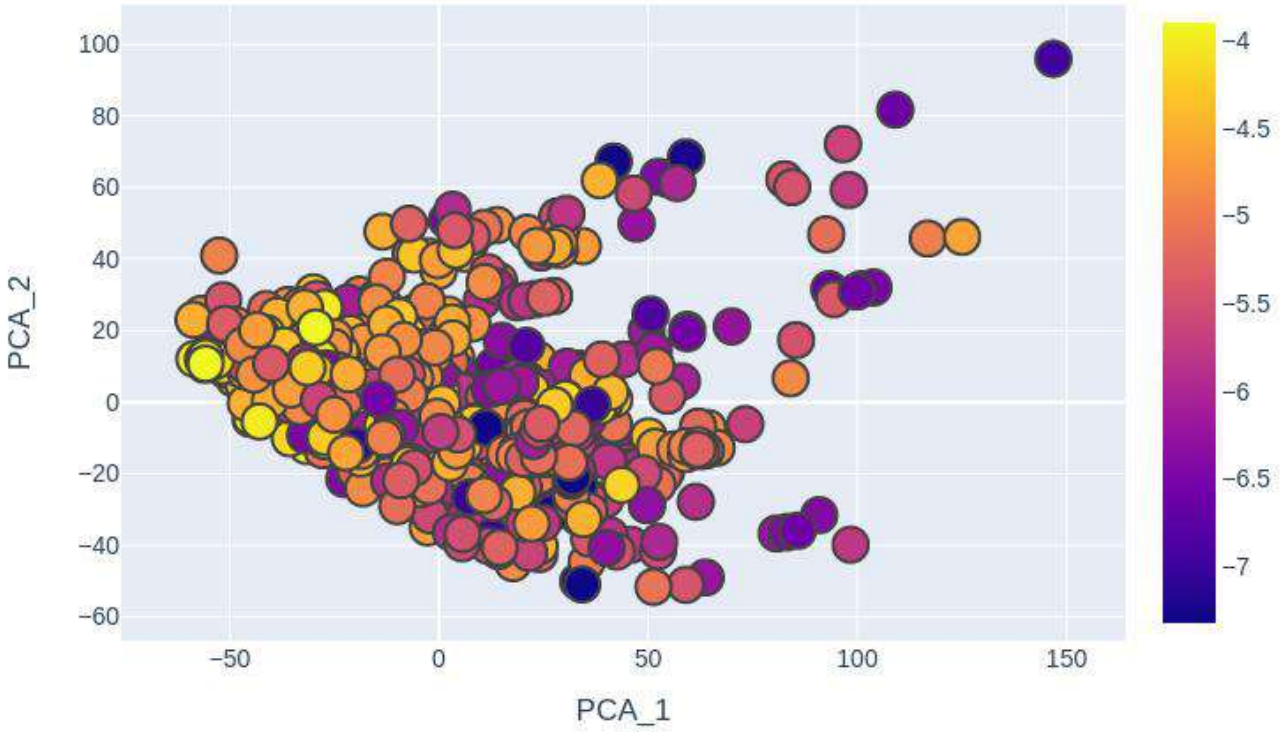


Рисунок 4.18 – Зменшення розмірності признаків у 2D простір для Caco-2

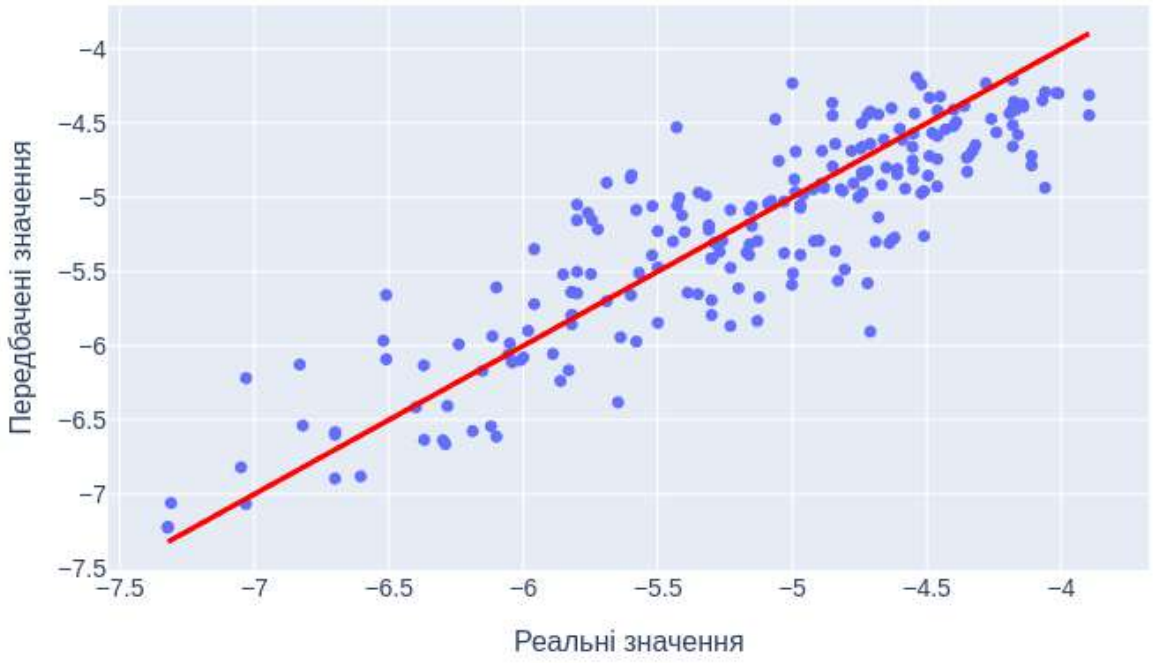


Рисунок 4.19 – Залежність між реальними й передбаченими значеннями для Caco-2. Червоним показано пряму x=y

4.5 Висновки

У даному розділі було описано SAAS платформу для розробки ліків. Також було проаналізовано експерименти засобами MLFlow і інтегровано AutoML пайплайн в інтерфейс SAAS платформи. Розділ завершується аналізом продуктивності розробленої системи на датасетах ADME-Tox, а саме BBB для бінарної класифікації та Caco-2 для регресії.

ВИСНОВКИ

На основі проведених досліджень розроблено архітектуру і компоненти програмного забезпечення, яке інтегровано в існуючу SAAS платформу з розробки ліків.

Практична значимість отриманих результатів полягає у можливості автоматичного тренування моделей машинного навчання на молекулярних датасетах і подальшого їх використання у SAAS платформі.

У першому розділі було проведено детальний огляд ключових складових, необхідних для AutoML пайплайну. Проаналізовано та порівняно різноманітні технології, що можуть бути використані для реалізації кожного елемента пайплайну. Окрім того, розглянуто різні методи представлення біологічних даних, придатних для використання у машинному навчанні. Розділ завершується формулюванням завдання та вибором відповідних технологій, що обґрунтовано на основі проведеного аналізу.

У другому розділі було проведено детальний огляд математичної моделі алгоритму вибору найкращих ознак, серед яких можна виділити фільтраційні, обгорткові та вбудовані методи. Також було наведено формулювання основних алгоритмів машинного навчання: лінійної та логістичної регресій, дерева рішень, випадкового лісу, одно- та багатошарових перцептронів, графової нейронної мережі, техніки машин опорних векторів (SVM) та підсилення градієнтного бустингу (XGBoost). Було розглянуто математичний опис процесу баєсівської оптимізації для тюнінгу гіперпараметрів моделі. Розділ завершується описом основних метрик для оцінки моделей бінарної класифікації та регресії.

У третьому розділі було досліджено вибір технологій для реалізації компонентів системи автоматизованого тренування та моніторингу моделей машинного навчання. Також було налаштовано роботу та взаємозв’язки компонентів системи і проведено тестування On-Premise провайдерів для побудови SAAS платформи. Розділ завершується описом On-Premise версії AutoML пайплайну за допомогою оркестратора AirFlow.

У четвертому розділі було описано SAAS платформу для розробки ліків. Також було проаналізовано експерименти засобами MLFlow і інтегровано AutoML пайплайн в інтерфейс SAAS платформи. Розділ завершується аналізом продуктивності розробленої системи на датасетах ADME-Tox, а саме BBB для бінарної класифікації та Caco-2 для регресії.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Vamathevan, J., Clark, D., Czodrowski, P., Dunham, I., Ferran, E., Lee, G. Applications of machine learning in drug discovery and development. *Nature Reviews Drug Discovery*. 2019. № 18. P. 463-477.

2. Turon, G., Hlozek, J., Woodland, J. G., Kumar, A., Chibale, K., Duran-Frigola, M. First fully-automated AI/ML virtual screening cascade implemented at a drug discovery centre in Africa. *Nature Communications*. 2023. № 14. P. 5736.

3. Liu, G., Lu, D., & Lu, J. An open-source, end-to-end automated machine learning package for clinical outcome prediction. *Pharmacometrics & systems pharmacology*. 2021. № 10. P. 478-488.

4. Ivanenkov, Y. , Polykovskiy, D., Bezrukov, D., Zagribelnyy, B., Aladinskiy, V., Kamya, P., Zhavoronkov, A. An AI-Driven platform for molecular design and optimization. Journal of Chemical Information and Modeling. *Chemistry42*. 2023. № 63. P. 695-701.

5. Koppad, S., Gkoutos, G. V., Acharjee A. Cloud computing enabled big multi-omics data analytics. *Bioinformatics and biology insights*. 2021. № 15. P. 117-122.

6. D'Agostino, D., Clematis, A., Quarati, A., Cesini, D., Chiappori, F., Milanesi, L., Merelli, I. Cloud infrastructures for in silico drug discovery: economic and practical aspects. *BioMed Research International*. 2018. P. 110-113.

7. Venugopal, Manu. Evolution of Digital Technologies and Use of Virtual Assistants in Drug Development. *Intelligent Systems for Healthcare Management and Delivery*. 2019. P. 1-20. URL: https://www.igi-global.com/chapter/evolution-of-digital-technologies-and-use-of-virtual-assistants-in-drug-development/218113

8. Caufield, John Harry, et al. Cardiovascular informatics: building a bridge to data harmony. *Cardiovascular Research*. 2022. № 118. P. 732-745. URL: https://academic.oup.com/cardiovascres/article/118/3/732/6159766

9. Dinov, Ivo D. Methodological challenges and analytic opportunities for modeling and interpreting Big Healthcare Data. *Gigascience 5.1*. 2016: № 16. P. 137-

142. URL: https://academic.oup.com/gigascience/article/5/1/s13742-016-0117-6/2720974

10. Karim, Md Rezaul, et al. Improving data workflow systems with cloud services and use of open data for bioinformatics research. *Briefings in bioinformatics*. 2018. № 19. P. 1035-1050.

11. Georgievskaya, Anastasia, et al. Artificial Intelligence Approaches for Skin Anti-aging and Skin Resilience Research. *Artificial Intelligence for Healthy Longevity*. 2023. P. 189-214.

12. Yang, Xiaoyu, et al. Cloud computing in e-Science: research challenges and opportunities. *The Journal of Supercomputing*. 2019. № 70. P. 408-464.

13. Langmead, Ben, and Abhinav Nellore. Cloud computing for genomic data analysis and collaboration. *Nature Reviews Genetics*. 2018. № 19. P. 208-219.

14. Thota, Chandu, and Daphne Lopez. Gunasekaran Manogaran. *HCI Challenges and Privacy Preservation in Big Data Security*. 2017. № 1. P. 56-62.

15. Llanes, Antonio, et al. Soft computing techniques for the protein folding problem on high performance computing architectures. *Current drug targets*. 2019. № 17. P. 1626-1648.

16. Delaney J. ESOL: Estimating aqueous solubility directly from molecular structure. *Journal of Chemical Information and Computer Sciences*. 2004. № 3. P. 1000-1005.

17. Eraso, Oscar Bolaños, Daniela Echeverri, Nikolas Donneys, Carolina Ameri, Tayebeh Perea. A present scenario of the computational approaches for ternary organic solar cells. *Journal of Renewable and Sustainable Energy*. 2023. № 15. P. 10-11. URL: https://www.researchgate.net/figure/Different-types-of-molecular-representations-are-applied-to-one-molecule-Reproduced-with_fig2_376150577

18. Gamo F., Sanz L., Vidal J., et al. Thousands of chemical starting points for antimalarial lead identification. *Journal of Nature*. 2010. № 7296. P. 305–310.

19. Glem R., Bender A., Arnby C., et. al. Circular fingerprints: flexible molecular descriptors with applications from physical chemistry to ADME. *Journal of Investigational Drugs*. 2006. № 3. P. 199–204.

20. Graves A., Wayne G., Danihelka I. Neural Turing machines. URL: https://arxiv.org/abs/1410.5401

21. Hachmann J., Olivares-Amaya R., Atahan-Evrenk S., et. al. The Harvard clean energy project: large-scale computational screening and design of organic photovoltaics on the world community grid. *The Journal of Physical Chemistry Letters*. 2011. № 17. P. 2241–2251.

22. Hershey J., Roux J., Weninger F. Deep unfolding: Model-based inspiration of novel deep architectures. URL: https://arxiv.org/abs/1409.2574

23. Hochreiter S., Schmidhuber J. Long short-term memory. *Journal of Neural computation*. 1997. № 8. P. 1735–1780.

24. Ioffe S., Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. URL: https://arxiv.org/abs/1502.03167

25. Kendall A., Gal Y., Cipolla R. Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. URL: https://arxiv.org/abs/1705.07115

26. Kingma D., Ba J. Adam: A method for stochastic optimization. URL: https://arxiv.org/abs/1412.6980

27. Lusci A., Pollastri G., and Baldi P. Deep architectures and deep learning in chemoinformatics: the prediction of aqueous solubility for drug-like molecules. *Journal of Chemical Information and Modelling*. 2013. № 7. P. 1563–1575.

28. Micheli A. Neural network for graphs: A contextual constructive approach. *Journal of Neural Networks*. 2009. № 3. P. 498–511.

29. Morgan H. The generation of a unique machine description for chemical structure. *Journal of Chemical Documentation*. 1965. № 2. P. 107–113.

30. Oliphant T. Python for scientific computing. *Journal of Computing in Science & Engineering*. 2007. № 3. P. 10–20.

31. RDKit: Open-source cheminformatics. URL: www.rdkit.org

32. RECEPTOR.AI Case Studies. URL: https://receptor.ai/case-studies

33. Ramsundar B., Kearnes S., Riley P., Webster D., et. al. Massively multitask networks for drug discovery. URL: https://arxiv.org/abs/1502.02072

34. Rogers R., Hahn M. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*. 2010. № 5. P. 742–754.

35. Tai K., Socher R., Manning C. Improved semantic representations from tree-structured long short-term memory networks. URL: https://arxiv.org/abs/1503.00075

36. Unterthiner T., Mayr A., Klambauer G., Hochreiter S. Toxicity prediction using deep learning. URL: https://arxiv.org/abs/1503.01445

37. Zaheer M., Kottur S., Ravanbakhsh S, et. al. Deep Sets. URL: https://arxiv.org/abs/1703.06114

38. S. J. Kim, K. Koh, M. Lustig, S. Boyd and D. Gorinevsky. An Interior-Point Method for Large-Scale L1-Regularized Least Squares. 2007. URL: https://web.stanford.edu/~boyd/papers/pdf/l1_ls.pdf

39. Zou, Hui, Trevor Hastie, and Robert Tibshirani. On the "degrees of freedom" of the lasso. 2007. URL: https://arxiv.org/abs/0712.0881

40. Michael E. Tipping. Sparse Bayesian Learning and the Relevance Vector Machine. 2001. URL: https://www.jmlr.org/papers/volume1/tipping01a/tipping01a.pdf

41. Aaron Defazio, Francis Bach, Simon Lacoste-Julien. SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives. 2014. URL: https://arxiv.org/abs/1407.0202

42. Noah Simon, Jerome Friedman, Trevor Hastie. A Blockwise Descent Algorithm for Group-penalized Multiresponse and Multinomial Regression. 2013. URL: https://arxiv.org/abs/1311.6529

43. John C. Platt. Probabilistic outputs for SVMs and comparisons to regularized likelihood methods. 1999. URL: https://home.cs.colorado.edu/~mozer/Teaching/syllabi/6622/papers/Platt1999.pdf

44. Wu, Lin and Weng. Probability estimates for multi-class classification by pairwise coupling. 2004. URL: https://www.csie.ntu.edu.tw/~cjlin/papers/svmprob/svmprob.pdf

45. Fan, Rong-En, et al. LIBLINEAR: A library for large linear classification. *Journal of machine learning research*. 2008. P. 1871-1874.

46. Chang and Lin. LIBSVM: A Library for Support Vector Machines. URL: https://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf

47. M. Mayer, S.C. Bourassa, M. Hoesli, and D.F. Scognamiglio. Machine Learning Applications to Land and Structure Valuation. *Journal of Risk and Financial Management*. 2022. № 5. P. 193.

48. Tianqi Chen, Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. 2016. URL: https://arxiv.org/abs/1603.02754

49. Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, Tie-Yan Liu. 2017. URL: https://papers.nips.cc/paper_files/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html

50. Friedman, J.H. Greedy function approximation: A gradient boosting machine. 2001. *Annals of Statistics*. № 29. P. 1189-1232.

51. Friedman, J.H. Stochastic gradient boosting. *Computational Statistics & Data Analysis*. 2002. № 38. P. 367-378.

52. G. Ridgeway. Generalized Boosted Models: A guide to the gbm package. 2024. URL: https://cran.r-project.org/web/packages/gbm/vignettes/gbm.pdf

53. L. Breiman. Random Forests. *Machine Learning*. 2001. № 45. P. 5-32.

54. L. Breiman. Arcing Classifiers. Annals of Statistics. 1998. № 53. P. 189-201.

55. P. Geurts, D. Ernst., and L. Wehenkel. Extremely randomized trees. *Machine Learning*. 2006. № 63. P. 3-42.

56. L. Breiman. Pasting small votes for classification in large databases and on-line. *Machine Learning*. 1999. № 36. P. 85-103.

57. L. Breiman. Bagging predictors. Machine Learning. 1996. № 24. P. 123-140.

58. T. Ho. The random subspace method for constructing decision forests. *Pattern Analysis and Machine Intelligence*. 1998. № 20. P. 832-844.

59. G. Louppe and P. Geurts. Ensembles on Random Patches. *Machine Learning and Knowledge Discovery in Databases*. 2012. P. 346-361.

60. Wolpert, David H. Stacked generalization. *Neural networks*. 1992. № 5. P. 241-259.

61. Richard G. Baraniuk. Compressive Sensing. 2007. URL: http://users.isr.ist.utl.pt/~aguiar/CS_notes.pdf

ДОДАТОК А
(обов'язковий)

ЛІСТИНГ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Модуль «Реалізація пайплайну для On-Premise деплойменту за допомогою оркестратора Airflow».

```
from datetime import timedelta

# The DAG object; we'll need this to instantiate a DAG
from airflow import DAG

# Operators; we need this to operate!
from airflow.operators.dummy import DummyOperator
from airflow.operators.bash import BashOperator
from airflow.operators.python import PythonOperator
from airflow.providers.docker.operators.docker import DockerOperator
from airflow.utils.dates import days_ago
from airflow.models import Variable

default_args = {
    'owner': 'admin',
    'depends_on_past': False,
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 0,
    'retry_delay': timedelta(minutes=5),

}
with DAG(
        'AutoML_pipeline_base_receptor_1_5',
        default_args=default_args,
        description='A simple tutorial DAG',
        start_date=days_ago(0),
```

```
        tags=['example'],
        render_template_as_native_obj=True,
) as dag:
    init_mlflow_experiment = DockerOperator(
        task_id='init_mlflow_experiment',

image='amazonaws.com/pipelines/general_blocks/init_mlflow_experiment
:latest',
        auto_remove=True,
         queue='automl_1_5',
        environment={
            'MLFLOW_DB_URL': '{{ dag_run.conf.MLFLOW_DB_URL }}',
            'MLFLOW_BUCKET': '{{ dag_run.conf.MLFLOW_BUCKET }}',
            'EXPERIMENT_NAME': '{{ dag_run.conf.EXPERIMENT_NAME }}',
            'RUN_NAME': '{{ dag_run.conf.RUN_NAME }}'
        }
    )

    generate_descriptors = DockerOperator(
        task_id='generate_descriptors',

image='amazonaws.com/pipelines/automl/generate_descriptors:latest',
        auto_remove=True,
        retries=3,
         queue='automl_1_5',
        environment={
            'EXPERIMENT_NAME': '{{ dag_run.conf.EXPERIMENT_NAME }}',
            'RUN_NAME': '{{ dag_run.conf.RUN_NAME }}',
            'DATASETS_BUCKET': '{{ dag_run.conf.DATASETS_BUCKET }}',
            'DATASET_FOLDER': '{{ dag_run.conf.DATASET_FOLDER }}',
            'DATASET_FILE': '{{ dag_run.conf.DATASET_FILE }}',
            'DESCRIPTORS_FILE': '{{ dag_run.conf.DESCRIPTORS_FILE
}}',
            'DESCRIPTORS_TYPES': '{{ dag_run.conf.DESCRIPTORS_TYPES
}}',
```

```
            'MLFLOW_DB_URL': '{{ dag_run.conf.MLFLOW_DB_URL }}',
                    }
    )

    select_descriptors = DockerOperator(
        task_id='select_descriptors',

image='amazonaws.com/pipelines/automl/select_descriptors:latest',
        auto_remove=True,
        retries=3,
         queue='automl_1_5',
        environment={
            'EXPERIMENT_NAME': '{{ dag_run.conf.EXPERIMENT_NAME }}',
            'RUN_NAME': '{{ dag_run.conf.RUN_NAME }}',
            'DATASETS_BUCKET': '{{ dag_run.conf.DATASETS_BUCKET }}',
            'DATASET_FOLDER': '{{ dag_run.conf.DATASET_FOLDER }}',
            'DESCRIPTORS_FILE': '{{ dag_run.conf.DESCRIPTORS_FILE
}}',
            'TARGET_FILE': '{{ dag_run.conf.TARGET_FILE }}',
            'SELECTED_DESCRIPTORS_FILE': '{{
dag_run.conf.SELECTED_DESCRIPTORS_FILE }}',
            'MLFLOW_DB_URL': '{{ dag_run.conf.MLFLOW_DB_URL }}',
        }
    )

    automl_base_task_list = [DockerOperator(
        task_id=f'automl_base_{i}',
        image='amazonaws.com/pipelines/automl/automl:latest',
        auto_remove=True,
        retries=3,
         queue='automl_1_5',
        environment={
            'EXPERIMENT_NAME': '{{ dag_run.conf.EXPERIMENT_NAME }}',
            'RUN_NAME': '{{ dag_run.conf.RUN_NAME }}',
            'DATASETS_BUCKET': '{{ dag_run.conf.DATASETS_BUCKET }}',
```

```
            'DATASET_FOLDER': '{{ dag_run.conf.DATASET_FOLDER }}',
            'DESCRIPTORS_FILE': '{{ dag_run.conf.DESCRIPTORS_FILE
}}',
            'SELECTED_DESCRIPTORS_FILE': '{{
dag_run.conf.SELECTED_DESCRIPTORS_FILE }}',
            'TARGET_FILE': '{{ dag_run.conf.TARGET_FILE }}',
            'PROBLEM_TYPE': '{{ dag_run.conf.PROBLEM_TYPE }}',
            'MODEL_FAMILIES': '{{ dag_run.conf.MODEL_FAMILIES }}',
            'OPTIMIZATION_METRIC': '{{
dag_run.conf.OPTIMIZATION_METRIC }}',
            "SCORE_THRESHOLD": "{{ dag_run.conf.SCORE_THRESHOLD }}",
            'TIMEOUT_BASE': '{{ dag_run.conf.TIMEOUT_BASE }}',
            'TIMEOUT_META': '{{ dag_run.conf.TIMEOUT_META }}',
            'LEVEL': 'base',
            'OPTUNA_DB_URL': '{{ dag_run.conf.OPTUNA_DB_URL }}',
        }
    ) for i in range(1, int(Variable.get('N_workers')) + 1)]

      log_automl_to_mlflow = DockerOperator(
        task_id='log_automl_to_mlflow',

image='amazonaws.com/pipelines/automl/log_automl_to_mlflow:latest',
        auto_remove=True,
        retries=3,
         queue='automl_1_5',
        environment={
            'EXPERIMENT_NAME': '{{ dag_run.conf.EXPERIMENT_NAME }}',
            'RUN_NAME': '{{ dag_run.conf.RUN_NAME }}',
            'MLFLOW_BUCKET': '{{ dag_run.conf.MLFLOW_BUCKET }}',
            'DATASETS_BUCKET': '{{ dag_run.conf.DATASETS_BUCKET }}',
            'DATASET_FOLDER': '{{ dag_run.conf.DATASET_FOLDER }}',
            'DESCRIPTORS_FILE': '{{ dag_run.conf.DESCRIPTORS_FILE
}}',
            'SELECTED_DESCRIPTORS_FILE': '{{
dag_run.conf.SELECTED_DESCRIPTORS_FILE }}',
```

```
            'TARGET_FILE': '{{ dag_run.conf.TARGET_FILE }}',
            'PROBLEM_TYPE': '{{ dag_run.conf.PROBLEM_TYPE }}',
            'MODEL_FAMILIES': '{{ dag_run.conf.MODEL_FAMILIES }}',
            'OPTIMIZATION_METRIC': '{{
dag_run.conf.OPTIMIZATION_METRIC }}',
            "SCORE_THRESHOLD": "{{ dag_run.conf.SCORE_THRESHOLD }}",
            'OPTUNA_DB_URL': '{{ dag_run.conf.OPTUNA_DB_URL }}',
            'MLFLOW_DB_URL': '{{ dag_run.conf.MLFLOW_DB_URL }}',
        }
    )

    init_mlflow_experiment >> generate_descriptors

    generate_descriptors >> select_descriptors

    [select_descriptors >> t3_i for t3_i in automl_base_task_list]

    [t3_i >> log_automl_to_mlflow for t3_i in automl_base_task_list]
```

ДОДАТОК Б
(обов'язковий)

КОПІЯ ПУБЛІКАЦІЇ У ФАХОВОМУ НАУКОВОМУ ВИДАННІ

RSC Advances

ROYAL SOCIETY OF CHEMISTRY

PAPER

View Article Online
View Journal | View Issue

Check for updates

Cite this: RSC Adv., 2023, 13, 10261

3DProtDTA: a deep learning model for drug-target affinity prediction based on residue-level protein graphs†

Taras Voitsitskyi, *[ac] Roman Stratiichuk,[ad] Ihor Koleiev,[a] Leonid Popryho,[a] Zakhar Ostrovsky,[a] Pavlo Henitsoi,[a] Ivan Khropachov,[a] Volodymyr Vozniak,[a] Roman Zhytar,[a] Diana Nechepurenko,[a] Semen Yesylevskyy, [abc] Alan Nafiiev[a] and Serhii Starosyla [a]

Received 14th January 2023
Accepted 26th March 2023

DOI: 10.1039/d3ra00281k

rsc.li/rsc-advances

Accurate prediction of the drug-target affinity (DTA) *in silico* is of critical importance for modern drug discovery. Computational methods of DTA prediction, applied in the early stages of drug development, are able to speed it up and cut its cost significantly. A wide range of approaches based on machine learning were recently proposed for DTA assessment. The most promising of them are based on deep learning techniques and graph neural networks to encode molecular structures. The recent breakthrough in protein structure prediction made by AlphaFold made an unprecedented amount of proteins without experimentally defined structures accessible for computational DTA prediction. In this work, we propose a new deep learning DTA model 3DProtDTA, which utilises AlphaFold structure predictions in conjunction with the graph representation of proteins. The model is superior to its rivals on common benchmarking datasets and has potential for further improvement.

Introduction

Modern drug discovery remains a painfully slow and expensive process despite all the recent scientific and technological advancements. It usually takes several years and the estimated cost of developing a new drug may run over a billion US dollars.[1] More than 30% of all drugs entering phase II of clinical trials and above 58% of drugs entering phase III fail.[2] It was reported that among 108 new and repurposed drugs, reported as phase II failures between 2008 and 2010, 51% were due to insufficient efficacy.[3] This observation highlighted the need for novel *in silico* techniques that can decrease the failure rate by filtering out compounds with low predicted efficacy in the early stages of the drug discovery pipeline. In this regard, the computational methods that assess drug-target binding affinities (DTA) are of great interest[4] because DTA is generally considered one of the best predictors of resulting drug efficacy. Accurate prediction of the DTA is of critical importance for filtering out inefficient molecules and preventing them from reaching clinical trials, thus a multitude of computational DTA techniques have been developed in recent years.

The most accurate computational estimate of DTA could be obtained from atomistic molecular dynamics simulations (either classical, quantum or hybrid) combined with one of the modern techniques of computing the free energy of ligand binding.[5] However, accuracy comes at the cost of very high computational demands, which makes these methods generally impractical for large-scale virtual screening.

That is why the common method of choice for estimating DTA in modern drug discovery is molecular docking, which provides a reasonable compromise between accuracy and computational efficiency.[6] However, it is generally believed that empirical scoring functions used in molecular docking have already approached the practical limit of accuracy, which is unlikely to be improved without introducing an additional computational burden.

In order to address these drawbacks the classical machine learning (ML) methods for determining DTA were developed. These methods do not depend on computing physical interactions between the target protein and the ligand. They are purely knowledge-based and rely on the idea that similar ligands tend

[a]Receptor.AI Inc., 20-22 Wenlock Road, London N1 7GU, UK. E-mail: taras27060@gmail.com

[b]Institute of Organic Chemistry and Biochemistry, Czech Academy of Sciences, CZ-166 10 Prague 6, Czech Republic

[c]Department of Physics of Biological Systems, Institute of Physics of The National Academy of Sciences of Ukraine, Nauky Ave. 46, 03028, Kyiv, Ukraine

[d]Department of Biophysics and Medical Informatics, Educational and Scientific Centre "Institute of Biology and Medicine", Taras Shevchenko National University of Kyiv, 64 Volodymyrska Str., 01601 Kyiv, Ukraine

† Electronic supplementary information (ESI) available: The best model architectures, tuned hyperparameters for the benchmark, detailed data on manual cross-validation experiments, InterPro entries statistics in the datasets, and protein residue-level graphs similarity between AlphaFold and experimental structures. See DOI: https://doi.org/10.1039/d3ra00281k

to interact with similar protein targets, which are encoded into the pre-trained neural networks. Applying such models to any given ligand is blazingly fast, which allows using them for screening the numbers of compounds, which are unreachable for molecular docking or MD simulations. However, even the most successful methods from this category, such as KronRLS[7] and SimBoost,[8] were recently outperformed by more modern deep learning (DL) techniques.

Seminal DL methods of assessing DTA used string representation of the target's amino acid sequence and a simplified linear representation of the ligands using SMILES (molecular-input line-entry system), which were subsequently encoded by 1D convolutional neural networks (CNNs) and/or long-short-term memory (LSTM) blocks.[9,10] The same linear representations were shown to be efficient for DTA prediction in combination with generative adversarial networks (GANs).[11] However, it is obvious that linear representations lead to a huge loss of information and keeping the knowledge about connectivity and 3D arrangements of both the target protein and the ligands could improve the results. Indeed, the introduction of graph neural networks (GNNs), which preserve information about connectivity and the 3D arrangement of atoms, improved the scores of the models on most of the benchmarking DTA datasets.[12]

In contrast to the sequence-based techniques, the performance of the GNNs depends strongly on the availability and accuracy of 3D structures of target proteins. The scarcity of such structures limited the size of the training datasets and hampered the progress of GNN-based DTA models. The huge success of AlphaFold[13] in protein structure prediction opens new opportunities for developing better DTA prediction models. Accurately predicted 3D structure of druggable protein domains that don't have experimentally resolved structures, adds unprecedented amounts of data for training ML models and improving their performance.

In this article, we proposed a new method of constructing efficient residue-level protein graphs based on the target's 3D structure predicted by AlphaFold and selected the best GNN architectures for this kind of data. This resulted in a new deep-learning model for predicting drug-target affinities: 3DProtDTA. When applied to common benchmark datasets our model is superior to its rivals on all evaluated metrics. The perspectives of applications and further improvements of our model are discussed.

Materials and methods

Datasets

We evaluated our approach on two widespread benchmark datasets for DTA prediction: Davis[14] and KIBA.[15]

The Davis dataset contains the pairs of kinase proteins and their respective inhibitors with experimentally determined dissociation constant (K_d) values. K_d values were transformed by eqn (1) and transformed scores were used as labels for benchmarking in the same way as in the baseline approaches. There are 442 proteins and 68 ligands in this dataset.

$$pK_d = -\log_{10}\left(\frac{K_d}{1e9}\right) \quad (1)$$

The KIBA dataset comprises scores originating from an approach called KIBA, in which inhibitor bioactivities from different sources such as K_i, K_d and IC_{50} are combined. The KIBA scores were pre-processed by the SimBoost algorithm[8] and the final values were used as labels for model training. Initially, the KIBA dataset contained 467 proteins and 52 498 ligands. For benchmarking purposes, the same authors[8] filtered the dataset to keep only the drugs and targets with at least 10 samples resulting in 229 unique proteins and 2111 unique ligands.

The numbers of affinity scores and unique entries in the datasets are summarised in Table 1.

We used isomorphic SMILES strings for both datasets and UniProt[16] accession codes for the KIBA dataset, provided by DeepDTA,[10] as initial entries representation. For the Davis

Table 1 Summary of the benchmark datasets

Dataset	Proteins	Ligands	Samples
Davis	442	68	30 056
KIBA	229	2111	118 254

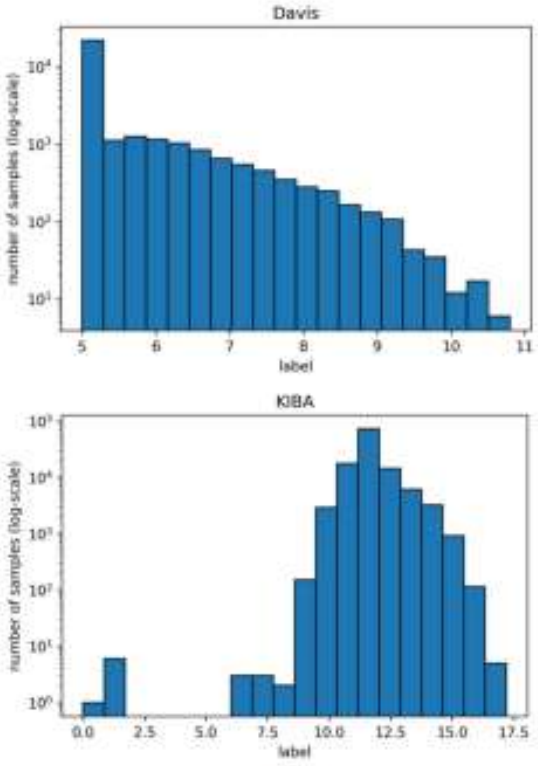


Fig. 1 Distribution of Davis (top) and KIBA (bottom) labels used directly in benchmarking (note that the y-axis is log-scale).

dataset, proteins with the same UniProt accession codes may represent different entries. Therefore, unique identifiers were assigned to each target entry annotated by UniProt accession code, mutation type, phosphorylation status, proteins in a complex, and protein domains. Fig. 1 shows distributions of labels for Davis (a) and KIBA (b) datasets.

Ligand representation

We utilised modified molecular graphs, initially proposed in the approach for drug property prediction Chemi-Net[17] along with the standard Morgan fingerprints[18] to represent ligands for DTA prediction.

Python API of an open-source cheminformatics package RDKit v. 2021.03 was used to generate both ligand representations based on isomorphic SMILES. We calculated 1024 bit vectors of classical Morgan fingerprints with radius 2 and 1024-bit vectors of feature-based fingerprint invariants[19] with the same radius. Both vectors were concatenated into a single 2048-bit vector.

The graphs for ligands were generated on the atomistic level (one node in the graph is one heavy atom in a ligand). Fig. 2 shows distributions of the number of heavy atoms in the used ligands, while Table 2 shows the features for a molecular graph representation of the ligands.

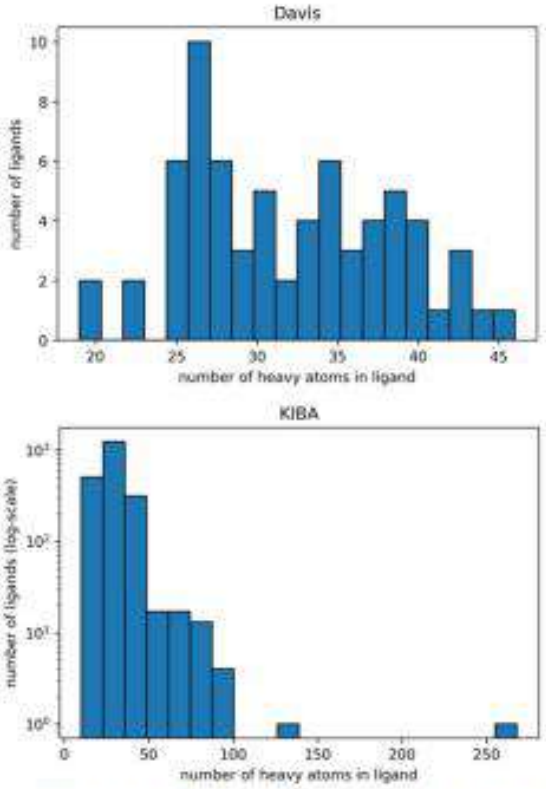


Fig. 2 Distribution of the number of heavy atoms in the ligands in Davis (top) and KIBA (bottom) datasets. The Y axis is log-transformed for the KIBA dataset.

Table 2 Atomic-level graph features for ligand representation

Name	Size
Node features	
One-hot encoded atom type (C, N, O, F, S, Cl, Br, P, I)	9
Atom mass (scaled by min–max)	1
Number of directly bonded atom neighbours (scaled by min–max)	1
Total number of bonded hydrogens (scaled by min–max)	1
One-hot encoded atom hybridization (sp^2, sp^3)	2
Is atom in a ring (1 – yes, 0 – no)	1
Is atom aromatic (1 – yes, 0 – no)	1
Is atom hydrophobic (1 – yes, 0 – no)	1
Is atom metal (1 – yes, 0 – no)	1
Is atom halogen (1 – yes, 0 – no)	1
Is atom donor (1 – yes, 0 – no)	1
Is atom acceptor (1 – yes, 0 – no)	1
Is atom positively charged (1 – yes, 0 – no)	1
Is atom negatively charged (1 – yes, 0 – no)	1
Edge features	
One-hot encoded bond type (single, double, triple, aromatic)	4
Bond is conjugated (1 – yes, 0 – no)	1
Bond is in the ring (1 – yes, 0 – no)	1

The Morgan fingerprints and ligand graphs are available in the GitHub repository. The order of the features in the graphs is the same as in Table 2.

Protein representation

We have developed the residue-level protein graph based on 3D protein structures generated by AlphaFold.[13] Approximately 50% of the proteins in both datasets have known 3D structures deposited in the Protein Data Bank but we decided to use AlphaFold predictions for all proteins to make our approach unified and to avoid additional tedious pre-processing of experimentally determined structures, which are often incomplete, contain irrelevant crystallographic ligands, *etc.*

For the KIBA dataset, all the structures were obtained from the AlphaFold protein structure database (**https://alphafold.ebi.ac.uk/**). For the Davis dataset only single protein entries without mutations were downloaded from the AlphaFold database. For the rest of the entries, 3D structures were generated manually using an accelerated implementation of the AlphaFold algorithm in Google Colaboratory: ColabFold.[20]

To avoid undesirable noise from the parts of proteins, which have weak or no relation to the ligand binding, we have parsed domain annotations from UniProt[16] to determine the ligand binding sites. Both datasets contain only the kinase enzymes and the ligands with kinase inhibitor activity. Consequently, only the domains with known kinase activity or related to the kinase activity (annotated by UniProt as a protein kinase, histidine kinase, PI3K/PI4K, PIPK, AGC-kinase, or CBS) were kept in the protein structures.

This preprocessing step not only decreased the noise in the data but also eliminated most residues with a low per-residue confidence score (pLDDT). In AlphaFold a pLDDT is

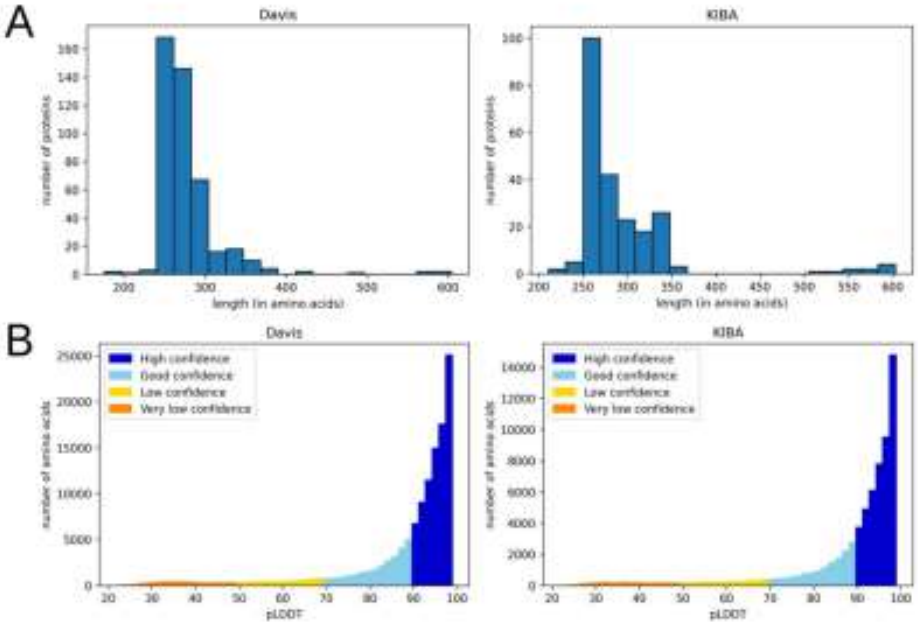


Fig. 3 Distribution of the number of amino acids (A) and pLDDT scores (B) in processed 3D protein structures (after removal of the parts not related to kinase activity) for Davis (left) and KIBA (right) datasets.

a continuous scale from 0–100 (higher is better), which shows the quality of structure prediction. pLDDT lower than 70 emerges in predicted 3D structures if they are unstructured in physiological conditions or the amino acid sequence has low alignment depth13. The regions with such low scores should be treated with caution. Since the domains related to kinase activity are mostly well studied and available in databases of experimental protein structures, keeping only them and removing other regions improves the average pLDDT score. Fig. 3 shows the number of amino acids in processed PDB files (A) as well as the distribution of pLDDT scores (B).

Filtered 3D structures were converted into the residue-level graphs using Biopython v. 1.79 (ref. 21) and Pteros.[22,23] Inspired by the Open Drug Discovery Toolkit,[24] we have developed an approach for encoding protein properties in the graph edge features. An edge was created if two amino acids form an either covalent bond or a non-covalent contact within a particular distance cutoff. The edge features define the type of this connectivity (Table 3). This technique allowed reducing the size of the protein graph in terms of the number of edges compared to the conventional protein graph generation approaches that define the same distance threshold for all types of residue–residue interactions.

Table 3 Bond types and corresponding distance cutoffs used for graph generation and assignment of edge features

Bond type	Distance cutoff (Å)
Covalent	—
Hydrophobic contacts	4
Hydrogen bond	3.5
Salt bridge	4
Cation–pi interaction	5
Perpendicular pi-stacking	5
Parallel pi-stacking	5

Such a graph is directed because of the hydrogen bonds, salt bridges, and cation–pi interactions, which are non-commutative and require specification of the roles for each of involved residues. For example, in the edge created by the hydrogen bond between nodes a and b, it is important to identify which node is a donor and which one is an acceptor. Thus, edge a–b is assigned edge features that are different from those of edge b–a. Similarly, the salt bridges require distinguishing between cationic residue and anionic residue nodes and the cation–pi interactions – between cationic and aromatic residues. In contrast, covalent bonds, pi-stacking and hydrophobic contacts are symmetric.

For each of the 20 standard amino acid types, we assigned seven characteristics AAPHY7 (ref. 25) and 23 BLOSUM62 values according to alignments of homologous protein sequences[26] provided by GraphSol.[27] In addition, the structure-dependent and sequence-dependent node and edge features were used (Table 4).

As the two last node features for the protein graphs we added phosphorylation status and mutation status. Each of the two features is either 1 (phosphorylated/mutated) or 0 (non-phosphorylated/non-mutated) and is the same for each node

Table 4 Residue-level node features for protein graph representation

Name	Size
Node features	
Solvent-accessible surface area (scaled by mean and standard deviation)	1
Phi angle (in degrees, divided by 180)	1
Psi angle (in degrees, divided by 180)	1
One-hot encoded belonging to secondary structure (alpha helix, isolated beta-bridge residue, strand, 3-10 helix, turn, bend)	6
AAPHY7 descriptors of a residue	7
BLOSUM62 descriptors of a residue	23
Phosphorylated (1 – yes, 0 – no, same for all nodes)	1
Mutated (1 – yes, 0 – no, same for all nodes)	1
Edge features	
Covalent bond (1 – yes, 0 – no)	1
Hydrophobic contact (1 – yes, 0 – no)	1
Hydrogen bond (from donor to acceptor; 1 – yes, 0 – no)	1
Hydrogen bond (from acceptor to donor; 1 – yes, 0 – no)	1
Salt bridge (from cation to anion; 1 – yes, 0 – no)	1
Salt bridge (from anion to cation; 1 – yes, 0 – no)	1
Pi-cation interaction (from aromatic ring to cation; 1 – yes, 0 – no)	1
Pi-cation interaction (from cation to aromatic ring; 1 – yes, 0 – no)	1
Parallel pi-stacking (1 – yes, 0 – no)	1
Perpendicular pi-stacking (1 – yes, 0 – no)	1

in a protein graph. It is essential in the case of the Davis dataset due to the presence of protein entries with the same sequence but annotated as phosphorylated/non-phosphorylated or wild-type and mutant entries with internal tandem duplication (ITD) mutation that is hard to translate into sequence unambiguously. Ignoring this data would cause the situation when proteins with identical graph representation have different binding affinities to the same ligand.

Generated protein graphs are available in the GitHub repository. The order of features in the graph is the same as in Table 4.

Model architecture

We used GNNs to extract features from the ligand and protein graphs followed by fully connected (FC) neural network layers. The general model architecture is represented in Fig. 4.

We tuned 3DProtDTA for the single GNN type or the combination of several GNN types that provides the best cross-validation results on the benchmark dataset. The following GNN types were considered: Graph Attention Network that fixes the static attention problem (GAT),[28] Graph Convolutional Network (GCN),[29] Graph Isomorphism Network (GIN),[30] Graph Isomorphism Network with incorporated edge features (GINE),[31] and GCN for learning molecular fingerprints (GMF).[32]

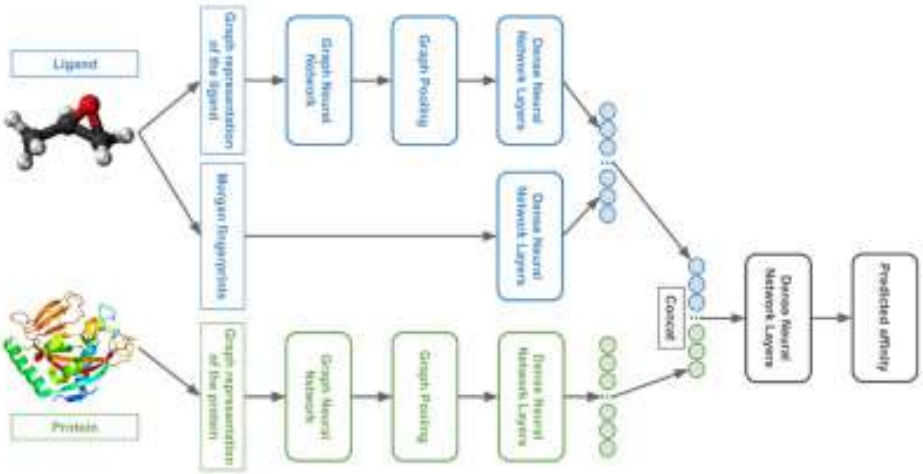


Fig. 4 The general architecture of the 3DProtDTA model.

Besides GNN types, the subjects to tuning included: the configuration of FC layers after 1D input data, GNN pooling output and final dense neural network; dropout rates; activation function for GNN layers; usage of batch normalisation; graph pooling type or combination of types.

The 3DProtDTA model was built with an open-source machine learning framework PyTorch[33] and the GNNs were implemented using PyTorch Geometric.[34]

Comparison with existing techniques

We compared the results of our approach to different classical machine learning-based and deep learning-based methods, which are considered to be state-of-art at the time of writing.

Similarity-based approaches KronRLS[7] and SimBoost[8] used a similarity matrix computed using Pubchem structure clustering server (Pubchem Sim, https://pubchem.ncbi.nlm.nih.gov) to represent ligands and the protein similarity matrix constructed with help of Smith–Waterman algorithm to represent targets.[35] KronRLS uses the regularised least-square model while SimBoost is the gradient boosting machine-based method.

The DeepDTA method[10] used 1D CNNs to process protein sequences and SMILES of the ligands. The GANsDTA[11] proposed a semi-supervised GANs-based method to predict binding affinity using target sequences and ligand SMILES. The same initial protein and ligand representations were used in the DeepCDA[9] method, where authors applied encoding by CNN and LSTM blocks. The GraphDTA[12] authors proposed GNNs to process ligand graphs, while proteins were still encoded by CNN applied to sequences.

Evaluation metrics

We selected 3 evaluation metrics used by most authors of the baseline approaches.

The mean squared error (MSE):

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - p_i)^2 \tag{2}$$

where n is the number of samples, y_i is the observed value, and p_i is the predicted value.

The concordance index (CI):[36]

$$\mathrm{CI} = \frac{1}{Z}\sum_{\delta_i > \delta_j} h(b_i - b_j) \tag{3}$$

where b_i is the prediction for the larger affinity δ_i, b_j is the prediction value for the smaller affinity δ_j, Z is a normalisation constant, $h(x)$ is the step function:

$$h(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0.5, & \text{if } x = 0 \\ 0, & \text{if } x < 0 \end{cases} \tag{4}$$

The r_m^2 index:[37]

$$r_m^2 = r^2 \times \left(1 - \sqrt{r^2 - r_0^2}\right) \tag{5}$$

where r^2 and r_0^2 are the squared correlation coefficients with and without intercept respectively.

Experimental setup

The experimental workflow can be split into the following steps: (1) tuning of the best model architecture to use in the benchmarking; (2) benchmarking itself (comparison with other approaches); (3) manual cross-validation experiments to assess the impact of particular choices in the input features model architecture; (4) assessment of the approach performance on cold (unseen by the model) protein target and generalisation over different kinase types.

To assess our approach properly, we used the same train-test and cross-validation data split as in the baseline approaches. Specifically, KIBA[13] and Davis[14] datasets were divided into six equal parts in which one part is selected as the independent test set. The remaining parts of the dataset were used to determine the hyper-parameters *via* 5-fold cross-validation. All 5-fold training sets were used for model training. Subsequently, 5 trained models were applied to predict test set affinity. Finally, an average for each metric was calculated and compared to the baseline approaches. Train and test folds of the datasets were obtained from DeepDTA[10] GitHub repository.

We tuned 3DProtDTA to choose the best GNN type or GNN types combination; the number of multi-head-attentions/size of output sample (number of output node features) in GNNs; usage of activation function after a GNN layer and type of the function (ReLU, Leaky ReLU, sigmoid); the configuration of FC layers; dropout rates; usage of batch normalisation; graph pooling type/types.

The tuning was performed with help of hyperparameter optimization software Optuna v. 3.0.3 (ref. 38) using the tree-

Table 5 The average MSE, CI, and r_m^2 scores of the test set trained on five different training sets for the Davis dataset

Approach	MSE	CI	r_m^2
KronRLS	0.379	0.871	0.407
SimBoost	0.282	0.872	0.644
DeepDTA	0.261	0.878	0.63
GANsDTA	0.276	0.881	0.653
DeepCDA	0.248	0.891	0.649
GraphDTA	0.229	0.893	0.63
3DProtDTA	**0.184**	**0.917**	**0.722**

Table 6 The average MSE, CI, and r_m^2 scores of the test set trained on five different training sets for the KIBA dataset

Approach	MSE	CI	r_m^2
KronRLS	0.411	0.782	0.342
SimBoost	0.222	0.836	0.629
DeepDTA	0.194	0.863	0.673
GANsDTA	0.224	0.866	0.675
DeepCDA	0.176	0.889	0.682
GraphDTA	0.139	0.891	0.673
3DProtDTA	**0.138**	**0.893**	**0.784**

structured Parzen estimator algorithm. We trained the model for 700 epochs and used a batch size of 32, the Adam optimiser with a learning rate of 0.0001, and the mean squared error loss function. The objective of the tuning was to minimise the MSE metric. The other metrics were used for testing the model performance but were not optimised during the training.

After the tuning by the tree-structured Parzen estimator algorithm, we ran a range of manual cross-validation experiments (after obtaining benchmarking results) keeping all the components in the best tuned architecture fixed except:

- The type of GNN architecture for a protein;
- The type of GNN architecture for a ligand;
- The type of graph pooling for both protein and ligand;
- Usage of ligand graph or Morgan fingerprint as the only ligand features.

These experiments were performed in order to assess in depth the impact of GNN architecture, graph pooling and ligand features. While changing the type of GNN model, we kept the size of output (or the number of attention heads in the case of GAT) equal or as close as possible to the tuned parameters.

The last set of experiments was conducted to assess how the 3DProtDTA performs on cold protein targets and, at the same time, to generalise over different kinase types. For that purpose, we annotated the proteins in both datasets using InterPro database entries.[69,69] The number of proteins annotated by the top 20 entries can be found in Tables S6 and S7.† As expected, two major kinase groups in both datasets were tyrosine-protein kinases (InterPro entry IPR008266) and serine/threonine-protein kinases (InterPro entry IPR008271). Subsequently, we randomly selected 10 tyrosine kinases and 10 serine/threonine kinases from each benchmarking dataset and separated all the samples containing those proteins into cold target test datasets. The rest of the samples were filtered to create 3 training datasets (separately for Davis and KIBA benchmarks): (1) all the remaining samples without filtering; (2) the samples with all tyrosine kinases excluded; (3) the samples with all serine/threonine kinases excluded. Consequently, we obtained 3 models per benchmarking dataset: trained on all data, trained on the data without tyrosine kinases, and trained on the data without serine/threonine kinases. We collected the metrics from all cold target test splits.

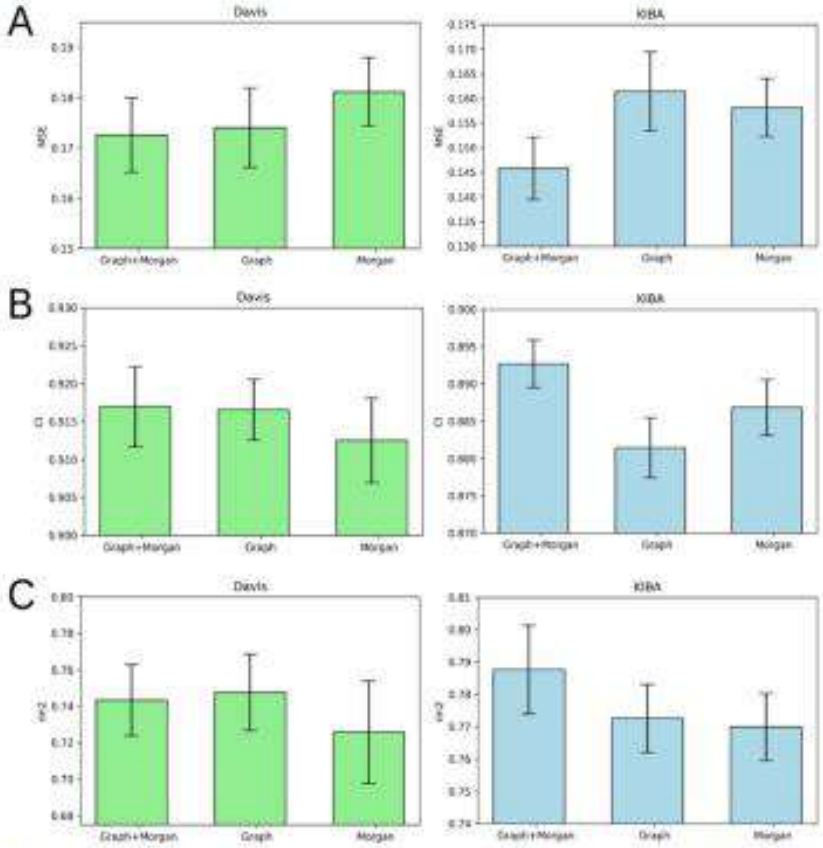


Fig. 5 Average MSE (A), CI (B), and r_m^2 index (C) after 5-fold cross-validation for Davis (left) and KIBA (right) datasets. Models that use molecular graph and Morgan fingerprint (graph + Morgan), only molecular graph (graph) or only Morgan fingerprint (Morgan) as ligand representation are compared. Error bars represent standard deviation.

Results and discussion

Benchmarking

The best model architectures and tuned hyperparameters for the benchmark are available in ESI (Fig. S1†). All the data for model training is freely available in the GitHub repository (https://github.com/vtarasw/3d-prot-dta.git).

Tables 5 and 6 compare obtained model performance metrics to the baseline approaches for Davis and KIBA datasets respectively. The best scores are shown in bold.

According to obtained results, 3DProtDTA considerably outperforms other approaches in terms of all 3 metrics on the Davis dataset. In the case of the KIBA dataset, the only metric that demonstrated significant improvement over competitors was r_m^2, while the two other metrics were comparable (but still superior) to the rivals.

Performance of ligand feature types

Atom-level molecular graphs and Morgan fingerprints are widely used types of features in the development of predictive models that take small molecules as input. The result of model training on each type separately or both types together is provided in Fig. 5.

The results demonstrate the nearly equal performance of molecular graph only and combined representation of ligands for the Davis dataset. Nevertheless, the combined representation is clearly superior to others in terms of the KIBA dataset.

We identified the molecular diversity in both datasets defined as 1 − Tanimoto similarity scores based on Morgan fingerprints with radius 3 and averaged for each pair of ligands in the dataset. They are equal to 0.882 and 0.892 for the Davis and KIBA datasets respectively. Despite comparable molecular diversity, the model performance on Davis and KIBA datasets is different when comparing graph only and combined representation. We attribute this to two factors: a different number of ligands (68 in Davis and 2111 in KIBA, Table 1) and a much wider distribution of the number of atoms in the molecules from the KIBA dataset (Fig. 2). The graph only representation is most likely insufficient to generalise over all the molecules in the KIBA dataset, especially, for the small cohort of ligands with more than 50 heavy atoms.

The model trained on the molecular graph and Morgan fingerprint together provided the best average MSE, CI, and r_m^2

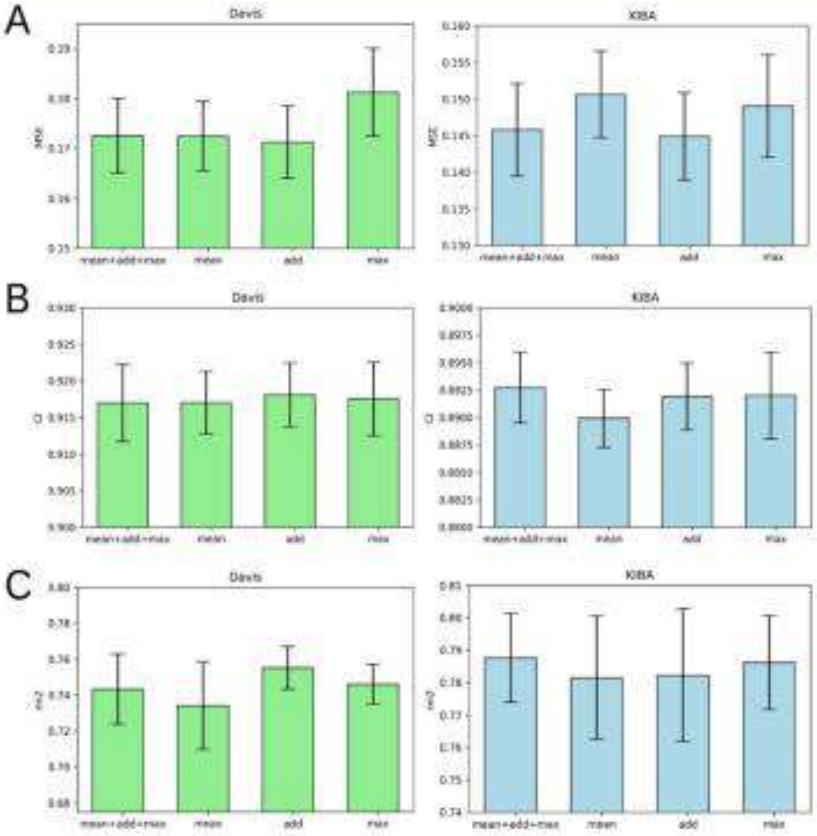


Fig. 6 Average MSE (A), CI (B), and r_m^2 index (C) after 5-fold cross-validation for Davis (left) and KIBA (right) datasets. Models that use mean graph pooling (mean), add graph pooling (add), max graph pooling (max) or all three poolings concatenated (mean + add + max) are compared. Error bars represent standard deviation.

index of both benchmark datasets (Tables S1–S5†). Therefore, we can suggest that these two ligand feature types contain some non-overlapping information useful for DTA prediction.

Performance of graph pooling methods

Graph pooling is a crucial step used to generate the same length 1D latent representation of data processed by GNN for subsequent processing by FC layers. There are three common types of pooling methods including mean pooling, max pooling and add (sum) pooling. Fig. 6 provides a comparison of these pooling approaches.

Fig. 6 demonstrates that add pooling is the best choice for the Davis dataset, while add pooling is comparable to the combined approach (mean + add + max poolings concatenated) in the KIBA dataset cross-validation results. The add pooling provides superior average MSE, CI, and r_m^2 index of both benchmark datasets (Tables S1–S5†).

Performance of ligand GNN types

Fig. 7 illustrates that there is no obvious leader or outsider as the GNN for ligand graph processing. The average of the two benchmark datasets is very close for all the GNN types considering any of the three evaluation metrics (Tables S1–S5†). It is important to highlight that the only GNN model that used edge features during the training was GINE. Thus, the specific characteristic of the edge (like covalent bond type) doesn't play a crucial role in affinity prediction.

Performance of protein GNN types

According to Fig. 8, it is hard to extract a particular GNN model that works the best as a protein graph encoder. The GAT, GCN, GIN and GINE provide very similar average results. The GMF model, however, is noticeably worse (Tables S1–S5†). Analogously to the performance of different GNNs on ligand graphs, the only GNN considering edge features of the protein graph GINE didn't show noticeable metrics improvement relative to other GNN types. This suggests that covalent and non-covalent interactions of amino acid residues alone provide enough information for DTA prediction.

Performance on cold protein target and different kinase types

The performance of our model trained on either all kinases, data without tyrosine kinases, or data without serine/

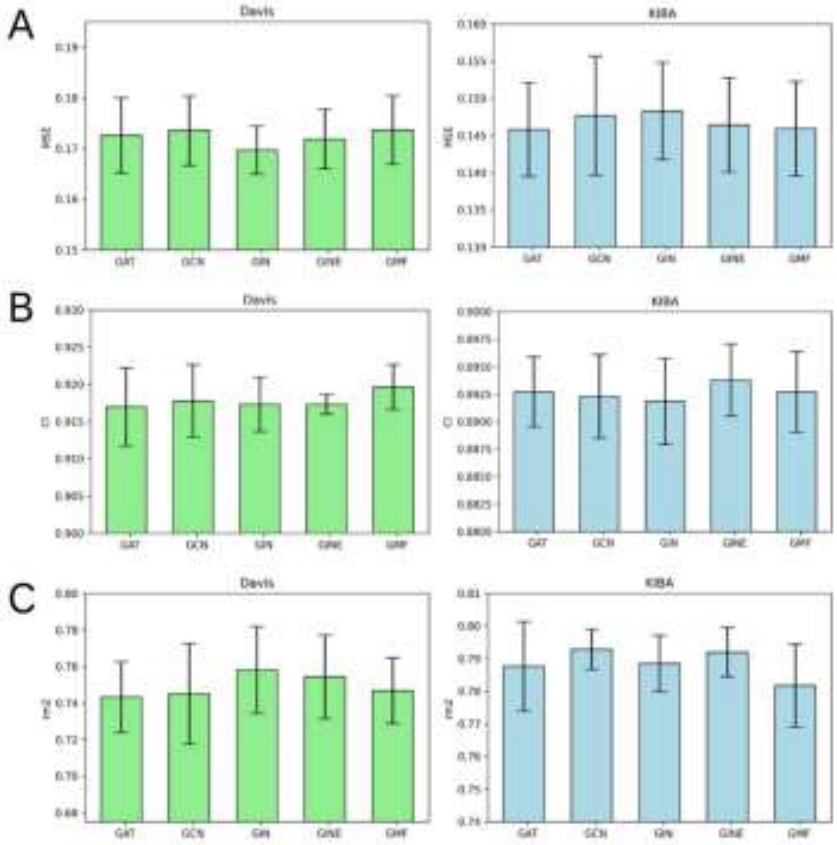


Fig. 7 Average MSE (A), CI (B), and r_m^2 index (C) after 5-fold cross-validation for Davis (left) and KIBA (right) datasets. Models that use GAT, GCN, GIN, GINE or GMF to process ligand atom-level graphs are compared. Error bars represent standard deviation.

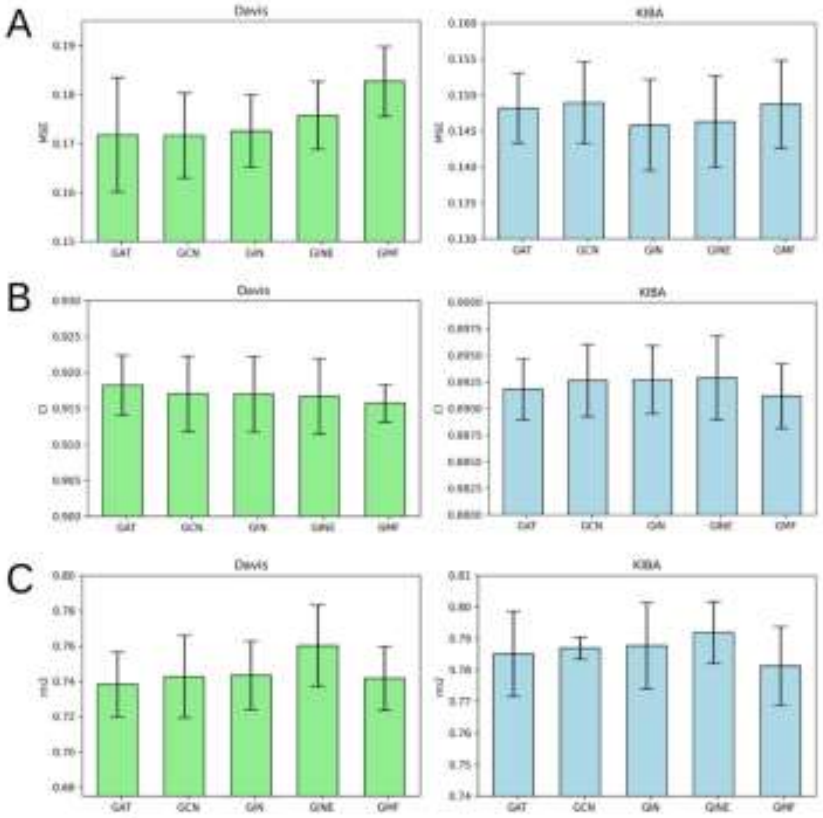


Fig. 8 Average MSE (A), CI (B), and r_m^2 index (C) after 5-fold cross-validation for Davis (left) and KIBA (right) datasets. Models that use GAT, GCN, GIN, GINE or GMF to process protein residue-level graphs are compared. Error bars represent standard deviation.

Table 7 The MSE, CI, and r_m^2 scores of the test sets composed of all samples containing one of 10 random tyrosine or serine/threonine kinases for the Davis dataset

	Test, cold tyrosine kinases			Test, cold serine/threonine kinases		
Train dataset	MSE	CI	r_m^2	MSE	CI	r_m^2
All kinases	0.305	0.880	0.536	0.311	0.889	0.509
No tyrosine kinases	0.719	0.746	0.254	**0.284**	**0.909**	**0.572**
No serine/threonine kinases	**0.239**	**0.909**	**0.629**	0.683	0.682	0.136

Table 8 The MSE, CI, and r_m^2 scores of the test sets composed of all the samples containing one of 10 random tyrosine or serine/threonine kinases for the KIBA dataset

	Test, cold tyrosine kinases			Test, cold serine/threonine kinases		
Train dataset	MSE	CI	r_m^2	MSE	CI	r_m^2
All kinases	0.383	0.668	0.599	**0.236**	0.857	**0.700**
No tyrosine kinases	0.904	0.634	0.155	0.238	**0.858**	0.693
No serine/threonine kinases	**0.359**	**0.729**	**0.625**	0.589	0.713	0.277

threonine kinases was evaluated on a cold protein, as well as on a cold kinase type (Tables 7 and 8). The best scores are shown in bold.

The results show that the model trained on all kinase types performs quite similarly for both tyrosine and serine/threonine kinases in the Davis dataset. In contrast, the model trained on all kinase types of the KIBA dataset demonstrates considerably better performance for serine/threonine kinases. We expected the opposite result because the rate of serine/threonine kinases to tyrosine kinases is roughly 2 : 1 in the KIBA dataset and 3 : 1 in the Davis dataset (Tables S6 and S7†).

The models trained without tyrosine kinases demonstrate the best or nearly the best performance on the cold serine/threonine kinases test and *vice versa*. Their performance is better than for the models trained on all kinase types. This means that adding new kinase types to the dataset may have a negative impact on the model inference for other kinase types. A potential solution to this issue is training a multi-task model, which will contain common layers for all kinase types in the beginning and separate layers for each kinase type at the end. Development and testing of such a model are out of the scope of this work, however.

The models trained without a particular type of kinases perform significantly worse on the test with omitted types of kinases. This is expectable behaviour, which emphasises that the models perform the best for protein types, which were present in the training set.

Limitations and perspectives

Although the usage of protein structures from the AlphaFold database provides data uniformity, it has some drawbacks. Particularly, this approach introduces uncertainty if multiple experimental structures are resolved and used as templates. In such cases, the AlphaFold could potentially produce a blend between several alternative protein conformations, which is not functionally relevant. A potential improvement would be the usage of all available experimentally determined protein structures along with the AlphaFold predictions. However, we observed that the residue-level graphs of several randomly picked experimental structures of kinases are nearly identical to ones generated from the AlphaFold predicted structures (Table S8†). The probable reasons for this are: (1) the abundance of various experimentally determined kinase domain structures available for AlphaFold that positively impacts the quality of predictions (Fig. 3B) and (2) the coarse-grained resolution on the residue-level graph representation used in this work (which would not be the case for atom-level graph representations, though). The usage of the AlphaFold structures provides additional benefits, such as covering proteins without known experimental structures and avoiding problems with incomplete structures and missing or ambiguous atoms.

Another straightforward improvement is the usage of atomic-level protein graphs instead of residue-level ones, and the usage of more comprehensive node and edge features. Particularly, the B-factors and other measures of protein flexibility, such as cross-correlation matrices of motions, could be used.

It is necessary to note that there are other ML-based approaches to predict drug-target affinity, which were not included in this study, such as CSatDTA,[41] FusionDTA,[42] NerLTR-DTA,[43] Masashi's method,[44] WGNN-DTA,[45] *etc.* In these techniques, the experimental setups and data split into training and testing datasets are either different from what we use in the current work or not specified in enough detail. Some of them also utilise specific evaluation metrics, which prevent their direct comparison with other methods.

Our attempts to re-train some of these models using our data split had failed. For example, there is no source code available for NerLTR-DTA, while provided binary crashes. Since the bug reports on GitHub have not been answered for several years, we decided that further attempts to use this technique are futile. The authors of the FusionDTA do not report the amount of computational resources required for the model training. According to our internal estimates, this model is much heavier than 3DProtDTA and its proper re-training is prohibitively long and expensive on our computational facilities. In addition, FusionDTA utilises different hyperparameters for each dataset, which diverges from our concept of the universal model architecture and hyperparameters for all datasets.

Therefore, we decided to limit the scope of our work to the methods, which possess identical setups, datasets and performance metrics and thus allow the apple-to-apple comparison without re-training the third-party ML models.

Conclusions

In this work we developed a new deep learning model for assessing drug–protein affinities called 3DProtDTA. The distinctive feature of this model is graph-based representation of both protein and the ligands, which retain a significant amount of information about their connectivity and spatial arrangement without introducing excessive computational burden. The features for model training were created using the AlphaFold database of predicted protein structures which allows for covering all proteins in two common benchmark datasets. We tuned a wide range of GNN-based model architectures and their combinations to achieve the best model performance. The 3DProtDTA outperforms its competitors on common benchmarking datasets and has a potential for further improvement.

Data availability

https://github.com/vtarasv/3d-prot-dta.git.

Author contributions

SS and AN designed the study and supervised model development and testing. SY coordinated the work, provided scripts for Pteros molecular modelling library and participated in results interpretation. TV developed the modules for features generation, model tuning and testing. TV and RS researched existing methods for drug-target affinity prediction. DN and IK participated in protein preprocessing before feature generation. LP, ZO, and RZ participated in consultations about the best practices and strategies for model tuning and efficient training. PH, IK, and VV performed and supervised the tuning and testing process. The manuscript was written by TV and SY.

Conflicts of interest

All authors are employees of Receptor.AI INC. SS, AN and SY have shares in Receptor.AI INC.

Notes and references

1 A. D. Roses, *Nat. Rev. Drug Discovery*, 2008, **7**, 807–817.
2 G. A. Van Norman, *JACC: Basic Transl. Sci.*, 2019, **4**, 428–437.
3 J. Arrowsmith, *Nat. Rev. Drug Discovery*, 2011, **10**, 328–329.
4 M. Thafar, A. B. Raies, S. Albaradei, M. Essack and V. B. Bajic, *Front. Chem.*, 2019, **7**, 782.
5 L. Pinzi and G. Rastelli, *Int. J. Mol. Sci.*, 2019, **20**, 4331.
6 J. Li, A. Fu and L. Zhang, *Interdiscip. Sci.: Comput. Life Sci.*, 2019, **11**, 320–328.
7 T. Pahikkala, A. Airola, S. Pietila, S. Shakyawar, A. Szwajda, J. Tang and T. Aittokallio, *Briefings Bioinf.*, 2015, **16**, 325–337.

8 T. He, M. Heidemeyer, F. Ban, A. Cherkasov and M. Ester, *J. Cheminf.*, 2017, **9**, 24.
9 K. Abbasi, P. Razzaghi, A. Poso, M. Amanlou, J. B. Ghasemi and A. Masoudi-Nejad, *Bioinformatics*, 2020, **36**, 4633–4642.
10 H. Öztürk, A. Özgür and E. Ozkirimli, *Bioinformatics*, 2018, **34**, i821–i829.
11 L. Zhao, J. Wang, L. Pang, Y. Liu and J. Zhang, *Front. Genet.*, 2020, **10**, 1243.
12 T. Nguyen, H. Le, T. P. Quinn, T. Nguyen, T. D. Le and S. Venkatesh, *Bioinformatics*, 2021, **37**, 1140–1147.
13 J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli and D. Hassabis, *Nature*, 2021, **596**, 583–589.
14 M. L. Davis, J. P. Hunt, S. Herrgard, P. Ciceri, L. M. Wodicka, G. Pallares, M. Hocker, D. K. Treiber and P. P. Zarrinkar, *Nat. Biotechnol.*, 2011, **29**, 1046–1051.
15 J. Tang, A. Szwajda, S. Shakyawar, T. Xu, P. Hintsanen, K. Wennerberg and T. Aittokallio, *J. Chem. Inf. Model.*, 2014, **54**, 735–743.
16 The UniProt Consortium, *Nucleic Acids Res.*, 2019, **47**, D506–D515.
17 K. Liu, X. Sun, L. Jia, J. Ma, H. Xing, J. Wu, H. Gao, Y. Sun, F. Boulnois and J. Fan, *Int. J. Mol. Sci.*, 2019, **20**, 3389.
18 D. Rogers and M. Hahn, *J. Chem. Inf. Model.*, 2010, **50**, 742–754.
19 A. Gobbi and D. Poppinger, *Biotechnol. Bioeng.*, 1998, **61**, 47–54.
20 M. Mirdita, K. Schütze, Y. Moriwaki, L. Heo, S. Ovchinnikov and M. Steinegger, *Nat. Methods*, 2022, **19**, 679–682.
21 P. J. A. Cock, T. Antao, J. T. Chang, B. A. Chapman, C. J. Cox, A. Dalke, I. Friedberg, T. Hamelryck, F. Kauff, B. Wilczynski and M. J. L. de Hoon, *Bioinformatics*, 2009, **25**, 1422–1423.
22 S. O. Yesylevskyy, *J. Comput. Chem.*, 2015, **36**, 1480–1488.
23 S. O. Yesylevskyy, *J. Comput. Chem.*, 2012, **33**, 1632–1636.
24 M. Wójcikowski, P. Zielenkiewicz and P. Siedlecki, *J. Cheminf.*, 2015, **7**, 26.
25 J. Meiler, A. Zeidler, F. Schmäschke and M. Müller, *J. Mol. Model.*, 2001, **7**, 360–369.
26 D. W. Mount, *Cold Spring Harbor Protocols*, 2008, **2008**, pdb.top39.
27 J. Chen, S. Zheng, H. Zhao and Y. Yang, *J. Cheminf.*, 2021, **13**, 7.
28 S. Brody, U. Alon and E. Yahav, *arXiv*, 2021, preprint, arXiv:2105.14491, DOI: 10.48550/ARXIV.2105.14491.
29 T. N. Kipf and M. Welling, *arXiv*, 2016, preprint, arXiv:1609.02907, DOI: 10.48550/ARXIV.1609.02907.
30 K. Xu, W. Hu, J. Leskovec and S. Jegelka, *arXiv*, 2018, preprint, arXiv:1810.00826, DOI: 10.48550/ARXIV.1810.00826.
31 W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande and J. Leskovec, *arXiv*, 2019, preprint, arXiv:1905.12265, DOI: 10.48550/ARXIV.1905.12265.
32 D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik and R. P. Adams, *arXiv*, 2015, preprint, arXiv:1509.09292, DOI: 10.48550/ARXIV.1509.09292.
33 A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai and S. Chintala, *arXiv*, 2019, preprint, arXiv:1912.01703, DOI: 10.48550/ARXIV.1912.01703.
34 M. Fey and J. E. Lenssen, *arXiv*, 2019, preprint, arXiv:1903.02428, DOI: 10.48550/ARXIV.1903.02428.
35 T. F. Smith and M. S. Waterman, *J. Mol. Biol.*, 1981, **147**, 195–197.
36 M. Gönen and G. Heller, *Biometrika*, 2005, **92**, 965–970.
37 K. Roy, P. Chakraborty, I. Mitra, P. K. Ojha, S. Kar and R. N. Das, *J. Comput. Chem.*, 2013, **34**, 1071–1082.
38 T. Akiba, S. Sano, T. Yanase, T. Ohta and M. Koyama, *arXiv*, 2019, preprint, arXiv:1907.10902, DOI: 10.48550/ARXIV.1907.10902.
39 M. Blum, H.-Y. Chang, S. Chuguransky, T. Grego, S. Kandasaamy, A. Mitchell, G. Nuka, T. Paysan-Lafosse, M. Qureshi, S. Raj, L. Richardson, G. A. Salazar, L. Williams, P. Bork, A. Bridge, J. Gough, D. H. Haft, I. Letunic, A. Marchler-Bauer, H. Mi, D. A. Natale, M. Necci, C. A. Orengo, A. P. Pandurangan, C. Rivoire, C. J. A. Sigrist, I. Sillitoe, N. Thanki, P. D. Thomas, S. C. E. Tosatto, C. H. Wu, A. Bateman and R. D. Finn, *Nucleic Acids Res.*, 2021, **49**, D344–D354.
40 T. Paysan-Lafosse, M. Blum, S. Chuguransky, T. Grego, B. L. Pinto, G. A. Salazar, M. L. Bileschi, P. Bork, A. Bridge, L. Colwell, J. Gough, D. H. Haft, I. Letunić, A. Marchler-Bauer, H. Mi, D. A. Natale, C. A. Orengo, A. P. Pandurangan, C. Rivoire, C. J. A. Sigrist, I. Sillitoe, N. Thanki, P. D. Thomas, S. C. E. Tosatto, C. H. Wu and A. Bateman, *Nucleic Acids Res.*, 2023, **51**, D418–D427.
41 A. Ghimire, H. Tayara, Z. Xuan and K. T. Chong, *Int. J. Mol. Sci.*, 2022, **23**, 8453.
42 W. Yuan, G. Chen and C. Y.-C. Chen, *Briefings Bioinf.*, 2022, **23**, bbab506.
43 X. Ru, X. Ye, T. Sakurai and Q. Zou, *Bioinformatics*, 2022, **38**, 1964–1971.
44 M. Tsubaki, K. Tomii and J. Sese, *Bioinformatics*, 2019, **35**, 309–318.
45 M. Jiang, S. Wang, S. Zhang, W. Zhou, Y. Zhang and Z. Li, *BMC Genomics*, 2022, **23**, 449.

ДОДАТОК В
(обов'язковий)

ПРЕЗЕНТАЦІЯ ДО ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Кваліфікаційна робота магістра

Універсальна система автоматизованого тренування та моніторингу моделей машинного навчання для SAAS платформи in silico розробки ліків. On-premise деплоймент на інфраструктурі фармакологічних компаній за допомогою оркестратора Airflow

Спеціальність: 123 – комп'ютерна інженерія

Магістр: Возняк Володимир Зіновійович

Науковий керівник: д.ф., ст.викл.каф. КІІС Павлова О.О.

RECEPTOR.AI

2

Мета, предмет та об'єкт дослідження

Мета: створення автоматизованого пайплайну тренування моделей машинного навчання на біологічних даних, який володіє властивостями модульності, детального налаштування конфігурації, паралелізації, швидкого масштабування та деплойменту у сценарії On-Premise.

Предмет дослідження: предметом дослідження є автоматизовані пайплайни машинного навчання для біологічних задач з розробки нових ліків для On-Premise інфраструктур.

Об'єкт дослідження: автоматизовані пайплайни машинного навчання.

RECEPTOR.AI

3

Задачі дослідження

1. Аналіз існуючих технологій оркестрації контейнерів та автоматизації процесів машинного навчання.
2. Розробка масштабованого та гнучкого пайплайну для тренування і оцінки моделей машинного навчання, який може бути інтегрований у SAAS платформу
3. Порівняння провайдерів хмарних послуг для визначення оптимального середовища для розгортання пайплайнів.

4

Наукова новизна

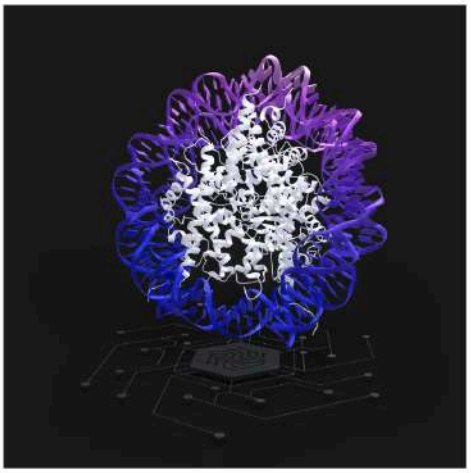
1. Розроблено On-Premise версію автоматизованої системи для тренування та моніторингу моделей машинного навчання на біологічних даних.
2. Розроблену автоматизовану систему інтегровано в SAAS платформу для in silico розробки ліків.
3. За допомогою створеної системи натреновано моделі для передбачення ADME-Tox параметрів молекул та інтегрувано їх у SAAS платформу.

Практична цінність

Практична значимість отриманих результатів полягає у можливості автоматичного тренування моделей машинного навчання на молекулярних даних і подальшого їх використання у SAAS платформі для розробки ліків.

Вступ

Сучасний світ науки та технологій переживає безпрецедентний розвиток інновацій, що прискорюються завдяки стрімкому прогресу в області інформаційних технологій. Особливе місце в цій революції займає машинне навчання, яке здатне трансформувати різні галузі людської діяльності, пропонуючи нові підходи до аналізу даних, прийняття рішень та автоматизації процесів.



Розділ 1. Огляд існуючих рішень

Pharm-AutoML

Переваги:

- автоматизація попередньої обробки даних та налаштування моделей;
- аналіз та інтерпретація результатів.

Недоліки:

- обмеження для мультикласифікації.

Chemistry42

Переваги:

- генерація нових молекулярних структур;
- швидкість роботи.

Недоліки:

- обмежене використання (лише для DDR1 та CDK20).

RECEPTOR.AI

7

Розділ 1. Огляд існуючих рішень

Pharm-AutoML

Переваги:

- автоматизація попередньої обробки даних та налаштування моделей;
- аналіз та інтерпретація результатів.

Недоліки:

- обмеження для мультикласифікації.

Chemistry42

Переваги:

- генерація нових молекулярних структур;
- швидкість роботи.

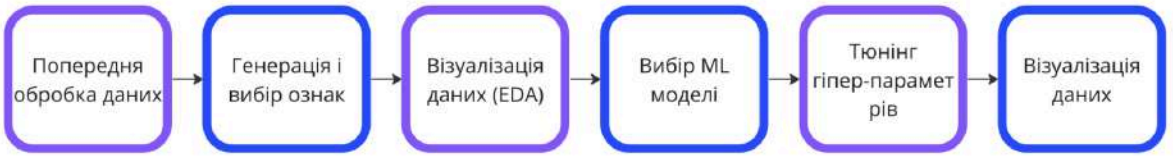
Недоліки:

- обмежене використання (лише для DDR1 та CDK20).

RECEPTOR.AI

8

Розділ 1. Основні складові пайплайну автоматичного тренування моделей машинного навчання



RECEPTOR.AI

9

Розділ 2. Математичне формулювання алгоритмів машинного навчання

$$\hat{y} = f(x; \theta),$$

де $\hat{y}$ – прогнозоване вихідне значення;

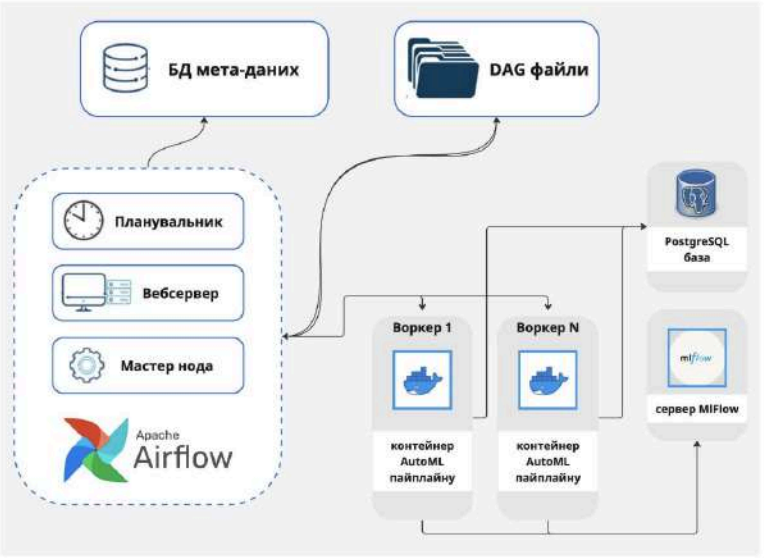
x – вектор вхідних атрибутів;

θ – параметри моделі (наприклад, ваги у нейронній мережі);

RECEPTOR.AI

10

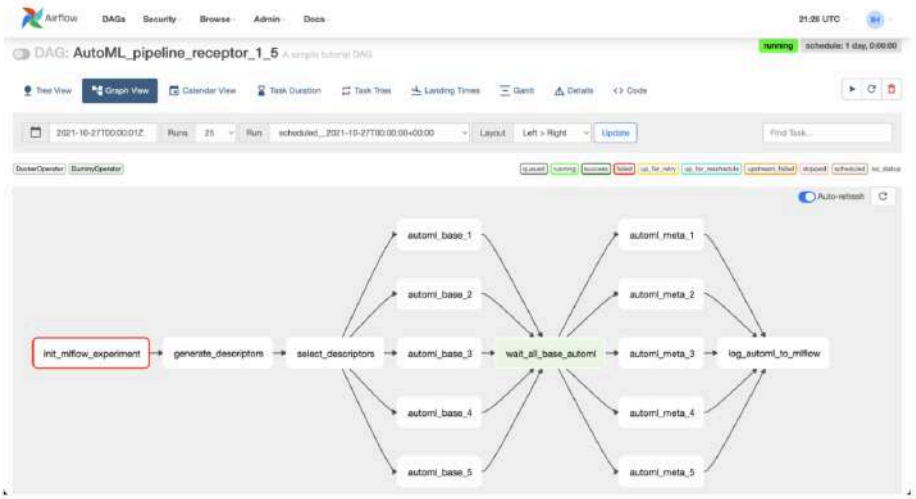
Розділ 3. Архітектура Airflow



RECEPTOR.AI

11

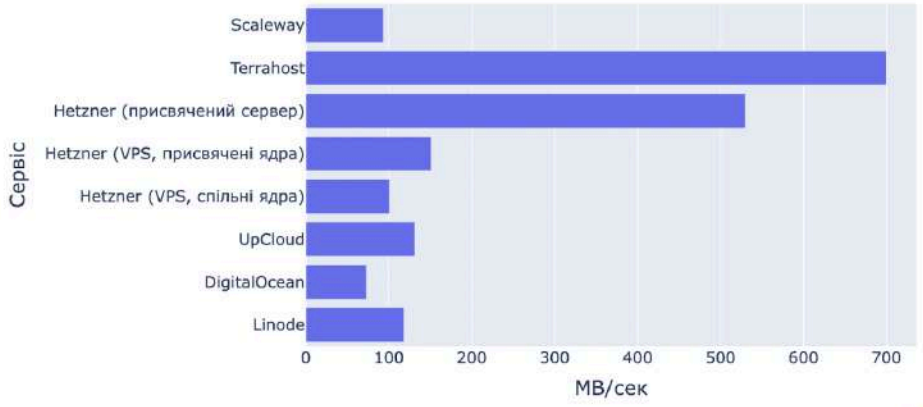
Розділ 3. Компоненти AutoML пайплайну



RECEPTOR.AI

12

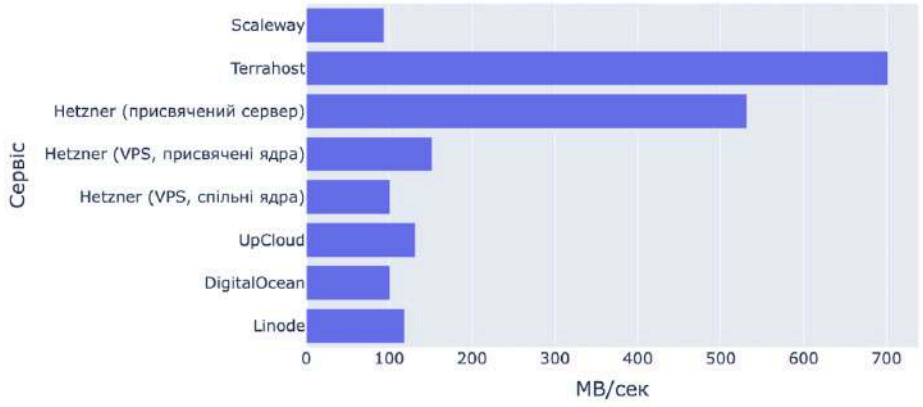
Розділ 3. FIO диск розміром блоку 4K – пропускна здатність на зчитування Мб/с



RECEPTOR.AI

13

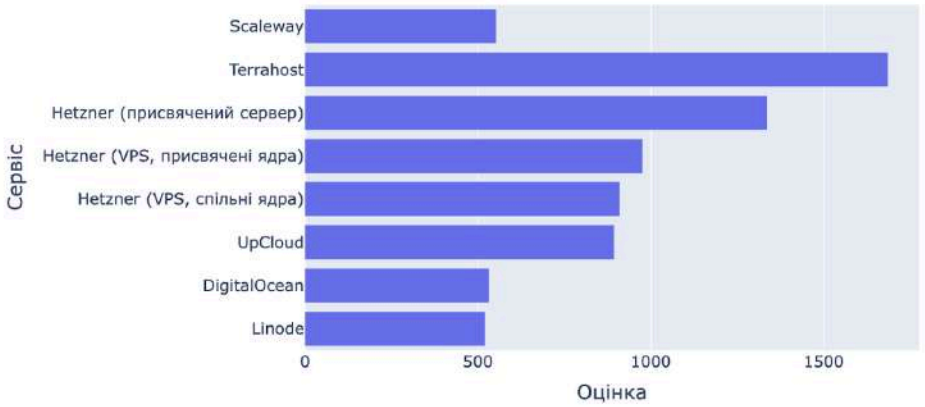
Розділ 3. FIO диск розміром блоку 4K – пропускна здатність на запис Мб/с



RECEPTOR.AI

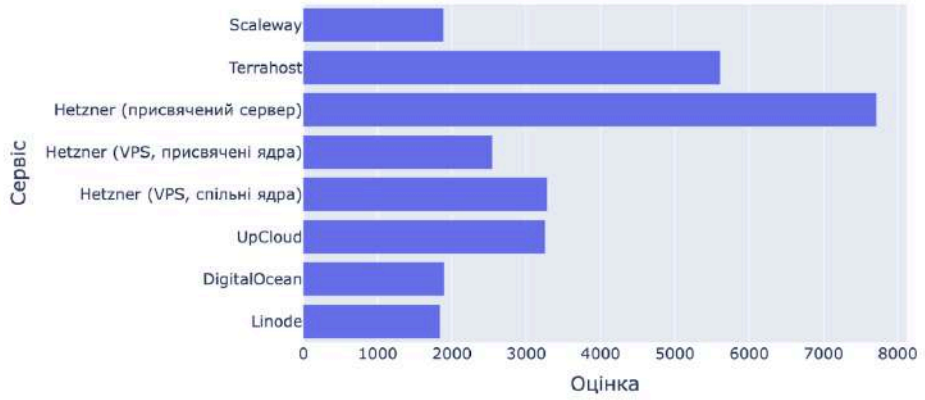
14

Розділ 3. Одноядрова оцінка Geekbench 5



15

Розділ 3. Багатоядрова оцінка Geekbench 5



16

Розділ 4. Користувальницький інтерфейс MLFlow

RECEPTOR.AI

17

Розділ 4. Відображення параметрів експеримента у MLFlow

Name	Value
base_CatBoost_auto_class_weights	None
base_CatBoost_l2_leaf_reg	0.5198349069607762
base_CatBoost_learning_rate	0.09268596443136451
base_CatBoost_max_depth	8
base_ExtraTrees_ccp_alpha	0.00019242246203737823
base_ExtraTrees_class_weight	None
base_ExtraTrees_max_depth	30
base_ExtraTrees_max_features	None
base_ExtraTrees_min_samples_leaf	9
base_ExtraTrees_min_samples_split	25
base_ExtraTrees_n_estimators	693

RECEPTOR.AI

18

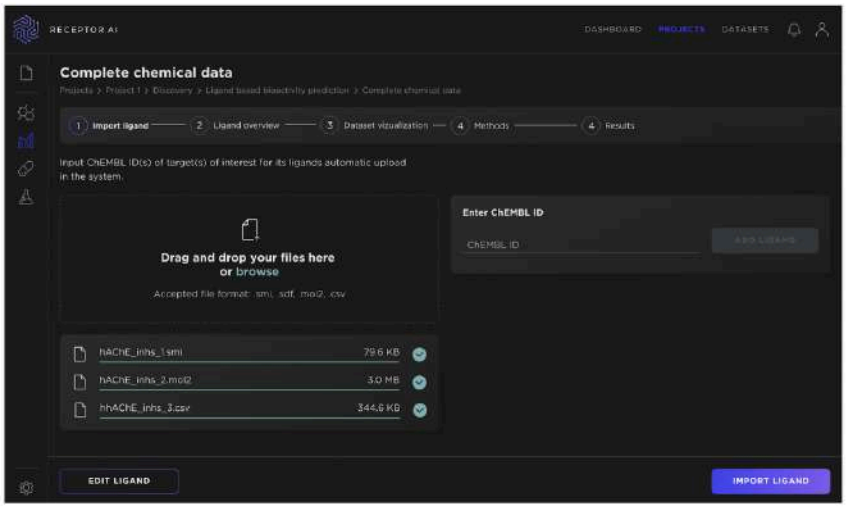
Розділ 4. Відображення додаткової інформації експеримента у MLFlow

Name	Value	Actions
BEST_BASE_MODEL_FAMILY	SVMLinearOvO	
BEST_META_MODEL_FAMILY	LinearOvO	
BEST_MODEL_SCORER	auto	
CV_SPLITTER_INSTANCE	{'method': 'StratifiedKFold', 'kwargs': {'n_splits': 5, 'shuffle': True, 'random_state': 47}}	

RECEPTOR.AI

19

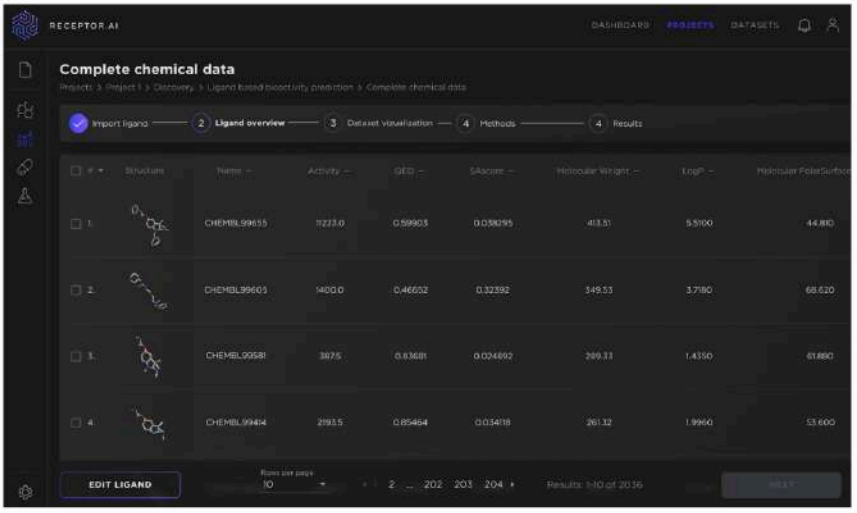
Розділ 4. Сторінка завантаження користувачем власного тренувального датасету



RECEPTOR.AI

20

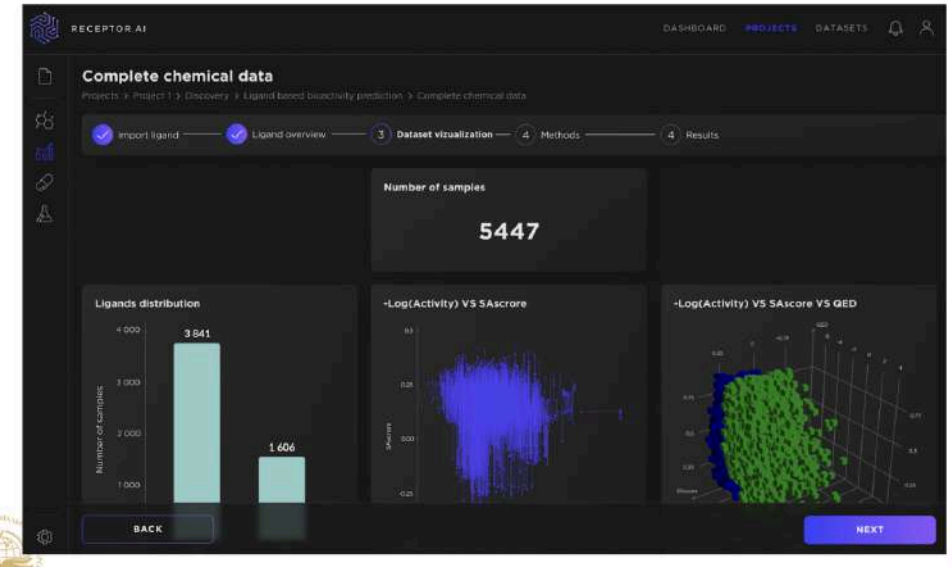
Розділ 4. Сторінка візуалізації завантаженого користувачем тренувального датасету



RECEPTOR.AI

21

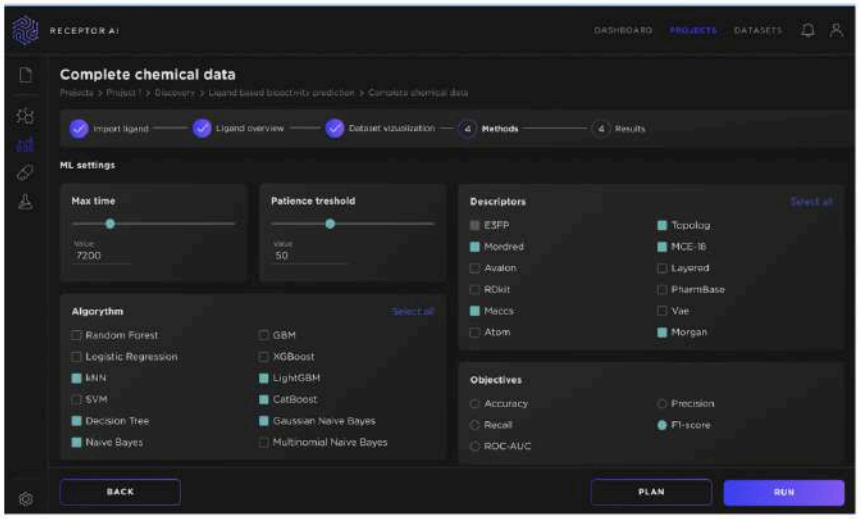
Розділ 4. Сторінка дослідницького аналізу даних



RECEPTOR.AI

22

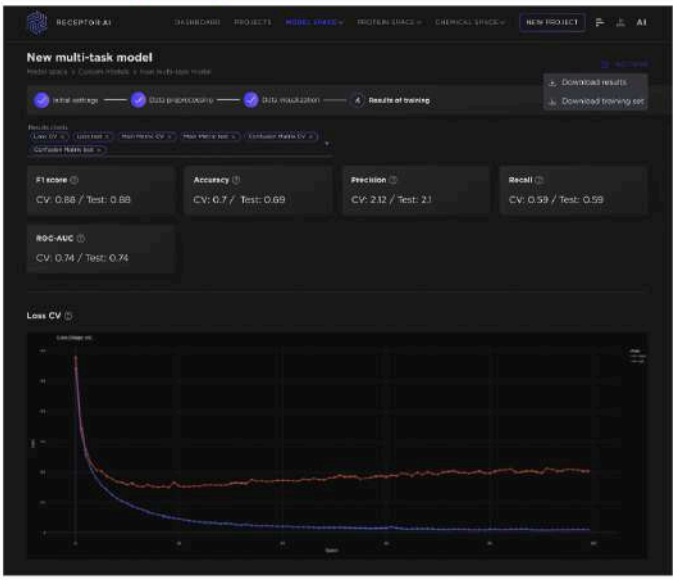
Розділ 4. Сторінка налаштування тренувального етапу пайплайну



RECEPTOR.AI

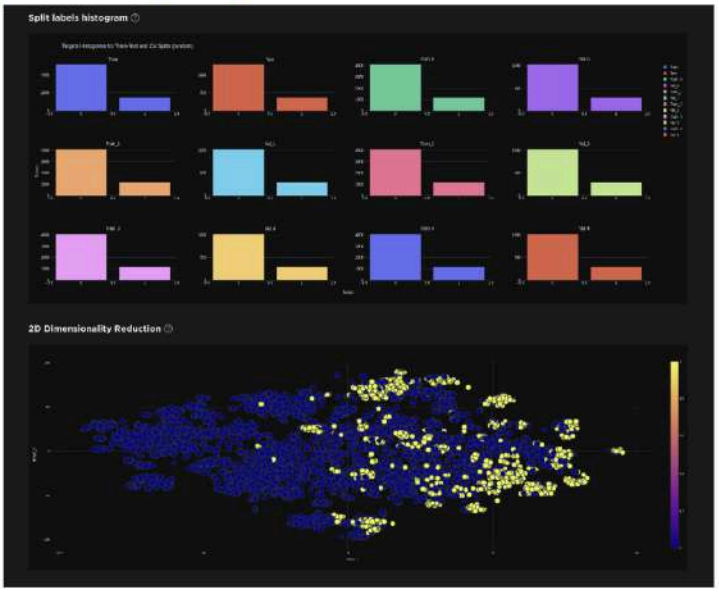
23

Розділ 4. Приклад результатів



RECEPTOR.AI

Розділ 4. Приклад результатів



RECEPTOR.AI

Наукова публікація

2023 | ***cited in Scopus*** | ***Q1 journal***

3DProtDTA: The Deep Learning Model For Drug-Target Affinity Prediction Based On The Residue-Level Protein Graphs

Article | *RSC Advances (Q1) 13 (15), 10261-10272*

T. Voitsitskyi, R. Stratiichuk, Z. Ostrovsky, V. Vozniak et al.


ROYAL SOCIETY OF CHEMISTRY

RSC Advances

PAPER

3DProtDTA: a deep learning model for drug-target affinity prediction based on residue-level protein graphs†

Introduction

RECEPTOR.AI

26

Висновки

- На основі проведених досліджень розроблено архітектуру і компоненти програмного забезпечення, яке інтегровано в існуючу SAAS платформу з розробки ліків.
- У першому розділі було проведено детальний огляд ключових складових, необхідних для AutoML пайплайну. Проаналізовано та порівняно різноманітні технології, що можуть бути використані для реалізації кожного елемента пайплайну.
- У другому розділі було проведено детальний огляд математичної моделі алгоритму вибору найкращих ознак, серед яких можна виділити фільтраційні, обгорткові та вбудовані методи. Також було наведено формулювання основних алгоритмів машинного навчання, а також було розглянуто математичний опис процесу баєсівської оптимізації для тюнінгу гіперпараметрів моделі.
- У третьому розділі було досліджено вибір технологій для реалізації компонентів системи автоматизованого тренування та моніторингу моделей машинного навчання.
- У четвертому розділі було описано SAAS платформу для розробки ліків.

RECEPTOR.AI

27

Дякую за увагу!

RECEPTOR.AI

UNICHECK
by Turnitin

Ім'я користувача:
Кафедра КІ

ID перевірки:
1016252702

Дата перевірки:
15.05.2024 07:09:10 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
15.05.2024 07:21:37 EEST

ID користувача:
100005591

Назва документа: Возняк_Універсальна система автоматизованого тренування та моніторингу моделей маши...

Кількість сторінок: 107 **Кількість слів:** 14783 **Кількість символів:** 121058 **Розмір файлу:** 11.77 MB **ID файлу:** 1016039020

1.53%
Схожість

Найбільша схожість: 0.54% з джерелом з Бібліотеки (ID файлу: 1010865386)

0.99% Джерела з Інтернету	81	Сторінка 109
0.68% Джерела з Бібліотеки	63	Сторінка 109

0.11% Цитат

Цитати	1	Сторінка 110
Посилання	1	Сторінка 110

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	19

Wed May 15 06:30:56 EEST 2024, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 13%**

	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
ID: 126267 Назва: МКР Універсальна система автоматизованого тренування та моніторингу моделей машинного навчання для SAAS платформи in silico розробки ліків. On-premise деплоймент на інфраструктурі фармакологічних компаній за допомогою оркестратора Airflow Додано в БД: 2024-05-15 Автора: Возняк В.З. Керівники: Павлова О.О. Консультанти: Опоненти:	96361	906	1998 (2%)	32 (4%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Возняк Володимир Зіновійович

Тема: Універсальна система автоматизованого тренування та моніторингу моделей Машинного Навчання для SAAS платформи in silico розробки ліків. On-premise деплоймент на інфраструктурі фармакологічних компаній за допомогою оркестратора Airflow.

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг дипломної роботи:

Кількість сторінок записки 87 с.

1. Короткий зміст роботи та прийнятих рішень: створено автоматизований пайплайн тренування моделей машинного навчання на біологічних даних, який володіє властивостями модульності, детального налаштування конфігурації, паралелізації, швидкого масштабування та деплойменту у двох сценаріях - On-Cloud та On-Premise. Розроблений пайплайн був інтегрований в якості сервісу в існуючу SAAS платформу in silico розробки ліків.

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі проаналізовано існуючі рішення у галузі автоматизації пайплайнів машинного навчання. У другому розділі побудовано математичну модель основних компонент пайплайну для автоматичного тренування моделей машинного навчання. У третьому розділі розглянуто методи реалізації версії пайплайну для On-Premise деплойменту на інфраструктурі фармакологічних компаній за допомогою оркестратора Airflow. У четвертому розділі Інтеграція пайлайну в SAAS платформу для розробки ліків. Наукова новизна отриманих результатів полягає у тому, що вперше розроблено On-Premise версію повністю автоматизованого пайплайну для тренування моделей машинного навчання на біологічних даних; розроблений

пайплайн впроваджено в SAAS платформу для in silico розробки ліків; за допомгою створеного пайплайну натреновано моделі для передбачення ADME-Tox параметрів молекул та інтегровано їх у SAAS платформу.

4. Позитивні сторони роботи: отримання трьох пунктів наукової новизни

5. Негативні сторони роботи:

6.Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на високому науково-технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: відмінно.

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) _____

Барлов Олександр Володимирович, к.т.н., професор, завідувач кафедри комп'ютерних наук ХНУ

"17" травня 2024 р. _____(підпис)

Завідувачу кафедри КІІС
д-р.техн.наук, проф. Говорущенко Т. О.

Возняка Володимира Зіновійовича
ПІБ здобувача вищої освіти

ФІТ, 2 курсу, групи КІ2м-22-2

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

20 травня 2024 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Універсальна система автоматизованого тренування та моніторингу моделей машинного навчання для SAAS платформи in silico розробки ліків. On-premise деплоймент на інфраструктурі фармакологічних компаній за допомогою оркестратора Airflow

Автор: Возняк Володимир Зіновійович

Спеціальність: 123 – Комп'ютерна інженерія

Освітня програма: освітньо-наукова

Науковий керівник: д.ф., старший викладач Павлова О.О.

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
2) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з джерелами на один фрагмент речення;
3) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості Unicheck, складає 1.53% і адресується до 144 першоджерел; та системою Anti-Plagiarism складає 1%, що, з урахуванням наведених обгрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ О. О. Павлова

Гарант ОП _____ О. С. Савенко

Завідувач кафедри КІС _____ Т. О. Говорущенко