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-Cloud деплоймент на Kubernetes кластерах хмарних провайдерів»

КвРКІП. 013042.17.01.01 ПЗ

Виконав: студент 2 курсу, група КІ2м-22-2 _____ Островський З.Ю.
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

Островському Захару Юрійовичу
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. On-Cloud деплоймент на Kubernetes кластерах хмарних провайдерів

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-Cloud 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)

6. Консультанти розділів кваліфікаційної роботи магістра

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

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 _____ Островський З.Ю.
Підпис Ініціали, прізвище

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

РЕФЕРАТ

Тема кваліфікаційної роботи магістра: Універсальна система автоматизованого тренування та моніторингу моделей машинного навчання для SAAS платформи in silico розробки ліків. On-Cloud деплоймент на Kubernetes кластерах хмарних провайдерів.

Автор роботи: Островський Захар Юрійович.

Керівник роботи: Павлова Ольга Олександрівна.

Пояснювальна записка: 84 с., 22 рис., 14 табл., 3 дод., 60 джерел.

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-Cloud інфраструктур.

Метою кваліфікаційної роботи магістра є створення автоматизованого пайплайну 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-Cloud версію повністю автоматизованого пайплайну для тренування моделей машинного навчання на біологічних даних;

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

− розроблено 3 сімейства скорингових функцій для вибору найкращої фінальної моделі та проведено детальний аналій їх результативності.

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 матиматичну модель основних компонентів пайплайну. Розглянуто принцип роботи таких алгоритмів класичного машинного начання, 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-Cloud деплойменту на Kubernetes кластерах хмарних провайдерів. Також наводиться порівняння різних технологій для реалізації поставленої задачі і обгрунтовується фінальне рішення.

У четвертому розділі міститься опис створення скорингових функцій для вибору найкращої моделі отриманої автоматичним навчанням. Проведено ретельне порівняння запропонованих сімейств скорингових функцій для задач бінарної класифікації та регресії, та обрано найкращі скорингові функції та значення їх параметрів для задач регресії та класифікації по критерії максимальної робастності натренованих моделей.

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-Cloud - Он-клауд (хмарне розміщення)

RDS - Relational Database Service (сервіс реляційних баз даних)

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

HTTP - Hypertext Transfer Protocol (протокол передачі гіпертексту)

SQL - Structured Query Language (мова структурованих запитів)

URL - Uniform Resource Locator (уніфікований локатор ресурсів)

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

ВСТУП

Сучасний світ даних та технологій постійно розвивається, пропонуючи все нові виклики та можливості у сфері машинного навчання та штучного інтелекту. В контексті стрімкого прогресу, важливість ефективного та швидкого впровадження машинного навчання у різноманітні промислові та дослідницькі процеси не може бути переоцінена. Особлива увага у цьому контексті приділяється підходам автоматизації машинного навчання, які відомі під загальною назвою AutoML.

Ця дипломна робота присвячена розробці та аналізу ефективних пайплайнів машинного навчання з використанням сучасних хмарних технологій. Зокрема, акцент робиться на використанні Kubernetes та KubeFlow для створення масштабованих та гнучких рішень, що дозволяють значно оптимізувати процеси розробки та впровадження моделей машинного навчання у виробництво.

Дослідження охоплює порівняння можливостей основних хмарних платформ, таких як AWS та GCP, з метою визначення найбільш ефективного середовища для розгортання Kubernetes кластерів. Основна увага приділяється аналізу та оптимізації вартісної ефективності та обчислювальної продуктивності, що є критично важливим для підтримки великих даних і складних обчислювальних задач, характерних для сучасного машинного навчання.

Робота також включає розробку та детальний опис AutoML пайплайну, що дозволяє автоматизувати процеси від вибору моделі до її тестування та оптимізації. Реалізація пайплайну розглядається на прикладі конкретних задач класифікації та регресії, з подальшим аналізом отриманих результатів та оцінкою ефективності рішень.

Враховуючи широкий спектр можливостей інтеграції та налаштування, який надають розглянуті технології, робота вносить вагомий внесок у розуміння того, як можна забезпечити більшу гнучкість і масштабованість у домені машинного навчання.

Метою кваліфікаційної роботи магістра є створення автоматизованого пайплайну 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-Cloud версію повністю автоматизованого пайплайну для тренування моделей машинного навчання на біологічних даних;

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

− розроблено 3 сімейства скорингових функцій для вибору найкращої фінальної моделі та проведено детальний аналій їх результативності.

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) виступає потужним інструментом для автоматизації процесу застосування машинного навчання до реальних проблем від початку і до кінця. Основна мета AutoML полягає у мінімізації або навіть усуненні потреби в кваліфікованому людському втручанні, тим самим прискорюючи розгортання та впровадження моделей машинного навчання. Пайплайн AutoML складається з послідовності етапів, кожен з яких відповідає за конкретний аспект робочого процесу машинного навчання. Основні компоненти пайплайну AutoML включають попередню обробку даних, інженерію ознак, вибір моделі та налаштування гіперпараметрів. У цьому розділі описуються основні складові пайплайну AutoML та пояснюється роль, яку вони грають у автоматизації процесу машинного навчання.

1.1.1 Попередня обробка даних

Попередня обробка даних є важливою фазою пайплайну AutoML, яка готує сирові дані для подальшого аналізу. Цей крок часто включає завдання, такі як очищення даних, обробка відсутніх значень, трансформація даних та нормалізація даних. Мета полягає у тому, щоб забезпечити високу якість даних, які подаються на наступні етапи пайплайну, і уникнути розбіжностей, які можуть негативно вплинути на продуктивність моделі.

Очищення даних: процес ідентифікації та виправлення (або видалення) помилок та невідповідностей в даних для покращення їх якості.

Обробка відсутніх значень: техніки обробки відсутніх даних включають імпутацію, видалення та алгоритмічні зміни.

Трансформація даних: це включає операції, такі як масштабування, нормалізація та кодування категоріальних змінних.

Вибір та зменшення ознак: вибір найбільш відповідних ознак або зменшення розмірності даних.

1.1.2 Інженерія ознак

Інженерія ознак є мистецтвом створення нових ознак з існуючих даних для покращення продуктивності моделі. У пайплайні AutoML цей процес автоматизовано для швидкого виведення ознак, які відображають основні закономірності в даних, без потреби в галузевій експертизі.

Цей етап включає такі компоненти:

- feature extraction – отримання нових ознак з існуючих даних.
- feature selection – вибір підмножини найбільш інформативних ознак.
- feature encoding – кодування категоріальних ознак у числовий формат.
- feature scaling – забезпечення того, щоб ознаки були на схожому масштабі, щоб запобігти упередженості в моделі.

1.1.3 Вибір моделі

Вибір моделі є ключовим компонентом пайплайну AutoML, у якому вибирається найбільш підходяща модель машинного або глибокого навчання на основі поставленої проблеми. На цьому етапі часто використовуються техніки мета-навчання або ансамблевого навчання для визначення найкращої моделі.

Оцінка моделі: оцінка продуктивності різних моделей на заданих даних.

Порівняння моделей: порівняння різних моделей для вибору тієї, яка має найвищу продуктивність.

Мета-навчання: використання минулого досвіду для вибору моделей для нових наборів даних.

Ансамблеве навчання: комбінування прогнозів кількох моделей для покращення загальної продуктивності.

1.1.4 Тюнінг гіперпараметрів

Тюнінг гіперпараметрів - це процес оптимізації гіперпараметрів обраної моделі для покращення її продуктивності. У AutoML налаштування гіперпараметрів виконується автоматично, часто з використанням складних технік оптимізації, таких як байєсівська оптимізація, пошук сітки або випадковий пошук.

Існують наступні методи для пошуку оптимальних гіперпараметрів:

- пошук по сітці: вичерпний пошук по заданій сітці гіперпараметрів;
- випадковий пошук: випадкове вибірка гіперпараметрів з визначеного простору;
- байєсівська оптимізація: техніка оптимізації на основі ймовірнісної моделі;
- еволюційні алгоритми: алгоритми оптимізації, натхненні процесом природного добору.

1.1.5 Візуалізація даних

Візуалізація даних є невід'ємним компонентом у пайплайні AutoML, що сприяє більш інтуїтивному розумінню даних та продуктивності моделі. Вона охоплює створення та вивчення візуального представлення даних, яке може варіюватися від простих діаграм розсіювання до складних багатовимірних діаграм. За допомогою візуалізацій зацікавлені сторони можуть отримати уявлення про дані, інтерпретувати результати моделі та приймати обґрунтовані рішення на основі цих візуальних представлень.

Дослідницький аналіз даних (EDA) - це використання візуалізацій для дослідження даних, виявлення закономірностей та розуміння їх структури.

Візуалізація продуктивності моделі: візуальні представлення метрик продуктивності моделі та результатів оцінювання.

Візуалізація важливості ознак: відображення значимості різних ознак у прогностичних моделях.

Аналіз помилок: візуалізація помилок, допущених моделлю, для виявлення можливих напрямків поліпшення.

1.1.6 Моніторинг продуктивності

Моніторинг продуктивності є вирішальним компонентом, який забезпечує те, щоб розгорнуті моделі утримували високий рівень точності та надійності з часом. Це включає в себе постійне відстеження продуктивності моделі, виявлення будь-якого знецінення та ініціювання повторного навчання або налаштування за потреби. Цей компонент є життєво важливим для підтримання надійності та ефективності рішень машинного навчання в виробничому середовищі.

Метрики оцінки продуктивності моделі: використання метрик, таких як точність, специфічність, селективність, F1 та інші.

Виявлення зміщення: виявлення змін у розподілі даних, які можуть вплинути на продуктивність моделі.

Зворотні зв'язки: збір відгуків на прогнози моделі для покращення її майбутньої точності.

Сповіщення та звітування: встановлення механізмів для сповіщення та звітування про спад точності моделі або інші проблеми.

1.1.7 Деплоймент моделі

Етап деплоймента моделі – це коли навчена модель стає доступною для виконання прогнозів на нових даних. Це важливий компонент пайплайну AutoML, який позначає перехід від розробки до продуктового етапу. Процес деплоїнга повинен забезпечити доступність моделі, масштабованість та здатність

обробляти дані в реальному часі або пакетну обробку відповідно до вимог застосування.

Деплоймент включає врахування наступних аспектів.

Платформи деплойменту: використання платформ або фреймворків, які сприяють розгортанню моделі, таких як хмарні рішення або сервери на власній території.

Контроль версій: управління різними версіями моделі для забезпечення відтворюваності та відстежуваності.

Масштабованість: забезпечення того, що рішення розгортання може обробляти збільшення обсягів даних та користувацьких запитів.

Моніторинг та обслуговування: постійний моніторинг розгорнутої моделі та виконання необхідного обслуговування для забезпечення оптимальної продуктивності.

1.1.8 Висновки

У таблиці 1.1 наведено відносну важливість та вплив кожного компонента у AutoML пайплайні, а також рівень досяжної автоматизації. Очевидно, що кожний компонент відіграє важливу роль у забезпеченні успіху процесу AutoML, високий рівень автоматизації сприяє швидкому та ефективному деплойменту моделей машинного навчання.

Таблиця 1.1 – Важливість та вплив кожного компонента у AutoML пайплайні

Складова	Важливість	Вплив на якість метрик	Рівень автоматизації
Препроцесинг даних	Високий	Високий	Високий

Кінець Таблиці 1.1 – Важливість та вплив кожного компонента у AutoML пайплайні

Feature Engineering	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
Деплоймент моделі	Високий	Відсутній	Високий

Візуалізація даних, хоча й помірно автоматизована, значно сприяє кращому розумінню даних та продуктивності моделі. З іншого боку, моніторинг продуктивності та деплоймент моделі високо автоматизовані та життєво важливі для успішної операціоналізації моделей машинного навчання у реальних сценаріях.

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

Наведемо огляд низки статтей, які описують реалізацію систем використання машинного навчання у відкритті та розробці ліків.

У [1] автори розробили повну стек-архітектуру хмари від IaaS до SaaS, засновану на відкритих технологіях, включаючи інтеграцію високопродуктивних передач даних у всі типи сервісів. Особливістю є стратегія динамічного масштабування PaaS для створення та виконання біоінформатичних та біомедичних робочих потоків із використанням Galaxy. Це демонструє здатність платформи до швидкого масштабування ресурсів та її придатність для двох

визначених випадків використання у біоінформатиці та біомедичних дослідженнях.

Автори [2] розглядають використання хмарних обчислень у комплексному аналізі мульти-омічних даних, особливо зосереджуючись на використанні концепцій віртуалізації та контейнеризації. Вони показують, як хмарні обчислення можуть сприяти легкому масштабуванню та створенню незалежних аналітичних пайплайнів для омічних даних, а також розкривають перевірену концептуальну модель універсального рішення на прикладі програмного пакету Bioconductor.

Між іншим у [3] обговорюють інвестиції наукової спільноти в хмарні біомедичні платформи, вказуючи на необхідність зосередження на мікросервісах для підтримки FAIR-принципів. Вони пропонують відійти від самодостатніх платформ і репозиторіїв даних на користь інтероперабельних мікросервісів. Це підход має на меті поліпшити інтеграцію та доступ до модульних даних, сприяючи створенню ефективнішого екосистеми веб-сервісів для біомедичного аналізу та отримання даних.

У [4] AIDDISON пропонує зручну, безпечну веб-платформу для відкриття ліків, інтегруючи генеративні моделі, прогнозування властивостей ADMET, пошуки в широких хімічних просторах та молекулярне докінг. Це забезпечує доступ до переваг AI/ML у відкритті ліків для науковців, які не є комп'ютерними експертами, демонструючи це на прикладі ідентифікації цінних молекул для оптимізації лідів. AIDDISON - це платформа від фарм гіганта Merck, по суті, іще в процесі їхньої розробки, яка робить акцент на штучно згенеровані молекули і в загальному відтворює воркфлоу, який був реалізований у RECEPTOR.AI в той момент, коли Merck тільки починав розробку своєї платформи.

У [5] статті обговорюється, як хмарні обчислення можуть допомогти життєвому циклу ML у відкритті ліків. Автори розглядають можливості контейнеризації, наукових робочих потоків та впровадження концепції MLOps для забезпечення відтворюваності та надійності моделювання ML. Вони також обговорюють застосування ML на приватних, чутливих і регульованих даних, а

хмарні обчислення та федеративне навчання можуть сприяти співпраці у відкритті ліків.

У статті [6] розглядаються інновації в обчислювальних технологіях та підходах, зокрема в хмарних обчисленнях, які пропонують обіцяюче, недороге та гнучке рішення у галузі біоінформатики. Розглядаються переваги хмарних обчислень в молекулярному моделюванні, аналітиці омічних даних та інтеграції, аналізі та інтерпретації фенотипічних даних.

1.3 Застосування пайплайну для тренування моделей машинного навчання на біологічних даних

Представлення молекул у форматі, який зручний для обробки алгоритмами машинного навчання, є ключовим етапом у таких областях, як медична хімія, розробка ліків та біоінформатика. Існує кілька основних типів представлення молекул, кожен з яких має свої переваги та обмеження. Основні типи включають: SMILES, Graph-Based Representations, Fingerprints, та 3D Structure Representations. На рисунку 1.1 зображено основні варіанти репрезантації молекул.

SMILES – це спосіб опису структури молекули за допомогою короткої ASCII рядка.

Переваги SMILES:

- компактність та простота зберігання;
- широка підтримка в хімінформатичних інструментах.

Недоліки SMILES:

- втрата деяких просторових характеристик молекули;
- можлива неоднозначність у представленні.

Інше поширене представлення – графове, де молекули представлені у вигляді графів, де атоми є вершинами, а зв'язки - ребрами.

Переваги графового представлення:

- зберігають просторову структуру молекули;
- підтримуються алгоритмами графових нейронних мереж.

Недоліки графового представлення:

– більша складність обчислень;

– вимагає більше ресурсів для зберігання та обробки.

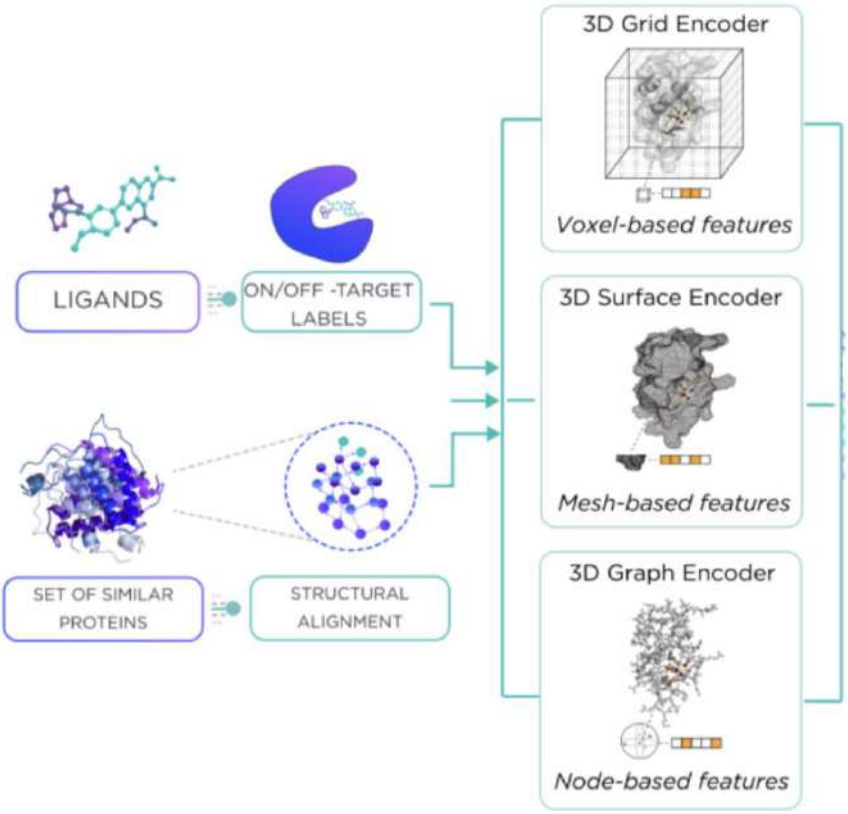


Рисунок 1.1 – Основні варіанти репрезентацій молекул для білків і лігандів [19]

Відбитки – це бінарні або числові вектори, які відображають наявність або відсутність певних хімічних структур у молекулі.

Переваги відбитків:

– компактність представлення;

– легкість порівняння молекул;

Недоліки відбитків:

– втрата інформації про просторову структуру;

– можливість "колізій" при збігу відбитків різних молекул.

Також можливим є тривимірне представлення молекул, де вказані точні координати атомів у просторі.

Переваги тривимірного представлення:

- найбільш точне представлення просторової структури молекули;
- важливе для розуміння молекулярної взаємодії.

Недоліки тривимірного представлення:

- висока обчислювальна складність;
- великий об'єм даних для зберігання.

У таблиці 1.2 наведено основні типи представлення молекул.

Таблиця 1.2 – Основні типи представлення молекул

Критерій / Тип представлення	SMILES	Графові Представлення	Відбитки (Fingerprints)	3D Структурні Представлення
Опис	ASCII рядок, що описує структуру молекули	Молекула у вигляді графа, атоми - вершини, зв'язки - ребра	Бінарні/числові вектори, що відображають наявність хімічних структур	Точні тривимірні координати атомів молекули
Просторова інформація	Обмежена	Висока	Обмежена	Висока
Обчислювальна складність	Низька	Середня/Висока	Низька	Висока
Підтримка ML-алгоритмів	Широка	Спеціалізовані алгоритми (напр., GNN)	Широка (особливо для класифікаційних задач)	Спеціалізовані алгоритми, потребують великих обчислень

Кінець таблиці 1.2 – Основні типи представлення молекул

Точність представлення	Середня	Висока	Середня	Висока
Сфера застосування	Швидкий скринінг, базові аналізи	Детальний аналіз структури та властивостей	Швидке порівняння та відбір	Дослідження просторових взаємодій та динаміки
Втрата інформації	Можлива	Мінімальна	Можлива	Мінімальна

Вибір представлення молекули для машинного навчання залежить від конкретних цілей дослідження та вимог до точності. SMILES та відбитки є ефективними для швидкого скринінгу та порівняння великої кількості молекул, але вони можуть не враховувати всі нюанси просторової структури. Графові представлення найкраще підходять для завдань, де важливі деталі зв'язків та структури молекул. 3D структурні представлення забезпечують найбільшу точність і важливі для розуміння просторових взаємодій, наприклад, у дослідженні зв'язування білків. У кінцевому рахунку, вибір методу залежить від специфіки задачі, доступних даних та обчислювальних ресурсів.

У контексті молекулярних даних, AutoML пайплайн може значно спростити та прискорити процес відкриття нових ліків, прогнозування властивостей молекул та інших біоінформатичних досліджень. Наприклад:

– прогнозування біологічної активності – автоматизація процесу виявлення молекул, які можуть бути потенційно активними проти певних біологічних мішеней;

– оптимізація лікарських засобів – аналіз та прогнозування властивостей лікарських засобів, таких як розчинність, токсичність, біодоступність;

– вивчення молекулярних взаємодій – розробка моделей для вивчення взаємодій між різними молекулами та білками.

1.4 Постановка задачі та вибір технологій для реалізації

У рамках даної магістерської роботи ставиться завдання створення універсальної системи автоматизованого тренування та моніторингу моделей машинного навчання для SAAS платформи in silico розробки ліків. Основна мета полягає у розробці автоматизованого пайплайну, який буде спрямований на роботу з біологічними даними, забезпечуючи модульність, детальне налаштування, паралелізацію, швидке масштабування та гнучкий деплоймент у двох режимах - On-Cloud та On-Premise. Цей пайплайн має бути інтегрований в якості сервісу в існуючу SAAS платформу, спеціалізовану на in silico розробці ліків.

On-Cloud розгортання та Kubernetes. використання Kubernetes для оркестрації контейнерів дозволяє забезпечити високу доступність, масштабованість та ефективне управління ресурсами. Kubernetes є оптимальним рішенням для On-Cloud деплойменту, оскільки він підтримується більшістю хмарних провайдерів (таких як AWS і GCP, які мають власні сервіси для розгортання середовища Kubernetes), є гнучким і потужним інструментом та дозволяє просто масштабуватися при потребі.

Автоматизовані пайплайни машинного навчання: використання Kubeflow, для створення гнучких пайплайнів з детальною конфігурацією. Цей інструмент дозволяє автоматизувати процеси тренування, оцінки та розгортання моделей машинного навчання. AutoML Пайплайн має складатися з компонент, які відповідають за наступні задачі: попередню обробку даних, інженерію ознак, вибір моделі та налаштування гіперпараметрів.

Вибір технічого стеку для машинного навчання: представлення молекул у форматі відбитків (Fingerprints) разом із використанням моделей класичного Машинного Навчання - є оптимальним рішенням для AutoML пайплайну з огляду

на типові розміри навчальних вибірок та практичність автоматизації підбору найкращого алгоритму. Тому доцільним є використання мови програмування Python та бібліотеки scikit-learn.

Інтеграція з SAAS платформою: розробка API та сервісів для інтеграції розробленого пайплайну з існуючою SAAS платформою. Це дозволить забезпечити ефективний обмін даними та результатами між пайплайном та платформою. Також необхідно буде реалізувати спеціальний інтерфейс, який довзолить користуватися натренованими моделями всередині платформи.

Для обробки біологічних даних буде використана спеціалізована бібліотека для Python - RDKit.

Контейнеризація з Docker: Docker є ключовим компонентом у створенні ефективного пайплайну машинного навчання. Він дозволяє пакувати додатки та їх залежності у стандартизовані контейнери, що спрощує деплоймент, масштабування та управління.

Ізоляція та відтворюваність: Docker забезпечує ізоляцію додатків, гарантуючи, що вони працюватимуть однаково в різних середовищах. Це особливо важливо для наукових обчислень та експериментів у машинному навчанні, де відтворюваність результатів є критичною.

Сумісність з Kubernetes: Docker контейнери можуть легко оркеструватися за допомогою Kubernetes, що дозволяє автоматизувати розгортання, масштабування та управління додатками.

Стандартизація робочих процесів: використання Docker допомагає стандартизувати робочі процеси, знижуючи ризик "це працює на моєму комп'ютері" проблематики. Це важливо для великих команд, які працюють над складними проектами in silico розробки ліків.

Безпека та ізоляція ресурсів: Docker дозволяє ефективно розділяти ресурси та ізолювати процеси, забезпечуючи високий рівень безпеки, що є критично важливим для обробки конфіденційних біологічних даних.

1.5 Висновки

У цьому розділі було проаналізовано основні компоненти, які повинен містити AutoML пайплайн. Було наведено детальний аналіз та порівняння технологій, завдяки яким можна реалізувати кожен з компонентів. Також було описано, які існують способи представлення біологічних даних для їх використання у алгоритмах машинного навчання. Завершується цей розділ постановкою задачі і обгрунтованим вибором відповідних технологій.

2 МАТЕМАТИЧНА МОДЕЛЬ КОМПОНЕНТІВ СИСТЕМИ АВТОМАТИЗОВАНОГО ТРЕНУВАННЯ ТА МОНІТОРИНГУ МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ

2.1 Математична модель алгоритму вибору найкращих ознак

Вибір найкращих ознак (feature selection) в машинному навчанні - це процес визначення та вибору тих ознак з набору даних, які найкраще відображають завдання, що вирішується. Існує кілька основних методів відбору ознак, кожен з яких має свої особливості, переваги та недоліки. На рисунку 2.1 зображено процес визначення та відбору ключових ознак.

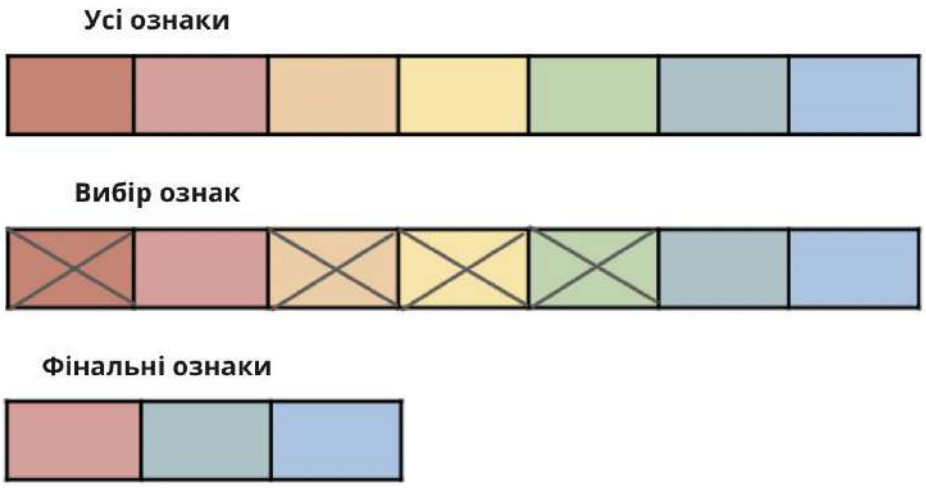


Рисунок 2.1 – Процес визначення та відбору ключових ознак

2.1.1 Фільтраційні методи

Суть: вибір ознак на основі їх статистичних характеристик. Переваги: простота обчислень; не вимагає тренування моделей. Недоліки: не враховує взаємодію між ознаками.

Приклади фільтраційних методів:

– кореляція Пірсона – міра лінійної залежності між двома змінними;

− хі-квадрат (chi-squared) тест – оцінює незалежність між категоріальними змінними;

− взаємна інформація (mutual information) – міра взаємної залежності між двома змінними; вимірює скільки інформації одна змінна містить про іншу;

− аналіз головних компонент (PCA) – зменшує розмірність даних, зберігаючи при цьому максимальну варіативність;

− ANOVA F-тест – використовується для визначення, чи є статистично значущі різниці між середніми значеннями більше двох груп.

Фільтраційні методи є потужним інструментом для первинного аналізу даних та відбору ознак. Вони дозволяють значно зменшити розмірність даних, при цьому зберігаючи важливу інформацію. Однак ці методи не враховують взаємодію між ознаками та модель машинного навчання, тому вони часто використовуються у поєднанні з іншими методами відбору ознак для досягнення оптимального результату.

2.1.2 Обгорткові методи

Опис: використання алгоритмів машинного навчання для оцінки важливості ознак.

Математичне формулювання: один з прикладів – метод випадкового лісу, де важливість ознаки I може бути оцінена через зниження точності моделі при видаленні цієї ознаки.

Переваги: враховує взаємодію між ознаками; часто дає кращі результати.

Недоліки: висока обчислювальна складність; ризик перенавчання.

2.1.3 Вбудовані методи

Опис: інтеграція процесу відбору ознак безпосередньо у процес навчання моделі.

Математичне формулювання: наприклад, використання регуляризації LASSO, де цільова функція моделі включає штраф за величину коефіцієнтів.

Переваги: ефективне визначення важливих ознак; зниження ризику перенавчання.

Недоліки: залежність від вибору моделі; може бути складним у налаштуванні.

2.1.4 Висновки

На рисунку 2.2 зображено методи вибору найкращих ознак. Кожен із цих методів відбору ознак має свої унікальні переваги та області застосування. Фільтраційні методи є простими та швидкими, але можуть пропускати важливі взаємодії між ознаками. Обгорткові методи забезпечують більш точний відбір, але вимагають значних обчислювальних ресурсів. Вбудовані методи інтегрують відбір ознак безпосередньо у процес навчання, пропонуючи збалансований підхід. Вибір методу залежить від конкретного завдання, доступних даних, та обчислювальних можливостей.

2.2 Математична модель алгоритмів машинного навчання

Алгоритми машинного навчання зазвичай можна описати за допомогою трьох основних компонентів: апроксимуючої функції, функції втрат та алгоритму оптимізації.

Апроксимуюча функція, часто називана просто "моделлю", це математична конструкція, яка намагається апроксимувати залежності між входами та виходами в навчальних даних. Формально, це може бути виражено як:

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

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

x – вектор вхідних атрибутів;

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

f – сама функція апроксимації.

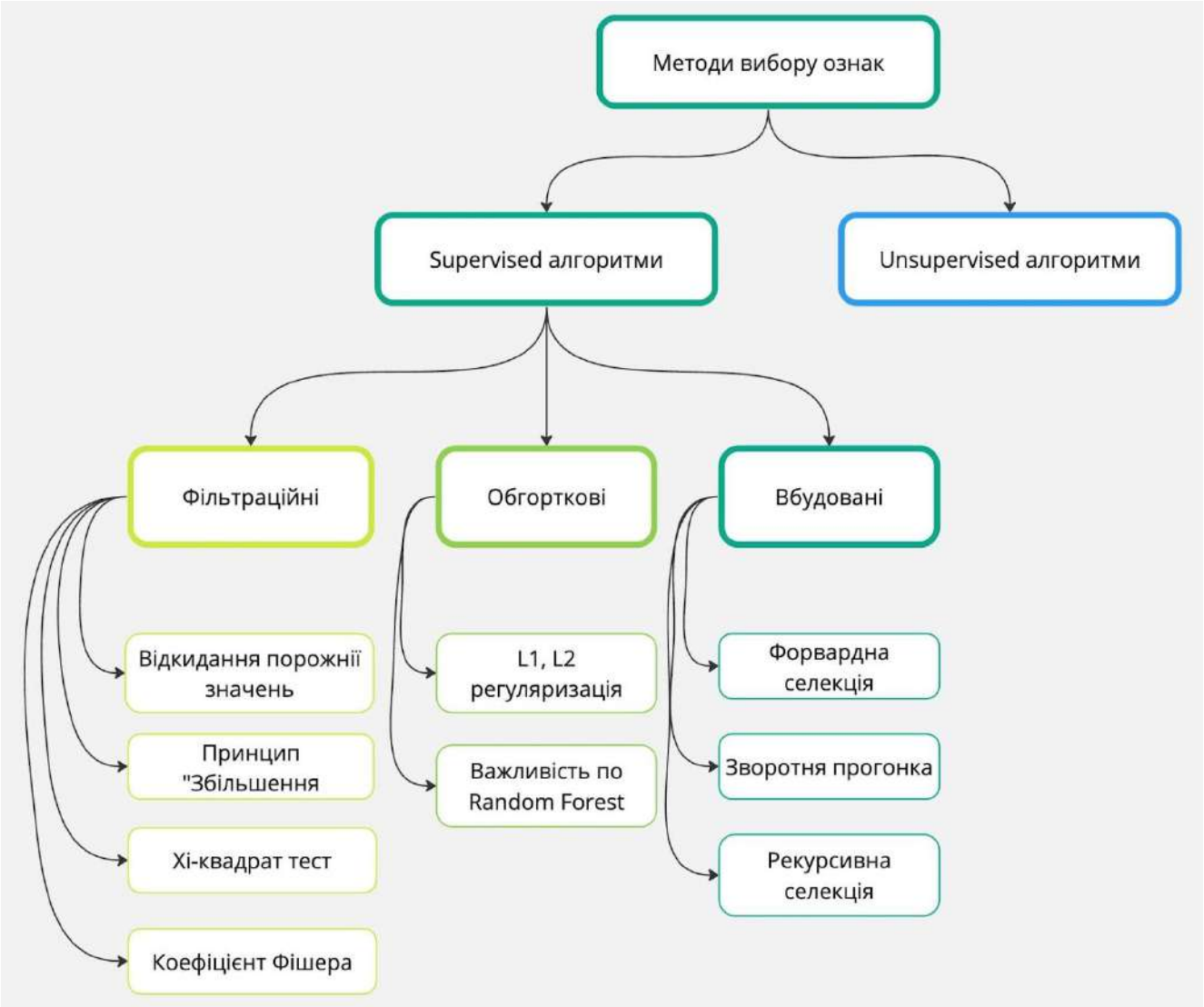


Рисунок 2.2 – Методи вибору найкращих ознак

Функція втрат оцінює, наскільки добре модель відображає реальні дані. Вона вимірює розбіжність між прогнозованими значеннями y* та істинними значеннями y. Метою навчання є мінімізація цієї функції втрат. Наприклад, для задачі регресії часто використовують середньоквадратичну помилку (MSE):

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

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.

Алгоритм оптимізації використовується для знаходження оптимальних значень параметрів моделі θ, які мінімізують функцію втрат. Це може бути досягнуто за допомогою різних методів, таких як градієнтний спуск, стохастичний градієнтний спуск (SGD), методи другого порядку тощо. Наприклад, в стохастичному градієнтному спуску параметри моделі оновлюються на кожному кроці за наступним правилом:

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

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.

Загальна математична модель алгоритму машинного навчання охоплює взаємодію між апроксимуючою функцією, функцією втрат та алгоритмом оптимізації. Ці компоненти разом визначають, як модель буде навчатися з даних та адаптуватися для вирішення конкретної задачі. Вибір конкретних формулювань для кожного з цих компонентів залежить від завдання, типу даних та конкретних цілей, які переслідує дослідник або інженер. Нижче наведено математичні для найпопулярніших алгоритмів машинного навчання.

2.2.1 Лінійна регресія

Сфера застосування: прогнозування неперервних змінних, наприклад, прогнозування цін на житло, прогнозування попиту.

Інтуїтивне пояснення: лінійна регресія намагається побудувати пряму лінію, яка найкращим чином відображає залежність між змінними.

Принцип роботи: мінімізація суми квадратів різниць між спостережуваними та прогнозованими значеннями.

Переваги: простота інтерпретації, швидкість обчислень.

Недоліки: припускає лінійність залежностей, чутлива до викидів.

Чутливість до аномалій: висока.

Обмеження: не підходить для моделювання не лінійних залежностей.

Швидкодія: висока.

2.2.2 Логістична регресія

Сфера застосування: бінарна класифікація, наприклад, визначення імовірності захворювання, прогнозування відтоку клієнтів.

Інтуїтивне пояснення: логістична регресія оцінює ймовірності належності до одного з двох класів.

Принцип роботи: мінімізація логістичної функції втрат.

Переваги: видає ймовірності класів, простота інтерпретації.

Недоліки: припускає лінійність рішення, обмежена до бінарної класифікації.

Чутливість до аномалій: середня.

Обмеження: не підходить для нелінійних задач, обмежена до бінарної класифікації.

Швидкодія: висока.

На рисунках 2.3 і 2.4 зображено порівняння лінійної та логістичної регресій.

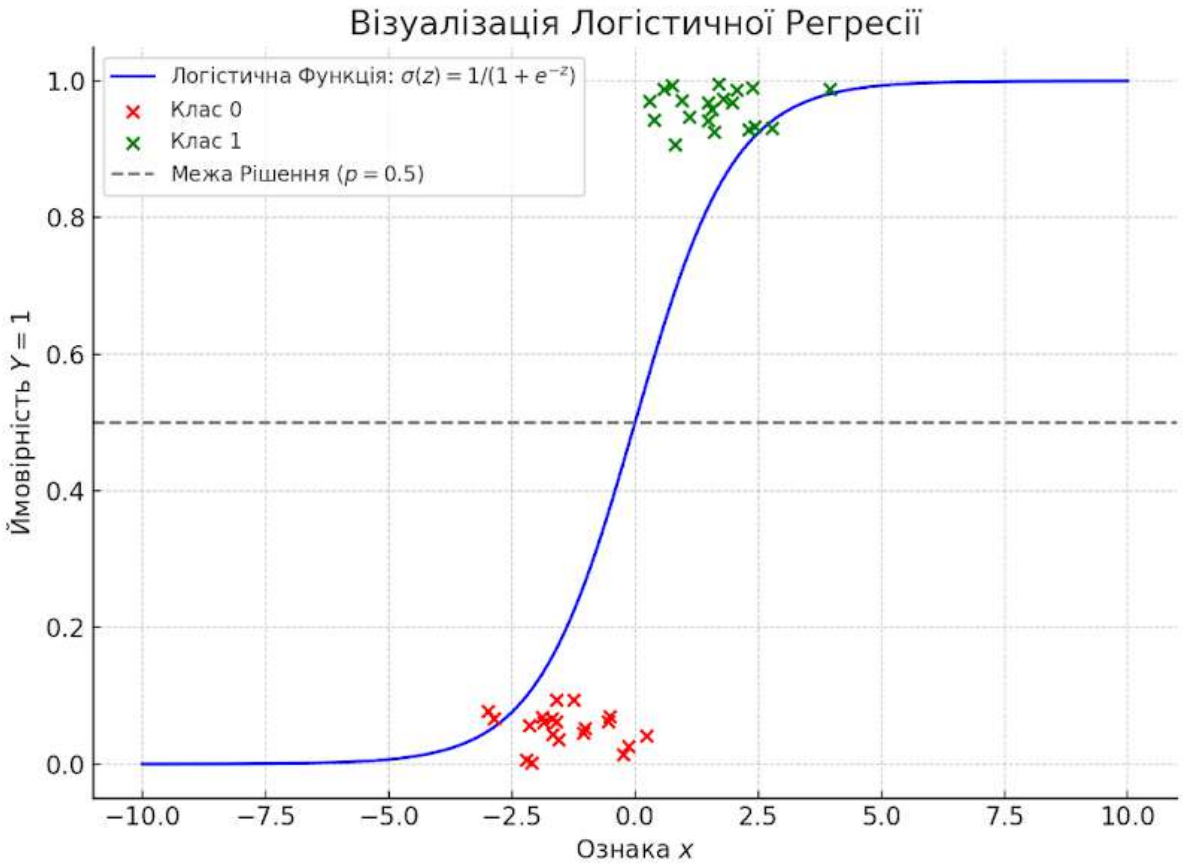


Рисунок 2.3 – Візуалізація логістичної регресії

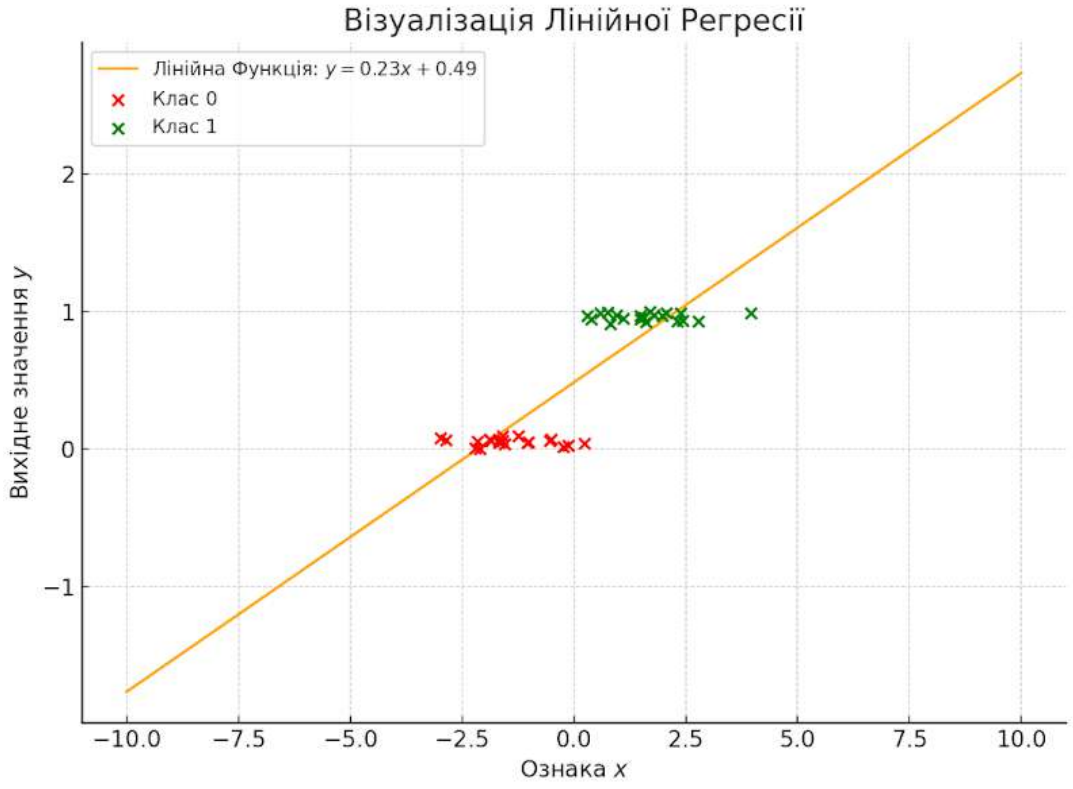


Рисунок 2.4 – Візуалізація лінійної регресії

2.2.3 Дерево рішень

Сфера застосування: класифікація та регресія, наприклад, визначення виду рослини, прогнозування вартості нерухомості.

Інтуїтивне пояснення: дерево рішень робить рішення, рухаючись від кореня до листків, вибираючи шляхи на основі ознак.

Принцип роботи: рекурсивне розділення даних з використанням критеріїв розділення.

Переваги: легкість інтерпретації, здатність до моделювання не лінійних залежностей.

Недоліки: схильність до перенавчання, велика чутливість до варіацій у даних.

Чутливість до аномалій: висока.

Обмеження: може створювати складні дерева, які важко інтерпретувати.

Швидкодія: залежить від глибини дерева та кількості ознак.

Математична модель: рекурсивне розбиття простору ознак на підпростори згідно з критерієм найкращого розділення (наприклад, за індексом Джині або ентропією).

Цільова функція: мінімізація критерію неоднорідності в кожному листку дерева.

2.2.4 Випадковий ліс

Сфера застосування: класифікація та регресія, використовується в багатьох сферах, включаючи біомедицину, фінанси та екологію.

Інтуїтивне пояснення: ансамбль рішаючих дерев, які тренуються на різних підвибірках даних, а їхні прогнози усереднюються.

Принцип роботи: кожне дерево тренується на випадковій підвибірці ознак та прикладів, а потім усі дерева голосують за кінцевий прогноз.

Переваги: зниження ризику перенавчання порівняно з одним рішаючим деревом, висока точність прогнозування.

Недоліки: великі вимоги до обчислювальних ресурсів, важкість інтерпретації.

Чутливість до аномалій: низька.

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

Швидкодія: залежить від кількості дерев та глибини кожного дерева.

Математична модель: ансамбль рішаючих дерев, де кожне дерево тренується на випадковій підвибірці даних та ознак.

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

2.2.5 Одношаровий перцептрон

Сфера застосування: бінарна класифікація, розпізнавання образів.

Інтуїтивне пояснення: найпростіша нейронна мережа, яка вирішує, чи активувати вихід на основі суми вагових входів.

Принцип роботи: якщо сумарний вхід перевищує певний поріг, нейрон активується, інакше – ні.

Переваги: простота реалізації та тренування.

Недоліки: може вирішувати лише лінійно роздільні задачі.

Чутливість до аномалій: висока.

Обмеження: не підходить для не лінійно роздільних задач.

Швидкодія: висока.

2.2.6 Багатошаровий перцептрон (MLP)

Сфера застосування: класифікація, регресія, розпізнавання образів, обробка природної мови.

Інтуїтивне пояснення: нейронна мережа з одним або кількома прихованими шарами, здатна моделювати не лінійні залежності.

Принцип роботи: кожен нейрон виходить з вагової суми своїх входів, яка потім проходить через не лінійну функцію активації.

Переваги: здатність моделювати складні не лінійні залежності.

Недоліки: схильність до перенавчання, великі вимоги до обчислювальних ресурсів.

Чутливість до аномалій: залежить від архітектури та регуляризації.

Обмеження: може бути складним у налаштуванні та тренуванні.

Швидкодія: залежить від кількості шарів та нейронів.

На рисунку 2.5 зображено візуалізацію шарів MLP архітектури.

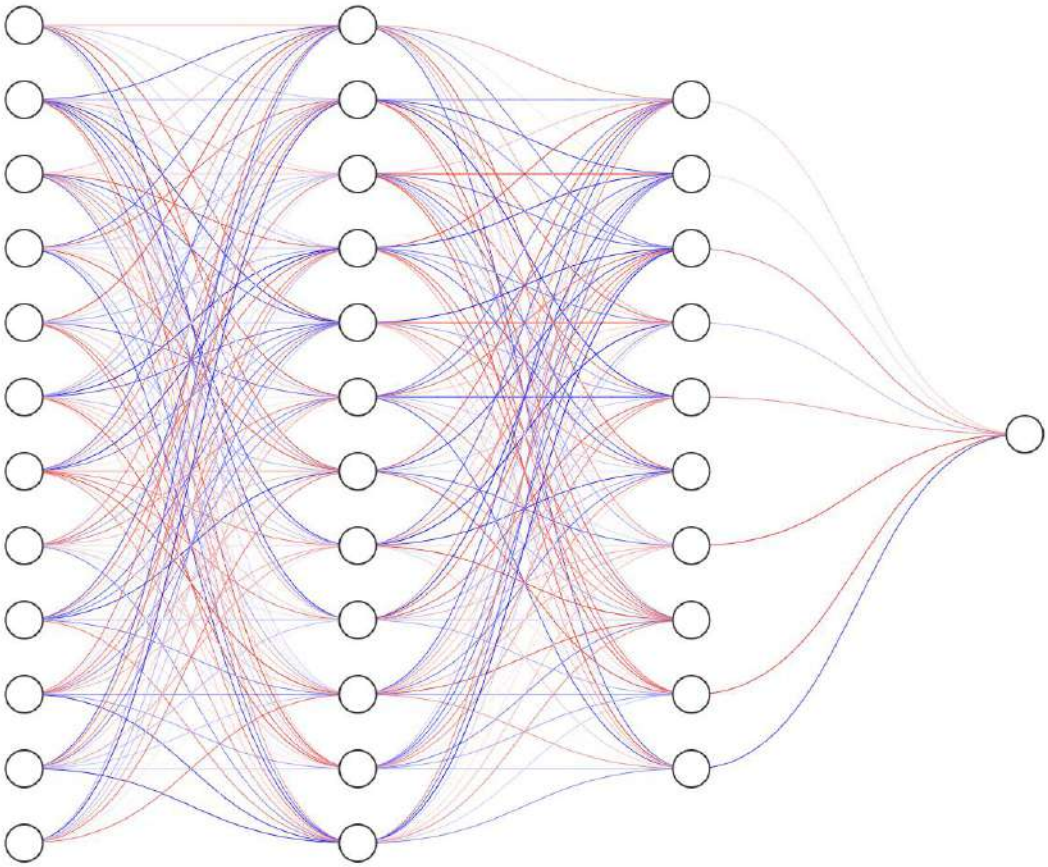


Рисунок 2.5 – Візуалізація багатошарового перцептрону

2.2.7 Графова нейронна мережа

Щоб побудувати математичну модель графових нейронних мереж, необхідно спочатку дати визначення “графу” та його властивостей. Розглянемо граф, який складається лише з набору вершин без зв'язків між ними (ребер). Нехай $x_i \in R^k$ – представляє ознаки вершини i. Об'єднуючи ці ознаки в матрицю, ми отримуємо матрицю ознак вершин наступної розмірності $n\ x\ k$:

$$X = (x_1, \dots, x_n)^T. \#(2.4)$$

Важливо зазначити, що перестановка вершин змінює матрицю ознак, тому вихід нейронної мережі повинен бути незалежним від порядку вершин. Для цього ми вводимо поняття інваріантності та еквіваріантності. Функція $f(X)$ є інварінтною до перестановок, якщо для будь-якої матриці перестановок P, виконується

$$f(PX) = f(X). \#(2.5)$$

Приклад даної моделі:

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

де ψ та ϕ – це функції, з параметрами, які навчаються, а функція суми може бути замінена на функцію середнього, мінімуму чи максимуму.

Визначення еквіваріантності для функції f(X) наступне, якщо для всіх P матриць перестановок виконуєтсья рівність:

$$f(PX) = Pf(X). \#(2.7)$$

Побудуємо еквіваріантну функцію обробки ознак вершин незалежно від послідовності зазначення нод у графі, використовуючи такий прийом:

$$h_i = \psi(x_i), \#(2.8)$$

де x_i – вхідні признаки;

h_i – вихідні нові признаки;

ψ – еквіваріантна функція.

Додатково визначається інваріантна функцію, яка обробляє еквіваріантні та переводить граф у простір чисельних векторів (які характеризують повний початковий граф):

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

де ψ – еквіваріантна функція;

ϕ – інваріантна функція.

Нехай маємо граф, що складається з множини вершин, а також граней між ними, де $E \subseteq VxV$. Для роботи з ними використовують матрицю суміжності:

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

Це збергігає еквіваріантність та інварінтність, проте тут для граней вони набудуть іншої форми.

Еквіваріантність:

$$f(PX, PAP^T) = Pf(X, A), \#(2.11)$$

де P – матриця перестановок;

A – матриця суміжності;

X – матриця признаків.

Інваріантність:

$$f(PX, PAP^T) = f(X, A), \#(2.12)$$

де P – матриця перестановок;

A – матриця суміжності;

X – матриця признаків.

Для враховування контексту оточення кожної вершини отримуєтьсґ інформація з її суміжних вершин. Таким чином, сусіди першого рівня позначаються як:

$$N_i = \{j: (i, j)\ \epsilon\ E\ \vee (j, i)\ \epsilon\ E\}. \#(2.13)$$

Це дозволяє ввести матрицю ознак всіх сусідніх вершин:

$$X_{N_i} = \left\{\left\{x_j: j\ \epsilon\ N_i\right\}\right\}. \#(2.14)$$

Далі отримуємо функцію $f(X, A)$, еквіваріантну для перестановок, через функцію g по сусідніх вершинах:

$$f(X, A) = \begin{bmatrix} - g\left(x_1, X_{N_1}\right) - \\ - g\left(x_2, X_{N_2}\right) - \\ \vdots \\ - g\left(x_n, X_{N_n}\right) - \end{bmatrix}, \#(2.15)$$

де X_{N_i} – матриця ознак суміжних вершин.

Отже, для еквіваріантності функції f, g повинна володіти властивістю незалежності до перестановок вершин у X_{N_i}. Тому g – інваріантна.

Дана функція оновлює ознаки вершини, беручи до уваги її суміжні вершини. Це аналогічно роботі одного шару в будь-якій графовій нейронній

мережі. За допомогою функції f можна вирішувати різноманітні завдання на графах, такі як передбачення граней, класифікація вершин та графів тощо.

Існує три основних підходи для реалізації функції g:

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

Розглянемо ці підходи детальніше.

У графових згорткових нейронних мережах ознаки сусідів вершин агрегуються через константні ваги c_{ij}:

$$h_i = \phi\left(x_i, \sum_{i \epsilon N_i} c_{ij}\psi(x_j)\right), \#(2.16)$$

де ϕ – інваріантна функція;

ψ – еквіваріантна функція;

N_i – сусіди вершини i 1-ого рівня.

У типовій ситуації ці ваги вираховуються на основі загального графа.

Даний тип графових нейронних мереж має сенс застосовувати для гомофілійних графів, де справджується твердження, що схожі вершини мають високу ймовірність бути з'єднаними. Однією з ключових переваг цього підходу є висока ефективність обчислень, оскільки ваги обчислюються лише один раз і залишаються незмінними. Серед найпоширеніших архітектур виділяється GCN. Оновлення ознак кожної вершини здійснюється за формулою:

$$h_i = \Theta^T \sum_{j \epsilon N_i} \frac{e_{j,i}}{\sqrt{\widehat{d}_j \widehat{d}_\iota}} x_j, \#(2.17)$$

де Θ^T – функція, яка тренується;

$\widehat{d_i} = 1 + \sum_{j \epsilon N_i} e_{j,i}$ – зважений степінь вершини.

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.

Графові уважні нейронні мережі (GAT) агрегують ознаки сусідів вершин за допомогою неявно обчислених ваг $a_{ij} = a(x_i, x_j)$. Формула для оновлення ознак вершини виглядає так:

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

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.

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.

Нейронні мережі з передачею повідомлень (Message Passing Neural Networks, MPNN) формують ознаки сусідніх вершин за допомогою повідомлень $m_{ij} = \psi(x_i, x_j)$. Формула для оновлення ознак вершини виглядає так:

$$h_i = \phi\left(x_i, \sum_{i \epsilon N_i} \psi(x_i, x_j)\right), \#(2.19)$$

де ϕ – інваріантна функція;

ψ – еквіваріантна функція;

N_i – сусіди вершини i 1-ого рівня.

Тут задіюється уся інформація про цільову вершину та суміжну, на “льоту” обчислюючи потрібні коефіцієнти.

Такі графові нейронні мережі потребують більше часу для тренування через динамічне обчислення інформації для кожного ребра. Це зумовлено тим, що під час тренування GNN для кожного графового об’єкта виконується агрегування інформації від сусідів, що збільшує обчислювальну складність і час тренування. Проте, ця особливість є і їхньою перевагою, оскільки дозволяє GNN ефективно моделювати складні взаємозв'язки в графах, що робить їх надзвичайно корисними для задач у біології та хімії. Наприклад, вони використовуються для прогнозування взаємодії між молекулами, класифікації біологічних мереж, виявлення властивостей хімічних сполук:

$$h_i = \theta\left(x_i + \sum_{j \epsilon N_i} x_j\right), \#(2.20)$$

де θ – нейронна мережа, така як багатошаровий перцептрон;

N_i – сусіди вершини i 1-ого рівня.

У GINE ознаки вершини оновлюються за формулою:

$$h_i = \theta\left(x_i + \sum_{j \epsilon N_i} ReLU\left(x_j + e_{j,i}\right)\right), \#(2.21)$$

де θ – звичайна нейронна мережа;

N_i – сусіди вершини i 1-ого рівня;

$e_{j,i}$ – це ознаки ребер.

Отримуємо, що GINE є модифікованою версією GIN, яка враховує ознаки ребер додатково до ознак вершин. Це робить GINE більш адаптивним для задач, де ознаки ребер відіграють важливу роль.

2.2.8 Техніка машин опорних векторів (SVM)

Сфера застосування: класифікація, регресія, виявлення аномалій.

Інтуїтивне пояснення: знаходження гіперплощини, яка найкращим чином розділяє дані на два класи.

Принцип роботи: максимізація відстані (маржі) між гіперплощиною та найближчими точками обох класів (опорними векторами).

Переваги: висока точність, здатність до генералізації.

Недоліки: великі вимоги до обчислювальних ресурсів, важкість налаштування параметрів.

Чутливість до аномалій: середня.

Обмеження: не підходить для великих наборів даних.

Швидкодія: залежить від обраного ядра та розміру даних.

2.2.9 Підсилення градієнтного бустингу (XGBoost)

Сфера застосування: класифікація, регресія, ранжування, виявлення аномалій.

Інтуїтивне пояснення: покращена версія градієнтного бустингу, яка оптимізує швидкість і точність.

Принцип роботи: послідовне додавання дерев, кожне з яких виправляє помилки попередніх.

Переваги: висока швидкість, ефективність, гнучкість.

Недоліки: складність налаштування параметрів, ризик перенавчання.

Чутливість до аномалій: низька.

Обмеження: може бути неефективний для дуже малих наборів даних.

Швидкодія: висока.

Математична модель: ансамбль дерев, де кожне нове дерево намагається виправити помилки попередніх.

Цільова функція: мінімізація суми функції втрат та регуляризаційного штрафу за складність моделі.

2.3 Математична модель алгоритму баєсівської оптимізації гіперпараметрів

Баєсівська оптимізація є потужним методом для знаходження оптимальних гіперпараметрів моделі машинного навчання. Вона використовує принципи баєсівської статистики для моделювання невідомої функції втрат і виконує ефективний пошук у просторі гіперпараметрів. Цей метод є актуальним, коли змінні, які потрібно оптимізувати, є дискретними або категоріальними, а функція втрат (цільова функція) є недиференційованою відносно змінних.

Нехай $f(x)$ – невідома функція втрат, яку ми хочемо мінімізувати, де x – це вектор гіперпараметрів. Метою баєсівської оптимізації є знаходження такого x^*, що мінімізує значення $f(x)$.

Побудова сурогатної моделі: використовується гаусівський процес (GP) для моделювання $f(x)$. Гаусівський процес задається як::

$$f(x) \sim GP\big(m(x), k(x, x')\big), \#(2.22)$$

де $m(x)$ – це функція середнього значення;
$k(x, x')$ – коваріаційна функція (ядерна функція).

Оцінка виразності (Acquisition Function): використовується для визначення, де в просторі гіперпараметрів слід здійснити наступний крок. Популярні варіанти включають очікуване поліпшення (EI), верхню межу довіри (UCB) та ймовірність поліпшення (PI). Оптимізація функції виразності: вибір наступної точки для оцінки шляхом максимізації функції виразності.

Оптимізація виразності: вибір наступного набору гіперпараметрів для оцінки шляхом максимізації функції виразності.

Оновлення сурогатної моделі: після оцінки сурогатна модель оновлюється з новими даними, і процес повторюється.

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

Отже, баєсівська оптимізація дозволяє ефективно знаходити оптимальні гіперпараметри, використовуючи інформацію про попередні оцінки функції втрат. Вона здатна обробляти складні залежності та невизначеність у функції втрат, роблячи її ідеальною для задач оптимізації в машинному навчанні, де пряме вивчення функції втрат може бути важким або дорогим.

На практиці для цих цілей часто використовують Python фреймворк Optuna. Optuna - це фреймворк для автоматизованого тюнінгу гіперпараметрів, який підтримує баєсівську оптимізацію серед інших методів. Баєсівська оптимізація в Optuna використовує гаусівські процеси для моделювання функції втрат та визначення наступних кроків у просторі гіперпараметрів.

Процес баєсівської оптимізації в Optuna:

- ініціалізація – визначення користувачем простору гіперпараметрів і функції, яка повертає метрику продуктивності моделі, яку потрібно мінімізувати або максимізувати;
- вибір початкової множини гіперпараметрів (зазвичай випадковим чином);
- побудова сурогатної моделі – використання гаусівського процесу для апроксимації функції втрат на основі доступних даних;
- визначення виразності – використання функції виразності, такої як очікуване поліпшення (EI), для визначення найбільш перспективних точок у просторі гіперпараметрів;
- розрахунок EI для кожної точки – оптимізація виразності та вибір наступного набору гіперпараметрів;

- максимізація функції виразності для вибору наступної точки для оцінки;
- оцінка функції втрат: тренування моделі з новим набором гіперпараметрів та оцінка її метрик продуктивності;
- оновлення сурогатної моделі: додавання нових даних до сурогатної моделі та оновлення її параметрів;
- повторення кроків 3-6: процес продовжується до заданого критерію зупинки, наприклад, до досягнення максимальної кількості ітерацій або до стабілізації результатів.

2.4 Висновки

У цьому розділі було проаналізовано основні компоненти, які повинен містити AutoML пайплайн. Було наведено детальний аналіз та порівняння технологій, завдяки яким можна реалізувати кожен з компонентів. Також було описано, які існують способи представлення біологічних даних для їх використання у алгоритмах машинного навчання. Завершується цей розділ постановкою задачі і обгрунтованим вибором відповідних технологій.

3 ON-CLOUD СИСТЕМА АВТОМАТИЗИВАНОГО ТРЕНУВАННЯ ТА МОНІТОРИНГУ МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ

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

Реалізація пайплайнів автоматизованого машинного навчання (AutoML) вимагає включення різноманітних технологій, кожна з яких відіграє ключову роль у автоматизації різних аспектів процесу машинного навчання. Цілісна інтеграція цих технологій сприяє безшовному функціонуванню AutoML пайплайну, починаючи від отримання даних і закінчуючи деплойментом та моніторингом моделей. У цьому розділі надається огляд необхідних технологій та пояснюється їх взаємодія у екосистемі AutoML.

Загальна схема AutoML пайплайну наступна: спочатку системи управління даними полегшують зберігання та отримання даних, які потім обробляються за допомогою інструментів попередньої обробки даних. Після обробки використовуються інструменти інжинірингу ознак для вибору ознак. Потім фреймворки машинного навчання використовуються для створення та навчання моделей на підготовлених даних, а бібліотеки налаштування гіперпараметрів допомагають у оптимізації цих моделей. Після оптимізації моделі деплояться за допомогою платформ для деплойменту моделей, які забезпечують їх доступність для виконання інференсу. Інструменти моніторингу продуктивності постійно відстежують продуктивність моделі в продуктивному середовищі, тоді як бібліотеки візуалізації даних надають візуальні відомості про дані та метрики моделі.

3.1.1 База даних

Отож, почнемо дослідження і вибір технологій із баз даних. Системи управління даними є важливими для ефективного оброблення великих обсягів даних. Вони надають механізми для зберігання, отримання та управління даними,

забезпечуючи цілісність та доступність даних. Можливими варіантами є PostgreSQL, MongoDB та Hadoop. Розглянемо детальніше кожен із них.

Найкращі випадки використання PostgreSQL: корпоративні застосунки, сховища даних, фінансові застосунки та будь-які сценарії, які вимагають дотримання ACID. Переваги: дотримання ACID, надійність, підтримка розширеного індексування, повнотекстовий пошук та сильна спільнотна підтримка. Недоліки: може бути занадто складним для простих застосунків, вимагає експертних знань для оптимізації.

Найкращі випадки використання MongoDB: системи, орієнтовані на документи, аналітика в реальному часі, застосунки з великим обсягом CRUD-операцій. Переваги: відсутність схеми, висока масштабованість, гарне рішення для неструктурованих даних. Недоліки: відсутність дотримання ACID, не підходить для складних транзакцій.

Найкращі випадки використання Hadoop: аналітика великих даних, розподілені обчислення, архітектури даних. Переваги: масштабованість, гарне рішення для обробки великих наборів даних, підтримка розподілених обчислень. Недоліки: складність в налаштуванні та обслуговуванні, не підходить для обробки в реальному часі.

У таблиці 3.1 наведено основні системи управління даними.

Таблиця 3.1 – Основні системи управління даними

Критерій	PostgreSQL	MongoDB	Hadoop
Тип бази	Реляційний	NoSQL	Розподілений
Підтримка ACID	Так	Ні	Ні
Масштабованість	Помірна	Висока	Висока
Схема	Фіксована	Гнучка	Гнучка

Враховуючи специфіку пайплайну, було вибрано PostgreSQL, як найоптимальнішу базу даних для наших задач.

3.1.2 Інструмент попередньої обробки даних

Наступним етапом є вибір інструментів попередньої обробки даних, які автоматизують процеси очищення даних та їх трансформації, підготовлюють дані для подальшого аналізу. Кандидатами для цієї задачі є Pandas, Scikit-learn та DataRobot.

Найкращі випадки використання Pandas: очищення даних, трансформація та аналіз у невеликих та середніх наборах даних. Переваги: простота використання, широкий спектр функцій для маніпулювання даними. Недоліки: не підходить для великих наборів даних.

Найкращі випадки використання Scikit-learn: попередня обробка для машинного навчання, вибір ознак та трансформація даних. Переваги: широкий діапазон технік попередньої обробки, легка інтеграція з іншими бібліотеками ML. Недоліки: не оптимізовано для великих даних.

Найкращі випадки використання DataRobot: автоматизована попередня обробка в пайплайнах машинного навчання. Переваги: автоматизована попередня обробка, гарне рішення для недосвідчених користувачів. Недоліки: обмежена можливість налаштування, комерційний софт.

У таблиці 3.2 наведено основні інструменти попередньої обробки даних.

Таблиця 3.2 – Основні інструменти попередньої обробки даних

Критерій	Pandas	Scikit-learn	DataRobot
Легкість використання	Висока	Висока	Помірна
Кастомізованість	Висока	Висока	Низька

Кінець таблиці 3.2 – Основні інструменти попередньої обробки даних

Масштабованість	Помірна	Помірна	Висока
Функціонал для препроцесингу	Розширений	Розширений	Розширений

Таким чином, для первинної обробки даних найкраще підійдуть інструменти Pandas.

3.1.3 Інструмент відбору ознакх

Наступний крок - визначення інструментів інженерії ознак, які автоматизують процес отримання нових ознак та вибору найінформативніших з них. Кандидатами для вибору є Scikit-learn, Featuretools та H2O.ai. Розглянемо їх детальніше.

Найкращі випадки використання Scikit-learn - це задачі вибору найінформативніших фіч з використанням алгоритмів класичного машинного навчання. Переваги: можливість самостійно написати повністю пайплайн вибору найкращих ознак. Недоліки: відсутній функціонал для отримання нових фіч шляхом комбінації існуючих.

Найкращі випадки використання Featuretools: автоматизована інженерія ознак для реляційних наборів даних. Переваги: автоматизоване генерування ознак, гарно обробляє реляційні дані. Недоліки: складність навчання для налаштування, може генерувати неактуальні ознаки.

Найкращі випадки використання H2O.ai: масштабна інженерія ознак та завдання з машинного навчання. Переваги: масштабованість, підтримка автоматизованої інженерії ознак. Недоліки: комерційний софт, може вимагати певного досвіду з машинним навчанням.

У таблиці 3.3 наведено основні інструменти інженерії ознак.

Таблиця 3.3 – Основні інструменти інженерії ознак

Критерій	Scikit-learn	Featuretools	H2O.ai
Легкість використання	Висока	Помірна	Помірна
Кастомізованість	Висока	Висока	Помірна
Масштабованість	Середня	Помірна	Висока
Автоматична генерація ознак	Ні	Так	Так

Отже, для нашої задачі найкраще підходить Scikit-learn, оскільки нам не потрібно генерувати нові фічі з існуючих (втрачається їх інтерпретабельність, оскільки кожна фіча – це біт фінгерпринту, який відповідає за присутність певної сабструктури у молекулі).

3.1.4 Фреймворк для машинного навчання

Далі необхідно визначитись із фреймворком машинного навчання для створення, навчання та оцінки моделей. Основними кандидатами є TensorFlow, PyTorch та Scikit-learn.

Найкращі випадки використання TensorFlow: застосунки глибокого навчання, масштабні завдання ML, застосунки ML, готові до виробництва. Переваги: масштабованість, широка підтримка бібліотек, гарна спільнотна підтримка. Недоліки: крутіший шлях навчання порівняно з іншими фреймворками.

Найкращі випадки використання PyTorch: дослідження та розробка, швидке створення прототипів. Переваги: динамічний обчислювальний граф, інтуїтивний та зручний для користувача. Недоліки: менш зрілі опції розгортання порівняно з TensorFlow.

Найкращі випадки використання Scikit-learn: класичні завдання з машинного навчання, невеликі та середні набори даних. Переваги: простота використання, широкий спектр алгоритмів. Недоліки: не підходить для глибокого навчання, не оптимізовано для великих наборів даних.

У таблиці 3.4 наведено основні фреймворки машинного навчання.

Таблиця 3.4 – Основні фреймворки машинного навчання

Критерій	TensorFlow	PyTorch	Scikit-learn
Легкість використання	Помірна	Висока	Висока
Кастомізованість	Висока	Висока	Помірна
Підтримка комьюніті	Помірна	Висока	Висока
Зоопарк моделей	Розширений	Розширений	Обмежений

Таким чином, найкращим рішенням для нас буде використання Scikit-learn для алгоритмів класичного машинного навчання та PyTorch для нейронних мереж.

3.1.5 Інструмент оптимізації гіперпараметрів

Наступним етапом є вибір бібліотеки тюнінгу гіперпараметрів, які автоматизують пошук найоптимальніших параметрів моделі. Серед варіантів, які варті детальнішого розгляду, є Optuna, Hyperopt і Scikit-Optimize.

Optuna - це бібліотека загального призначення оптимізація гіперпараметрів. Переваги: підтримка різних стратегій оптимізації (зокрема Баєсовської оптимізації), можливість написання власної обгортки для тюнінгу будь-якої моделі, підтримка розпаралеленої оптимізації шляхом синхронізації перебору

простору гіперпараметрів через спільну базу даних. Недоліки: може бути повільною для високовимірних просторів пошуку.

Найкращі випадки використання Hyperopt: Баєсівська оптимізація для налаштування гіперпараметрів. Переваги: підтримує Баєсівську оптимізацію, паралельну оптимізацію. Недоліки: API може бути менш інтуїтивним порівняно з іншими бібліотеками.

Найкращі випадки використання Scikit-Optimize: налаштування гіперпараметрів для моделей Scikit-learn. Переваги: інтеграція з Scikit-learn, дружній для користувача API. Недоліки: обмежено моделями Scikit-learn, менше функцій порівняно з Optuna та Hyperopt.

У таблиці 3.5 наведено основні бібліотеки тюнінгу гіперпараметрів.

Таблиця 3.5 – Основні бібліотеки тюнінгу гіперпараметрів

Критерій	Optuna	Hyperopt	Scikit-Optimize
Легкість використання	Висока	Помірна	Висока
Масштабованість	Помірна	Помірна	Помірна
Технології оптимізації	Кілька	Кілька	Кілька
Паралелізація	Так	Так	Так

Таким чином, найкращим вибором є Optuna, оскільки дозволяє ефективно використати Баєсовську оптимізацію у розпаралеленому варіанті.

3.1.6 Сервіси хмарних провайдерів для побудови ML пайплайнів

Кожна з основних платформ хмарних обчислень, як то AWS, Google Cloud та Azure, мають спеціалізовані сервіси для побудови ML пайплайнів і деплойменту моделей.

Від AWS таким сервісом є SageMaker, призначений для Enterprise-level хмарних рішень, large-scale застосунків для машинного навчання.

Від Azure таким аналогом є Azure ML, створений для розгортання машинного навчання на продакшн рівні та гібридних хмарних конфігурацій.

Натомість Google пропонує Google Cloud AI Platform для end-to-end розгортання пайплайну машинного навчання.

У таблиці 3.6 наведено основні платформи хмарних обчислень та деплойменту моделей.

Таблиця 3.6 – Основні платформи хмарних обчислень та деплойменту моделей

Критерій	AWS SageMaker	Azure ML	Google Cloud AI Platform
Легкість використання	Помірна	Помірна	Помірна
Масштабованість	Висока	Висока	Висока
Кастомізованість	Висока	Висока	Висока
Built-in моделі	Так	Так	Так

Перевагами кожного з цих сервісів є спрощений процес деплойменту та налаштування середовища, а також дуже тісна інтеграція з іншими сервісами цих же хмарних провайдерів. Проте це ж є і найголовнішим недоліком, оскільки розроблений пайплайн в межах однієї з платформ прив’язує користувача до неї і не дозволяє швидко перенести його на іншу платформу за потреби. Окрім того, реалізація пайплайнів виходить досить дорогою. Окрім того, реалізація пайплайнів виходить досить дорогою.

Таким чином, було прийнято рішення утриматися від використання спеціалізованих сервісів хмарних провайдерів для побудови пайплайнів машинного навчання.

3.1.7 Інструмент трекінгу та візуалізації експериментів

У сфері машинного навчання трекінг та візуалізація експериментів є ключовими для ефективного управління процесами розробки та аналізу. MLFlow, Weights & Biases та Aim - це передові платформи, які надають інструменти для цих завдань. Розглянемо їх характеристики, переваги та недоліки.

MLFlow – це відкрите програмне забезпечення від Databricks, що забезпечує інтеграцію з багатьма популярними бібліотеками машинного навчання.

Основні особливості MLFlow:

- модульність – MLFlow має чотири основні компоненти: MLflow Tracking, MLflow Projects, MLflow Models, MLflow Registry;
- підтримка мов – підтримує Python, R, Java та REST API;
- інтеграція з іншими інструментами – легко інтегрується з Jupyter Notebook, Apache Spark тощо.

Переваги MLFlow:

- широка інтеграція з іншими інструментами;
- гнучкість у виборі середовища зберігання даних (локально, сервер, AWS, Azure);
- відкритий код.

Недоліки MLFlow:

- відносно складний у налаштуванні та використанні для новачків;
- обмежена функціональність візуалізації порівняно з іншими платформами.

Weights & Biases (W&B) – це комерційний інструмент, який надає детальний трекінг експериментів та потужні інструменти візуалізації.

Основні особливості Weights & Biases:

- детальний трекінг – автоматичне логування гіперпараметрів, метрик, вихідних даних моделі тощо;
- візуалізація – потужні інструменти для візуалізації, включаючи порівняння метрик між різними запусками;
- інтеграція – легко інтегрується з багатьма популярними бібліотеками.

Переваги Weights & Biases:

- інтуїтивно зрозумілий інтерфейс;
- широкі можливості для візуалізації;
- підтримка командної роботи та спільного доступу.

Недоліки Weights & Biases:

- вартість використання для комерційних проєктів;
- може бути перевантаженням для простих проєктів.

Aim - це інструмент відкритого коду, зосереджений на трекінгу та візуалізації експериментів.

Основні особливості Aim:

- фокус на візуалізації – забезпечує детальну візуалізацію метрик та гіперпараметрів;
- гнучка налаштування – можливість налаштування трекінгу та відображення даних;
- сумісність – легко інтегрується з багатьма фреймворками машинного навчання.

Переваги Aim:

- висока кастомізація візуалізацій;
- відкритий код та безкоштовне використання;
- легкість у встановленні та налаштуванні.

Недоліки Aim:

- може вимагати більше часу на налаштування для специфічних потреб;
- обмежені можливості для спільного використання та командної роботи порівняно з W&B.

У таблиці 3.7 наведено основні платформи для трекінгу та візуалізації експериментів.

Для цілей нашого проекту вирішено було використати продукт з відкритим кодом, тому W&B було відхилено. Між MlFlow та Aim, було вирішено надати перевагу MlFlow, через потенційні можливості використання його додаткових сервісів з екосистеми для машинного навчання. Проте, після впровадження, було також протестовано Aim і зрештою W&B. Тож по кінцевих результатах, найкращим рішенням було б використання W&B, а MlFlow - найгіршим (додаткові сервіси не знадобилися, а функціонал для аналізу є занадто обмеженим і негнучким).

Таблиця 3.7 – Основні платформи для трекінгу та візуалізації експериментів

Критерій	Aim	MLFlow	W&B
Open-source	Так	Так	Ні
Масштабованість	Середня	Низька	Висока
Кастомізованість	Середня	Низька	Висока
Тип бази даних	Key-value	SQL	Key-value
Швидкість роботи при великій кількості проектів	Помірна	Повільна	Швидка

3.2 Дослідження і вибір технології On-Cloud оркестрації комонентів системи автоматизованого тренування та моніторингу моделей машинного навчання

У даному підрозділі здійснюється детальне порівняння двох популярних технологій для роботи з Kubernetes кластерами у контексті побудови машинно-навчальних пайплайнів: KubeFlow і Argo.

Ці системи є ключовими інструментами для ефективної реалізації пайплайнів у хмарному середовищі, кожна з яких має свої особливості та переваги.

3.2.1 KubeFlow

KubeFlow — це відкрита платформа, спроєктована для оркестрації машинно-навчальних пайплайнів на Kubernetes. Головна мета KubeFlow полягає у спрощенні процесу розгортання, масштабування та управління машинно-навчальними моделями, роблячи цей процес більш доступним і менш часозатратним для розробників і науковців.

KubeFlow включає набір компонентів, які покривають різні аспекти машинного навчання:

- JupyterHub для управління Jupyter Notebooks;
- Pipelines для створення та управління машинно-навчальними пайплайнами;
- Katib для оптимізації гіперпараметрів;
- Serving для служб розгортання моделей за допомогою TensorFlow Serving, Seldon Core або інших інструментів.

Переваги:

- інтеграція з широким спектром інструментів для машинного навчання;
- глибока інтеграція з екосистемою Kubernetes;
- підтримка багатокористувацьких середовищ.

Недоліки:

- налаштування та управління може викликати складність;
- велика кількість компонентів, що може ускладнити використання для нових користувачів, а також спеціалізація на задачах машинного навчання, що робить технологію менш зручною для більш загальних задач.

3.2.2 Argo

Argo — це система для оркестрації контейнерів, яка підтримує складні паралельні та послідовні робочі процеси в Kubernetes. Argo Workflows використовується для автоматизації, зокрема, в ML пайплайнах, де кожен етап може бути виконаний як окремий контейнер. Argo складається з кількох компонентів:

- Argo Workflows для управління робочими процесами;
- Argo CD для безперервної розробки та розгортання;
- Argo Events для зв'язку на основі подій;
- Argo Rollouts для керування прогресивними розгортаннями.

Переваги:

- висока гнучкість у оркестрації завдань;
- підтримка паралельного та умовного виконання;
- простота інтеграції з іншими сервісами Kubernetes.

Недоліки:

- більш обмежені можливості для задач ML, порівняно з KubeFlow;
- вимагає додаткової інтеграції з інструментами для повноцінної підтримки всього життєвого циклу ML.

У таблиці 3.8 наведено порівняння Kubeflow та Argo. Це порівняння демонструє, що обидві технології мають свої переваги та області застосування. Але оскільки KubeFlow “під капотом” використовує функціонал Argo, а також спеціалізується на задач машинного навчання, то було прийнято рішення обрати саме його для реалізації On-Cloud деплойменту пайплайну.

Таблиця 3.8 – Порівняння Kubeflow та Argo

Параметр	KubeFlow	Argo
Сфера застосування	Спеціалізовано для ML	Загальна оркестрація робочих процесів

Кінець таблиці 3.8 – Порівняння Kubeflow та Argo

Комплексність	Висока	Середня
Гнучкість	Висока в контексті ML	Висока в контексті загальної оркестрації
Функціональність	Висока для ML пайплайнів	Середня для ML пайплайнів
Інтеграція	Глибока інтеграція з ML інструментами	Проста інтеграція з Kubernetes

3.3 AutoML пайплайн системи автоматизованого тренування та моніторингу моделей машинного навчання у середовищі Kubernetes з оркестрацією KubeFlow

У цьому розділі розглядається реалізація AutoML пайплайну, а також детально описується налаштування взаємозв’язків компонентів Kubernetes, KubeFlow та AutoML пайплайну для їх цілісної роботи.

Загальна схема системи втоматизованого тренування та моніторингу моделей машинного навчання наведена на рис. 3.1.

Розбір цієї схеми почнемо з основних компонентів Kubernetes, що забезпечують його функціонування відповідно до високих стандартів надійності та масштабування, та охоплюють елементи, зображені на рисунку 3.1.

kube-apiserver – це компонент узла управління, який виступає як основний інтерфейс для кластера Kubernetes. Він приймає REST запити, перевіряє їх, виконує бізнес-логіку, і потім, за потреби, оновлює стан об'єктів у etcd.

etcd – це високонадійна база даних, що зберігає ключові значення, яка використовується як репозиторій для усіх кластерних даних. Вона забезпечує консистентне зберігання даних і управління спільними конфігураціями для кластера.

kube-scheduler – цей компонент спостерігає за нещодавно створеними подами без призначеного вузла і вибирає вузол, на якому вони повинні працювати. Планувальник враховує доступні ресурси, вимоги подів, політики безпеки, специфікації, афінітетність, розміщення подів тощо.

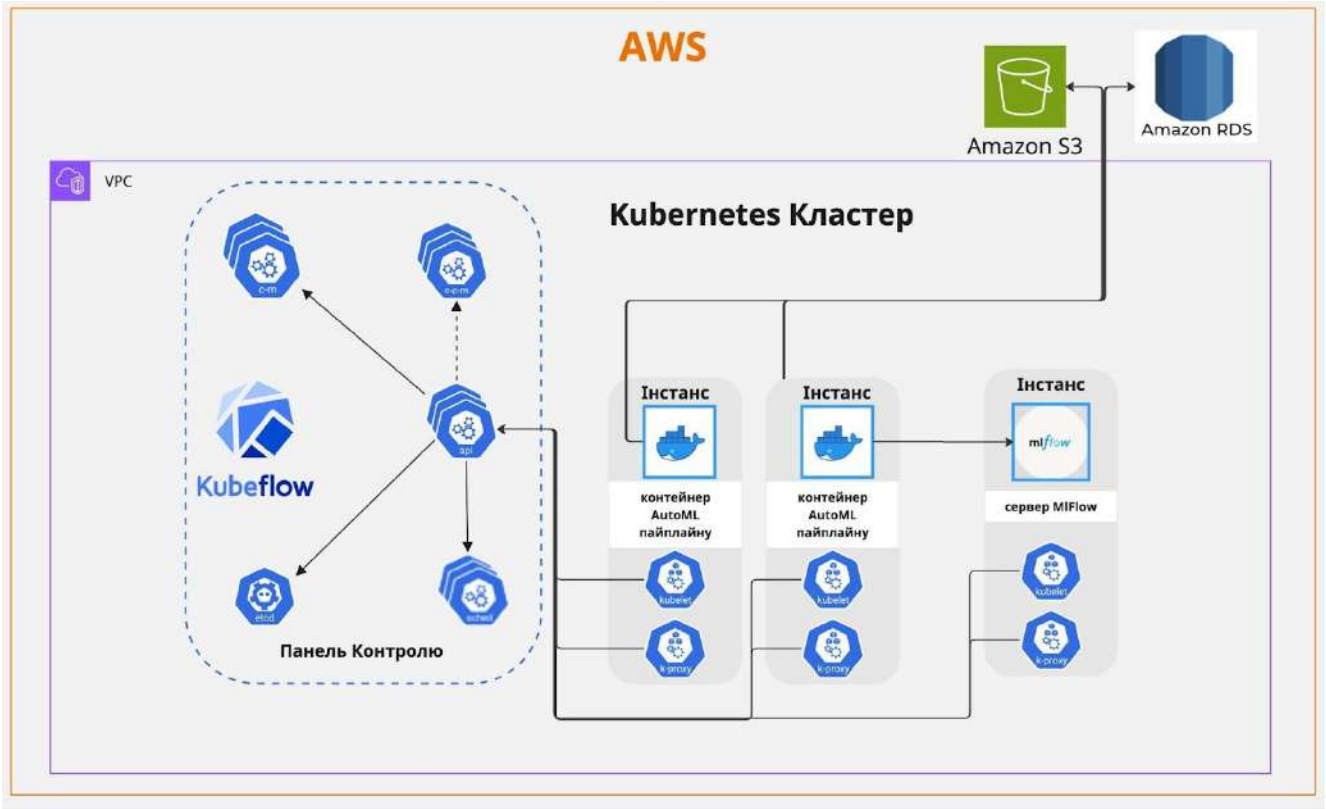


Рисунок 3.1 – Основні компоненти Kubernetes

kube-controller-manager — цей компонент включає в себе набір контролерів. Контролери Kubernetes — це фонові процеси, які стежать за станом кластера через apiserver і вживають необхідних заходів, щоб перейти до бажаного стану. Наприклад, контролер вузлів відповідає за реагування, коли вузли стають недоступними.

Kubelet — це основний компонент на кожному вузлі, який гарантує, що контейнери запущені в подах працюють належним чином. Він керує PodSpecs (специфікаціями подів), які визначають, як под повинен працювати.

kube-proxy — підтримує мережеві правила на вузлах. Ці мережеві правила дозволяють мережевому спілкуванню до подів з мережевих сесій всередині або поза кластером.

Container runtime — це програмне забезпечення, відповідальне за запуск контейнерів. Kubernetes підтримує кілька рантаймів: Docker, containerd, CRI-O тощо.

Web UI (Dashboard) — це веб-інтерфейс користувача для Kubernetes. Він дозволяє користувачам створювати та керувати ресурсами кластера та переглядати інформацію про помилки.

KubeFlow Pipelines дозволяє користувачам визначати, оркеструвати та реалізовувати машинно-навчальні пайплайни. На рисунку 3.2 зображено екосистему KubeFlow.

Специфікація пайплайну: розробники використовують Python SDK для створення специфікацій пайплайну. Вони визначають різні завдання (стадії пайплайну), що включають підготовку даних, тренування моделі, її оцінку та розгортання, їх взаємодію, послідовність, вхідні параметри.

Компоненти: кожне завдання описується як компонент, який може бути контейнером, що виконує певну функцію. Компоненти специфікуються через контейнерні зображення, вхідні та вихідні дані.

Компіляція пайплайну: з використанням KubeFlow SDK, специфікація пайплайну компілюється в формат YAML, який використовується Kubernetes для розгортання пайплайну.

Розгортання пайплайну: KubeFlow Pipelines використовує Argo Workflows для оркестрації робочих процесів. Кожне завдання в пайплайні стає етапом в Argo Workflow.

Запуск пайплайну: користувачі можуть запускати пайплайни через KubeFlow UI або програмно через KubeFlow Pipelines API.

Виконання завдань: завдання виконуються в ізольованих контейнерах на worker nodes в Kubernetes. Kubelet на кожному вузлі керує життєвим циклом цих контейнерів.

Моніторинг стану: kube-scheduler розподіляє контейнери по вузлах на основі ресурсних вимог, а kube-proxy забезпечує мережеве сполучення між контейнерами.

Autoscaling: KubeFlow Pipelines можуть використовувати можливості Kubernetes HPA (Horizontal Pod Autoscaler) для автоматичного масштабування завдань пайплайну в залежності від поточного навантаження.

Збір результатів і логів: Вивід кожного контейнера зберігається і доступний через Kubernetes logs для аналізу результатів або виявлення помилок.

Архітектуру AutoML пайплайну зображено на рисунку 3.2., а кодову реалізацію наведено в Додатку А.

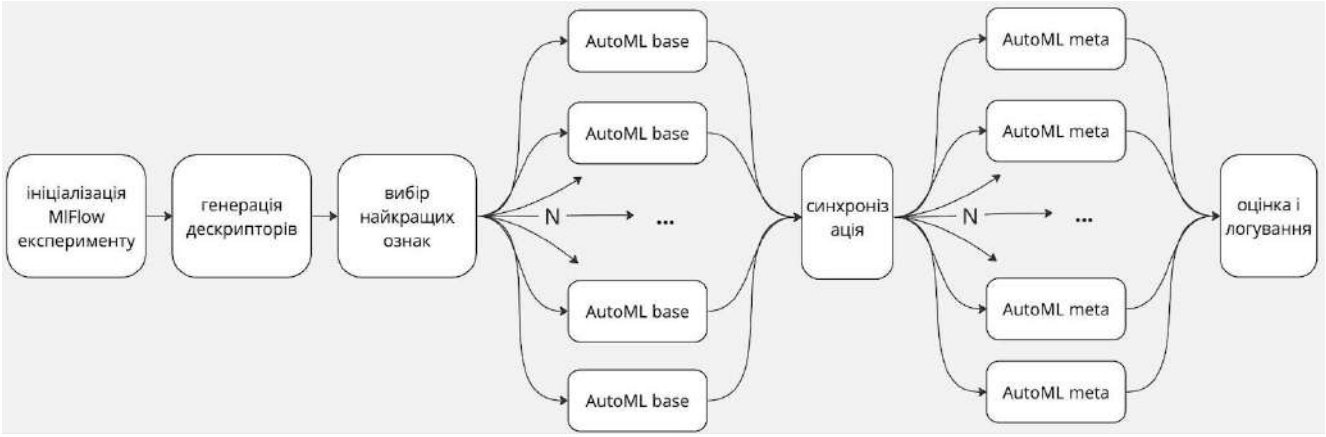


Рисунок 3.2 – Архітектура AutoML пайплайну

Компоненти пайплайну взаємодіють один з одним через передачу даних і параметрів, створюючи гнучкий і масштабований робочий процес.

Розглянемо детальніше функцію “init_mlflow_experiment_op”. Опис: цей крок встановлює початкові налаштування для експерименту в MLflow, забезпечуючи трекінг і збереження результатів тренування та валідації моделі. Параметри: включає основні налаштування оточення (environment variables), такі як адреси серверів, конфігураційні параметри доступу до ресурсів тощо. Виконання: використовує образ Docker для виконання задачі, з конфігурацією, що визначається динамічно через KubeFlow.

Розглянемо детальніше генерацію дескрипторів. Функція: “generate_descriptors_op”. Опис: автоматизовано генерує дескриптори (ознаки) з вхідних даних, які будуть використані для тренування моделі. Це може включати попередню обробку та нормалізацію даних. Параметри: входи забезпечуються через середовище, з управлінням версіями через KubeFlow і контейнери Docker.

Вибір найкращих дескрипторів. Функція: “select_descriptors_op”. Опис: вибірка найефективніших описів з сгенерованого набору, щоб оптимізувати продуктивність майбутньої моделі. Виконання: автоматизований вибір виконується на основі статистичного аналізу і важливості ознак для прогнозування.

Розглянемо детальніше функцію “automl_op”, яка реалізовує логіку AutoML. Опис: проведення тренувань різних моделей машинного навчання, використовуючи вибрані описи та налаштування гіперпараметрів для знаходження оптимальної моделі. Параметри: включає рівень (LEVEL), який дозволяє налаштувати тренування моделей різного рівня складності (зі стекінгом чи без).

Логування результатів у MLflow. Функція: “log_automl_to_mlflow_op”. Опис: збереження результатів кожного тренування моделі в MLflow, що включає метрики продуктивності, параметри моделі та інше. Виконання: дані передаються через середовище виконання та зберігаються за допомогою спеціалізованого контейнера.

Налаштування AutoML пайплайну відбувається шляхом специфікації параметрів, наведених у таблиці 3.9.

Таблиця 3.9 – Параметри налаштування AutoML пайплайну

Параметр	Опис
USERNAME	Ім'я користувача для авторизації у сервісах.
PASSWORD	Пароль для авторизації у сервісах.

Продовження таблиці 3.9 – Параметри налаштування AutoML пайплайну

DATASETS_BUCKET	Назва bucket
EXPERIMENT_NAME	Назва експерименту у MLflow.
RUN_NAME	Назва запуску експерименту.
DATASET_FOLDER	Шлях до папки з набором даних.
DATASET_FILE	Ім'я файлу набору даних.
DESCRIPTORS_FILE	Ім'я файлу з описами.
SELECTED_DESCRIPTORS_FILE	Ім'я файлу з вибраними описами.
TARGET_FILE	Ім'я файлу цільових змінних.
PROBLEM_TYPE	Тип задачі машинного навчання
OPTIMIZATION_METRIC	Метрика для оптимізації моделі.
SCORE_THRESHOLD	Поріг для оцінки моделі.
TEST_SIZE	Розмір тестової вибірки.
DEVICE	Пристрій для виконання тренування моделі
BASE_TIMEOUT	Час вичерпання для одного етапу.
BASE_MODEL_FAMILIES	Сім'ї моделей для використання.
STRATIFY	Параметр стратифікації для розбиття даних.
CV_SPLITTER_INSTANCE	Інстанція для розділення даних.
BASE_IMPOSED_TUNING_PARAMS	Налаштування параметрів налаштування.
BASE_IMPOSED_DEFAULT_PARAMS	Стандартні параметри моделі.

Кінець таблиці 3.9 – Параметри налаштування AutoML пайплайну

BASE_TUNING_STRATEGY	Стратегія налаштування моделі.
BASE_MODEL_SELECTION_CRITERION	Критерій вибору моделі.
BEST_MODEL_SCORER	Функція для оцінки кращої моделі.
METRICS_TO_EVALUATE	Метрики для оцінки моделі.

3.4 Вибір хмарного провайдера для деплойменту

При виборі між AWS (Amazon Web Services) та GCP (Google Cloud Platform) для розгортання Kubernetes кластеру з KubeFlow, важливо враховувати кілька аспектів, що стосуються здатностей обох хмарних платформ.

Для початку розглянемо архітектуру та інтеграцію. AWS: Amazon EKS (Elastic Kubernetes Service) є високо інтегрованим рішенням, яке дозволяє легко запускати Kubernetes на AWS, використовуючи управління контрольними площинами Kubernetes, не вимагаючи від користувачів їх встановлення чи обслуговування. AWS надає автоматичне масштабування та налаштованість під конкретні вимоги, і добре інтегрується з іншими сервісами AWS. GCP: Google Kubernetes Engine (GKE) пропонує управління контейнеризованими додатками за допомогою Kubernetes, забезпечуючи автоматизацію багатьох задач, включаючи моніторинг та управління здоров'ям кластера. GKE також забезпечує глибоку інтеграцію з іншими сервісами Google, включаючи BigQuery та Cloud Storage.

Наступним аспектом розглянемо ціноутворення та знижки. AWS: Модель ціноутворення AWS включає оплату за вимогою, зарезервовані екземпляри, спотові інстанси та спеціальні хости. AWS також пропонує плани збереження, які дозволяють отримувати знижки на використання комп'ютингу. GCP: Google Cloud пропонує автоматичні знижки за тривале використання та значні знижки за зобов'язання використовувати певні ресурси на тривалий термін. Такий підхід може заохочувати економію коштів на тривалих проектах.

Продуктивність мережі та доступність. AWS: має широку глобальну інфраструктуру з дата-центрами у багатьох регіонах, що забезпечує низьку затримку та високу пропускну спроможність, особливо важливу для глобально розподілених додатків. GCP: має власну високопродуктивну приватну оптоволоконну мережу, що забезпечує швидке та ефективне з'єднання між регіонами.

Підтримка та документація. AWS: вважається більш складним у використанні, пропонує широкий спектр документації та ресурсів для навчання. GCP: більш очевидна взаємодія з сервісами, також має високу якість документації та підтримки, зокрема з використанням Google Cloud's operations suite для моніторингу та управління.

Отже, вибір між AWS (Amazon Web Services) та GCP (Google Cloud Platform) для розгортання Kubernetes кластера з KubeFlow в більшості випадків є рівноцінним. Тому, враховуючи те, що в компанії вже попередньо було використано сервіси AWS, було прийнято рішення зупинитися на цьому хмарному провайдері.

3.5 Вибір інстансів на AWS для CPU-bound задач і оцінка їх фінансової складової

Розглянемо три оптимальні варіанти EC2 інстансів для CPU-інтенсивних задач, таких як тренування класичних моделей машинного навчання, на платформі AWS:

- c5.9xlarge – 36 vCPU, 72 Gb RAM, годинна вартість близько $1.53 (Linux на вимогу у регіоні США, Схід), висока продуктивність CPU, оптимальний для обчислювальних задач, таких як обробка даних і машинне навчання;
- m5.24xlarge – 96 vCPU, 384 Gb RAM, годинна вартість близько $4.61 (Linux на вимогу у регіоні США, Схід), використання загального призначення, ідеально підходить для задач, які вимагають великої кількості пам'яті та високої продуктивності CPU;

− r5.12xlarge – 48 vCPU, 384 Gb RAM, годинна вартість близько $3.06 (Linux на вимогу у регіоні США, Схід), оптимізовано для задач, які потребують більше пам'яті, ідеально підходить для баз даних, аналітики великих даних.

У таблиці 3.10 наведено порівняння оптимальних варіантів EC2 інстансів для CPU-інтенсивних задач. Ці інстанси вибрані на основі їх здатності забезпечувати високу продуктивність обчислень та оптимальність вартості для тренування моделей машинного навчання. Вибір конкретного типу інстансу залежить від специфіки задачі та обсягів даних, необхідних для її виконання.

Таблиця 3.10 – Порівняння оптимальних варіантів EC2 інстансів для CPU-інтенсивних задач

Тип інстансу	vCPU	RAM (GiB)	Годинна вартість (USD)
c5.9xlarge	36	72	1.53
m5.24xlarge	96	384	4.61
r5.12xlarge	48	384	3.06

3.6 Вибір інстансів на AWS для GPU-bound задач і оцінка їх фінансової складової

Для тренування нейронних мереж на AWS, оптимальні варіанти інстансів з GPU включають P3, P4, та G4. Кожен з цих типів інстансів має свої особливості, які роблять їх підходящими для різних сценаріїв використання.

Amazon EC2 P3 Instances мають до 8 NVIDIA Tesla V100 GPUs, які підходять для вимогливих завдань глибокого навчання, таких як тренування складних моделей з великою кількістю даних. P3 є ідеальним вибором для задач, які вимагають значної обчислювальної потужності.

Amazon EC2 P4 Instances обладнані до 8 NVIDIA Tesla A100 GPUs, P4 інстанси забезпечують ще більшу потужність для тренування моделей, що

вимагають інтенсивних обчислень, і є відмінним варіантом для найновіших досліджень в галузі штучного інтелекту.

Amazon EC2 G4 Instances мають до 4 NVIDIA T4 GPUs і є більш бюджетними порівняно з P3 та P4. G4 підходять для більш широкого спектру машинного навчання та графічних завдань, де потрібна менша потужність GPU.

Кожен з цих типів інстансів має різну вартість та конфігурацію, що дозволяє адаптувати обчислювальні ресурси до конкретних потреб вашого проєкту. У таблиці 3.11 наведено порівняння оптимальних варіантів EC2 інстансів для GPU-інтенсивних задач.

Таблиця 3.11 – Порівняння оптимальних варіантів EC2 інстансів для GPU-інтенсивних задач

Instance Type	GPUs	Use Case	Pros	Cons	Pricing
P3	Up to 8 NVIDIA V100	Advanced deep learning	Highest compute power	Higher cost	Premium
P4	Up to 8 NVIDIA A100	Cutting-edge AI research	Highest performance	Very high cost	Most expensive
G4	Up to 4 NVIDIA T4	General ML and graphics	Cost-effective	Lower performance	More affordable

Ці інстанси забезпечують потужність та гнучкість для різноманітних задач машинного навчання, від загального моделювання до глибокого навчання та AI досліджень. Вибір відповідного типу інстанса залежатиме від специфіки завдань та вимог до обчислювальної потужності.

3.7 Висновки

У цьому розділі було проаналізовано основні компоненти, які повинен містити AutoML пайплайн. Було наведено детальний аналіз та порівняння технологій, завдяки яким можна реалізувати кожен з компонентів. Також було описано, які існують способи представлення біологічних даних для їх використання у алгоритмах машинного навчання. Завершується цей розділ постановкою задачі і обгрунтованим вибором відповідних технологій.

4 СТВОРЕННЯ СКОРИНГОВИХ ФУНКЦІЙ ДЛЯ ЕТАПУ ВИБОРУ НАЙКРАЩОЇ МОДЕЛІ

4.1 Опис проблеми

Автоматизація тренування моделей машинного навчання (ML) знаменує собою принципову зміну у сфері in silico відкриття ліків. Основною метою цього процесу є підвищення точності прогнозування та ефективності, що досягається за допомогою складних алгоритмів, які можуть автономно налаштовувати гіперпараметри та вибирати оптимальну модель. Ключовим елементом автоматичного вибору є його здатність адекватно оцінювати та знижувати ризик перенавчання, гарантуючи придатність моделей до нових даних. У цьому розділі аналізується ефективність використання показників крос-валідації (CV) для вибору моделей та подальшої оцінки їх узагальнюваності за допомогою окремих тестових наборів даних. Досліджується розбіжності між метриками CV та тестовими метриками як індикатори перенавчання та вивчаються продуктивність та стабільність різних сімейств ML моделей, застосованих до молекулярних наборів даних.

Розроблена автоматизована система вибору моделей працює за двома основними напрямками: налаштування гіперпараметрів і вибір моделі на основі метрик CV. Налаштування гіперпараметрів — це ітеративний процес, який має на меті знайти оптимальну конфігурацію для даної моделі, покращуючи її продуктивність на тренувальних даних. Вибір найкращої моделі, втім, ґрунтується на її продуктивності за декількома ітераціями CV, метою яких є гарантування, що ефективність моделі не є простим наслідком перенавчання на тренувальних даних.

Для оцінки узагальнюючої здатності моделі проводиться незалежна оцінка на зарезервованій тестовій підмножині. Розбіжність між метриками CV та тестовими метриками є мірою перенавчання; більша різниця свідчить про те, що модель, хоч і показує хороші результати на тренувальних та CV даних, може не бути ефективною на нових даних. Цей аспект підкреслює необхідність створення

надійного алгоритму вибору, який може розпізнати та усунути тенденцію до перенавчання.

Проведене дослідження ефективності автоматизованого методу вибору моделей включає порівняльний аналіз різних сімейств ML моделей: випадковий ліс, Extra Trees, CatBoost, XGBoost, лінійна та логістична регресія (далі лінійні моделі) для завдань регресії та класифікації відповідно, SVM (з лінійними та RBF ядрами), KNN і неглибокі кількашарові перцептрони (з 1 і 2 шарами, відповідно позначені як MLP1 та MLP2). Кожне сімейство має свої сильні та слабкі сторони, зумовлені їхньою складністю, характером даних і здатністю моделювати нелінійні зв’язки.

Оцінка продуктивності на молекулярних наборах даних включає:

− тренування моделі – використання широкого простору параметрів і алгоритму пошуку оптимізації Байеса для дослідження сітки;

− крос-валідація – застосування 5-кратної CV для оцінки продуктивності моделі та керівництва процесом вибору;

− оцінка на тестових даних – аналіз продуктивності моделі на невідомих даних для оцінки узагальнюваності.

4.2 Цільові метрики оптимізації

У якості основної метрики оптимізації для задач регресії використовується середня абсолютна помилка (MAE), а для бінарної класифікації — F1-метрика.

Середня абсолютна помилка (MAE) є широко використовуваною метрикою у задачах регресії через її інтуїтивно зрозуміле тлумачення та стійкість до викидів. MAE обчислює середню величину помилок між прогнозованими та фактичними значеннями, не беручи до уваги їхній напрямок. Це робить MAE особливо підходящою для застосувань, де важливіше масштаб помилки, ніж її напрямок.

Одна з ключових переваг MAE — її інтерпретованість. Метрика виражена в тих же одиницях, що й цільова змінна, що спрощує розуміння середньої помилки,

яку модель може згенерувати. Крім того, MAE менш чутлива до викидів порівняно з іншими метриками, такими як середньоквадратична помилка (MSE). У відкритті ліків, де набори даних можуть містити шум або аномальні точки даних, стійкість MAE до викидів забезпечує, що продуктивність моделі не буде надмірно впливатися екстремальними значеннями.

F1-метрика — це гармонійне середнє між точністю та відгуком, двома критичними метриками у бінарних задачах класифікації. Точність вимірює пропорцію істинно позитивних прогнозів серед усіх позитивних прогнозів, тоді як відгук оцінює пропорцію істинно позитивних, правильно ідентифікованих серед усіх фактично позитивних. F1-метрика збалансовує ці дві метрики, роблячи її відмінним вибором для сценаріїв, де важливі як точність, так і відгук.

У відкритті ліків, де бінарні класифікаційні завдання часто мають справу з незбалансованими наборами даних (наприклад, ідентифікація активних проти неактивних сполук), F1-метрика особливо корисна. Вона забезпечує, що при оцінці моделі враховується здатність моделі правильно ідентифікувати позитивні випадки (відгук) і мінімізувати кількість хибнопозитивних результатів (точність). Це важливо у відкритті ліків, де неідентифікація потенційно ефективної сполуки (низький відгук) або помилкове визначення сполуки як ефективної (низька точність) може мати значні наслідки.

4.3 Оцінка варіабельності метрик і степеню перенавчання

У прагненні зрозуміти продуктивність моделей та ступінь перенавчання ми пропонуємо проаналізувати відносні різниці між оцінками крос-валідації (CV) та оцінками тестових наборів даних. Аналіз візуалізовано у вигляді графіків (рис. 4.1 та рис. 4.2), де використовуються дві ключові метрики: відносна різниця (RD або rel_{diff}) та відносна абсолютна різниця (RAD або $rel_{abs_{diff}}$).

Метрика відносної різниці дозволяє зрозуміти, чи моделі перевищують або не досягають показників на тестовому наборі порівняно з CV-набором. Вона обчислюється як різниця між оцінкою тесту та CV, нормалізована CV-оцінкою:

$$\mathrm{r}el_{diff} = \frac{test - cv}{cv}, \#(4.1)$$

де $\mathrm{r}el_{diff}$ – відносна різниця;

$test$ – значення метрики на тестовій вибірці;

cv – значення метрики на крос-валідаційній вибірці.

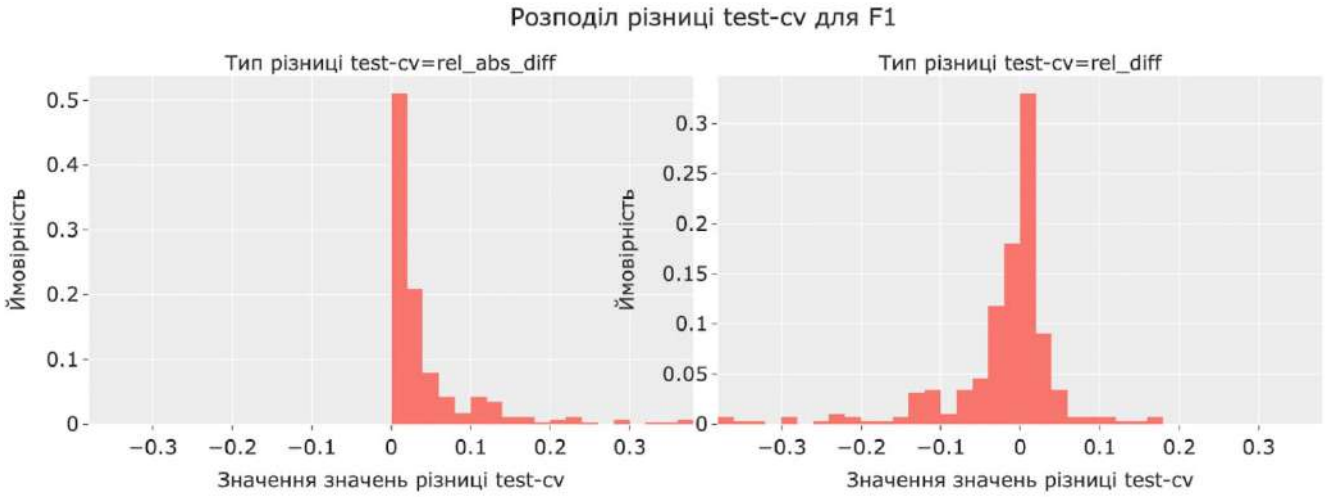


Рисунок 4.1 – Загальний розподіл $\mathrm{r}el_{abs_{diff}}$ і $\mathrm{r}el_{diff}$ для F1

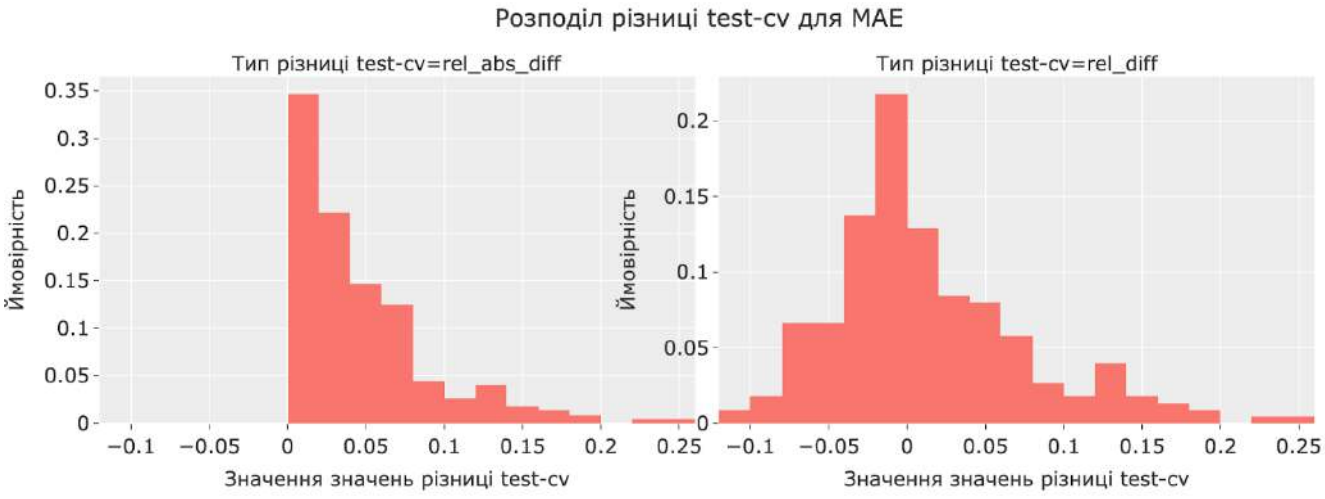


Рисунок 4.2 – Загальний розподіл відмінностей $\mathrm{r}el_{abs_{diff}}$ і $\mathrm{r}el_{diff}$ для MAE

Метрика Відносної Абсолютної Різниці, у свою чергу, відображає магнітуду відхилення між оцінками тесту та CV незалежно від напрямку відхилення. Вона розраховується як абсолютне значення різниці між тестовими та CV оцінками, нормалізоване за CV-оцінкою:

$$\mathrm{r}el_{abs_{diff}} = \frac{|test - cv|}{cv}, \#(4.2)$$

де $\mathrm{r}el_{abs_{diff}}$ – відносна абсолютна різниця;

$test$ – значення метрики на тестовій вибірці;

cv – значення метрики на крос-валідаційній вибірці.

Ці метрики є цінними інструментами для оцінки стабільності моделі та її узагальнюваності. Метрика $\mathrm{r}el_{diff}$ показує, чи має модель тенденцію до перенавчання або недонавчання даних, як це видно з її продуктивності на невідомих тестових даних у порівнянні з даними CV. Позитивне значення $\mathrm{r}el_{diff}$ свідчить про те, що модель краще працює на тестовому наборі, ніж на CV, що може вказувати на недонавчання, тоді як негативне значення $\mathrm{r}el_{diff}$ свідчить про перенавчання.

Метрика $\mathrm{r}el_{abs_{diff}}$ надає оцінку загальної варіативності продуктивності моделі між CV та тестовими наборами. Нижчі значення $\mathrm{r}el_{abs_{diff}}$ вказують на те, що продуктивність моделі послідовна між CV та тестовими наборами, що свідчить про хорошу узагальнюваність. Навпаки, вищі значення $\mathrm{r}el_{abs_{diff}}$ вказують на більшу варіативність продуктивності, сигналізуючи про потенційні проблеми зі стабільністю та узагальнюваністю моделі.

Аналіз рисунку 4.1 показує, що більше ніж 82.5% абсолютних різниць у класифікації є меншими або рівними 7.5%. Тому розумно встановити поріг толерантності T у межах 2-7%. Моделі класифікації, як правило, показують нижчі результати на тестовому наборі. Викиди відповідають наборам даних низької або сумнівної якості.

На рисунку 4.2 вказано, що більше ніж 83% регресійних абсолютних різниць не перевищують 8%. Так само, як і результати з аналізу класифікації, розумно встановити поріг толерантності T у межах 2-7%. Моделі регресії мають більшу ймовірність показати кращі результати на тестовому наборі, ніж моделі класифікації. Більшість викидів відповідає наборам даних з незбалансованими розподілами.

На рисунку 4.3 показано розподіл RD та RAD для класифікації (F1) та регресії (MAE) залежно від сімейства моделей.

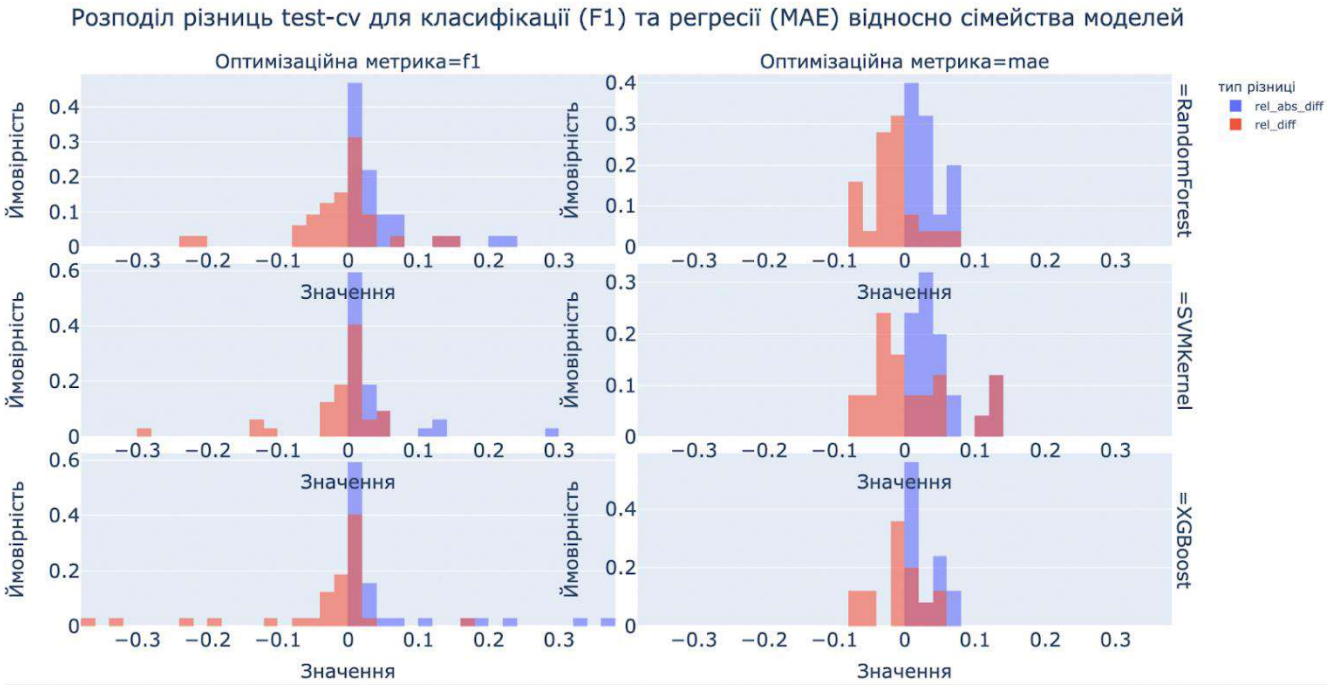


Рисунок 4.3 – Розподіл RD та RAD для класифікації (F1) та регресії (MAE) залежно від сімейства моделей. Приклад для 3 сімейств

У контексті завдань класифікації аналіз показує, що такі сімейства моделей, як SVMKernel, лінійні, MLP1, XGBoost та CatBoost, виявляють помітну стабільність. Зокрема, ці моделі демонструють імовірність щонайменше 0.6, що їх Відносна Абсолютна Різниця (RAD) є меншою або рівною 2.5%. Це вказує на меншу схильність цих сімейств моделей до перенавчання та більшу послідовність їхньої продуктивності між CV та тестовими наборами. Однак спостерігається, що

моделі, загалом, мають тенденцію показувати гірші результати на тестовій валідації, що вказує на потенційне перенавчання до CV-набору.

Для інших моделей машинного навчання графіки наведено на рис 4.4.

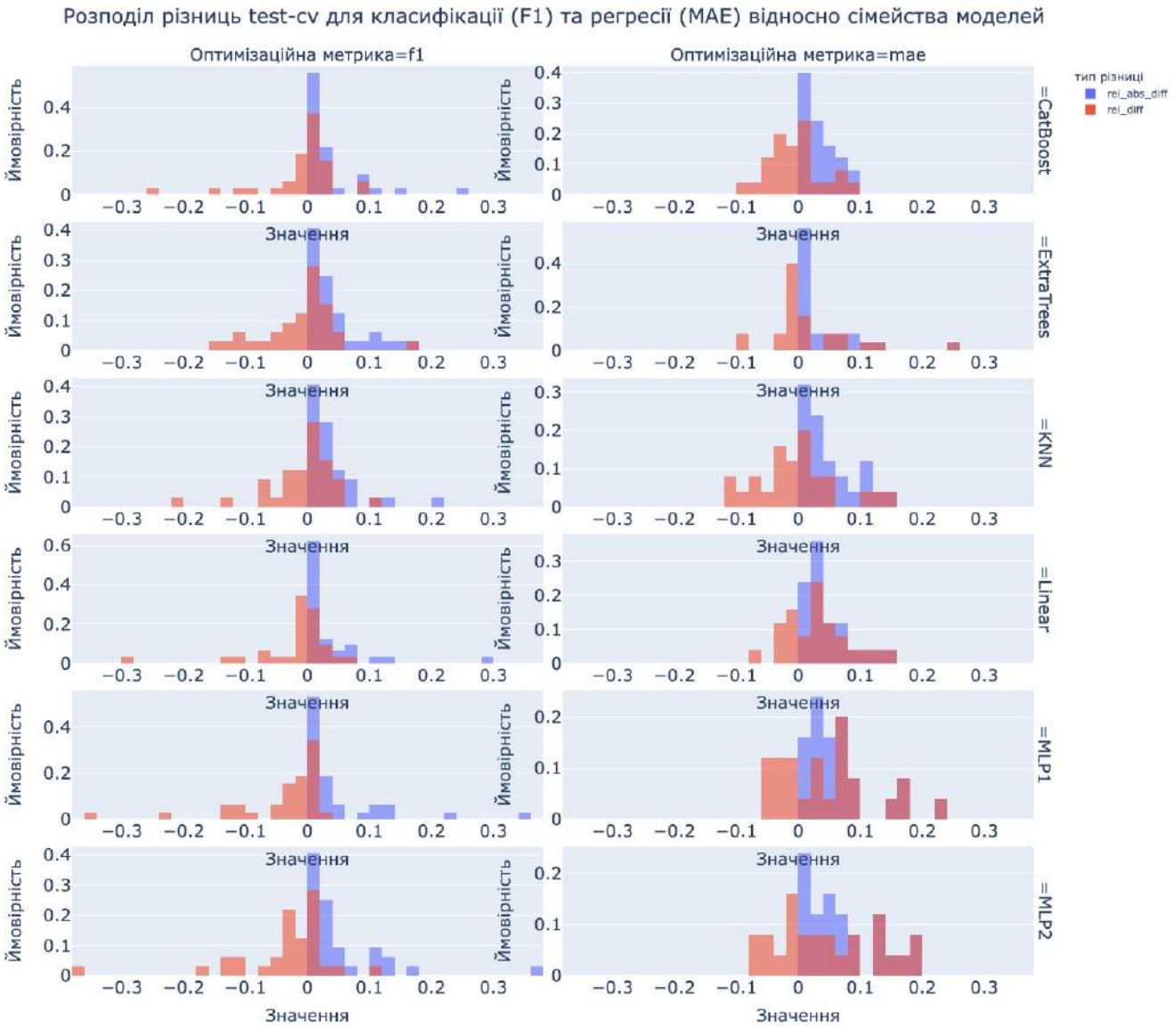


Рисунок 4.4 – Розподіл RD та RAD для класифікації (F1) та регресії (MAE) залежно від сімейства моделей. Графіки для інших сімейств моделей

Для завдань регресії XGBoost і RandomForest виявляються найстабільнішими сімействами моделей, з максимальним RAD менше 7.5%. CatBoost слідує тісно за ними як конкурентна альтернатива, з максимальним RAD менше 10%. Незважаючи на деякі сімейства моделей, що показують густі зони біля нуля, що вказує на гарну послідовність продуктивності, вони також

виявляють "товстохвості" розподіли RAD. Ця характеристика вказує на окремі випадки значної нестабільності.

Цікаво, що сімейства моделей у завданнях регресії частіше показують кращі результати на тестовій валідації, що свідчить про те, що розподіл тренувальних/тестових даних для цих завдань вимагає додаткового дослідження для забезпечення надійної оцінки моделі.

Загалом, алгоритм вибору найкращої моделі на основі найвищих оцінок CV визначив сімейства моделей SVM і MLP (з 2 і 1 шарами) як найефективніші для даного набору даних.

На рисунку 4.5 зображено топ-6 сімейств моделей. Вісь x відповідє кількості наборів даних, на яких сімейство моделей було найефективнішим.

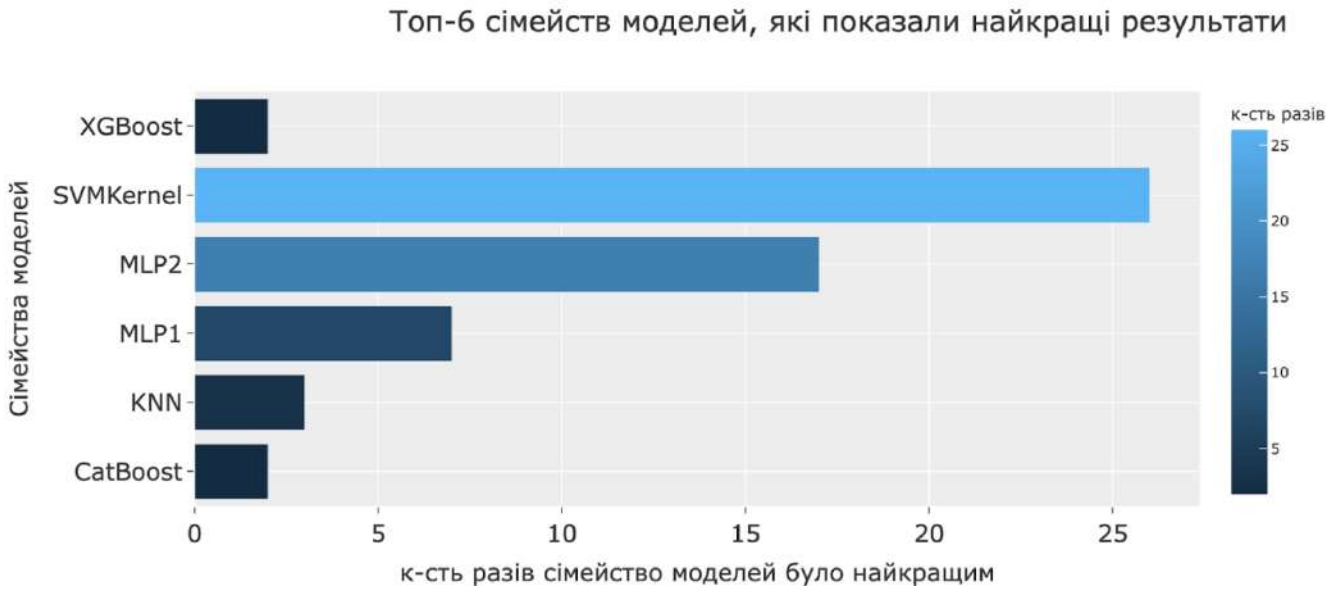


Рисунок 4.5 – Топ-6 сімейств моделей

Попередній аналіз містив порівняння щодо основних метрик оптимізації. Хоча критично важливо досліджувати кілька метрик для виявлення різних аспектів поведінки моделі, одночасна оптимізація всіх з них є складним завданням. Отже, виникає важливе питання: наскільки варіабельними є допоміжні

метрики відносно значень основної метрики? Для вирішення цього завдання використовується візуальний аналіз графіків, наведених на рисунку 4.6.

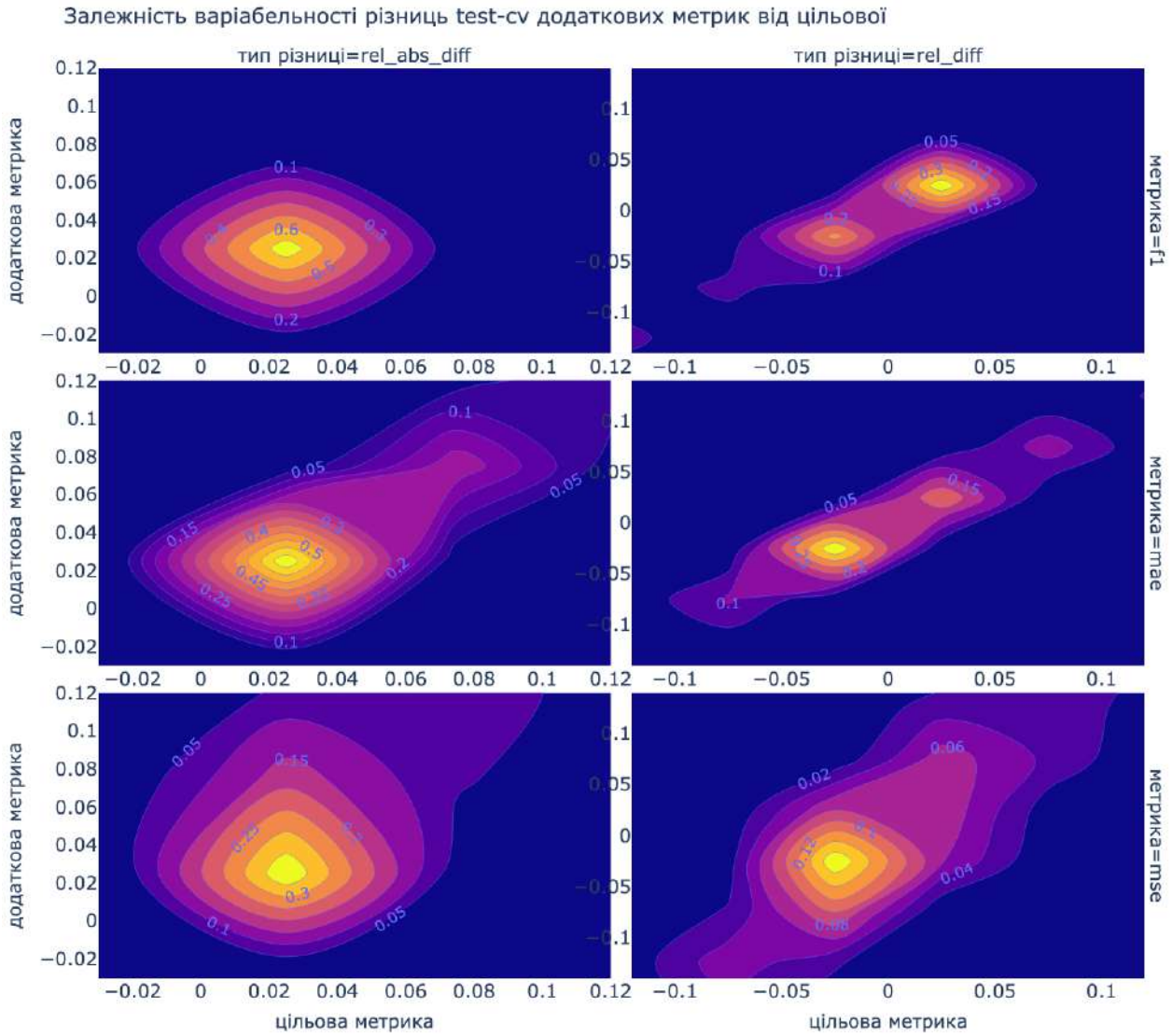


Рисунок 4.6 – Залежність варіабельності допоміжних метрик від різниці метрики оптимізації

Точність: ця метрика показує найменшу варіабельність у відповідності до змін у F1-оцінці, що вказує на її відносну стабільність при різних продуктивностях моделі.

Логарифмічна втрата: ця метрика чутлива до змін у F1-оцінці, зі зростанням логарифмічної втрати при зниженні F1. Ця залежність підкреслює вплив точності прогнозу на логарифмічну втрату.

Точність і відгук: хоча обидві метрики вносять вклад у F1-оцінку, аналіз показує, що зміни у F1-оцінці переважно лінійно залежать від змін у відгуку, а не у точності. Це свідчить про те, що зміни в продуктивності моделі, швидше за все, пов'язані зі здатністю моделі виявляти істинно позитивні випадки (відгук) замість її точності у класифікації позитивних прогнозів (точність).

Значуще спостереження: коли відгук знижується на тестовому наборі, а точність залишається майже незмінною, це може вказувати на те, що модель правильно класифікує молекули тестового набору, схожі на позитивні молекули в тренувальному наборі (отже стабільна точність), але не вдається ідентифікувати позитивні молекули в тестовому наборі з новими або недостатньо представленими структурами. Цей сценарій підкреслює потенційну цінність стратифікації тренувальних/тестових розділів на основі молекулярних кластерів і цільових значень для покращення нашого розуміння поведінки моделі.

ROC AUC: не спостерігається залежності між ROC AUC і F1-оцінкою, що свідчить про те, що площа під кривою приймача залишається незмінною незалежно від змін у балансі між точністю і відгуком.

MAPE (середня абсолютна відсоткова помилка): ця метрика демонструє високу нестабільність незалежно від змін у MAE (середня абсолютна помилка), що вказує на її чутливість до змін у продуктивності моделі.

MSE (середньоквадратична помилка): MSE показує квадратичне зростання з підвищенням MAE, відображаючи посилення помилок у метриках квадратичної помилки.

RMSE (коренева середньоквадратична помилка): RMSE зростає як корінь зі збільшенням MAE, забезпечуючи більш помірну реакцію на посилення помилок порівняно з MSE.

R^2 (Коефіцієнт Детермінації): ця метрика знижується квадратично зі збільшенням MAE, що вказує на зниження пояснювальної сили моделі зі зростанням помилок.

4.4 Запропоновані сімейства скорингових функцій

Як ми бачили в підрозділах 4.2 та 4.3, вибір найкращої моделі на основі оцінок крос-валідації (CV) часто призводить до перенавчання, оскільки моделі можуть ефективно працювати на CV-наборі, але показувати значне зниження продуктивності на тестовому наборі. У таблиці 4.1 наведено усереднені статистичні дані.

Таблиця 4.1 – Усереднені статистичні дані.

Метрика оптимізації	Метрика	Середнє	Стандартне відхилення
F1	rel_abs_diff	0.049	0.075
	rel_diff	-0.039	0.080
MAE	rel_abs_diff	0.058	0.058
	rel_diff	0.027	0.078

Щоб вирішити проблему, описану вище, ми вводимо три нові функції оцінювання, які включають відносну абсолютну різницю (RAD) між оцінками CV та тестовими оцінками, а також саму оцінку CV. Ці функції мають на меті винагородити моделі з нижчим перенавчанням та штрафувати ті, що мають вищий RAD, тим самим сприяючи вибору більш узагальнюваних моделей.

Лінійна функція оцінювання:

$$f_{lin} = CV + CV * M * (-R + T), \#(4.3)$$

де CV – оцінка CV за метрикою оптимізації;

R – відносна абсолютна різниця між оцінками CV та тестовими оцінками;

T – поріг толерантності, що представляє максимально допустиму величину R;

M – частка оцінки CV, що додається як бонус у ідеальному випадку, коли $R = 0$.

Експоненціальна функція оцінювання:

$$f_{exp} = CV + CV * M * \left(-EXP(LN(2) * \frac{R}{T}) + 2\right), \#(4.4)$$

де CV – оцінка CV за метрикою оптимізації;

R – відносна абсолютна різниця між оцінками CV та тестовими оцінками;

T – поріг толерантності, що представляє максимально допустиму величину R;

M – частка оцінки CV, що додається як бонус у ідеальному випадку, коли $R = 0$.

Квадратична експоненціальна функція оцінювання:

$$f_{exp^2} = CV + CV * M * \left(-EXP(LN(2) * \frac{R^2}{T^2}) + 2\right), \#(4.5)$$

де CV – оцінка CV за метрикою оптимізації;

R – відносна абсолютна різниця між оцінками CV та тестовими оцінками;

T – поріг толерантності, що представляє максимально допустиму величину R;

M – частка оцінки CV, що додається як бонус у ідеальному випадку, коли $R = 0$.

Ці функції розроблені для додавання додаткових балів до оцінки CV, заохочуючи моделі з нижчим рівнем перенавчання. Коли R дорівнює порогу T, кінцева оцінка дорівнює оцінці CV. Якщо R перевищує поріг, оцінка продуктивності моделі штрафується, отримуючи значення нижче оцінки CV. Значно, що нелінійні функції оцінювання штрафують агресивніше, ніж винагороджують, що відображає обережний підхід до перенавчання.

Лінійна функція оцінювання пропонує простий підхід, лінійно коригуючи оцінку залежно від відхилення від порога толерантності. Натомість, експоненціальна та квадратична експоненціальна функції вводять нелінійний штраф за перенавчання, причому остання є агресивнішою у штрафуванні моделей, коли RAD зростає. Ці нелінійні функції особливо корисні у сценаріях, де потрібен більш суворий контроль над перенавчанням.

Щоб краще зрозуміти, як поводять себе запропоновані функції оцінювання, детальніше розглянемо графіки на рисунку 4.7, рисунку 4.8 та рисунку 4.9. Для всіх графіків зафіксуємо CV = 0.5 і T = 0.05 та виберемо 0.025, 0.05 і 0.075 як три різні значення максимальної винагороди M. Тоді графіки ілюструють, як різні функції оцінювання та значення M впливають на кінцеву корекцію оцінки CV на основі значення R.

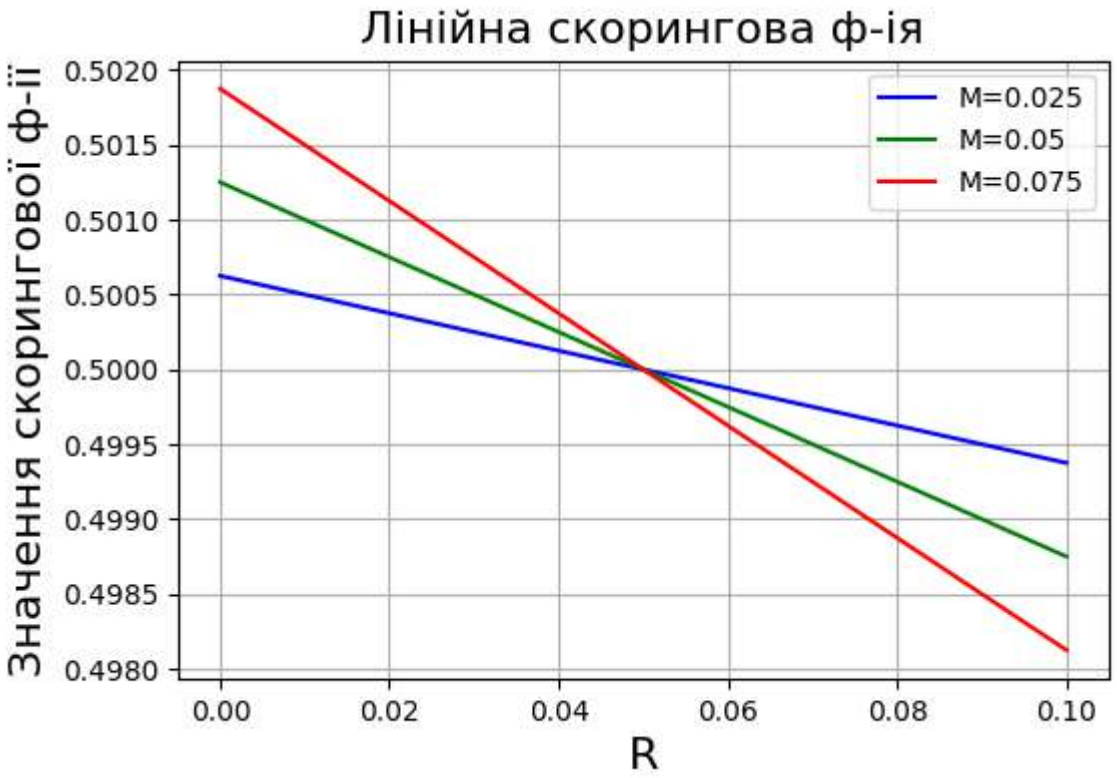


Рисунок 4.7 – Лінійна скорингова функція

Для вибору найоптимальніших значень параметрів M і T ми починаємо з розрахунку оцінок для кожної моделі за допомогою лінійної, експоненціальної і

квадратичної експоненціальної функцій, з конкретними значеннями для порога толерантності (T) та частки винагороди (M). Оптимальне значення M шукається для збалансування винагороди за хорошу продуктивність на наборі CV і штрафу за перенавчання. Функції оцінювання застосовуються до набору даних, що містить оцінки CV моделей та їхні значення RAD, з окремими підходами для метрик оптимізації, таких як середня абсолютна помилка (MAE) та оцінка F1.

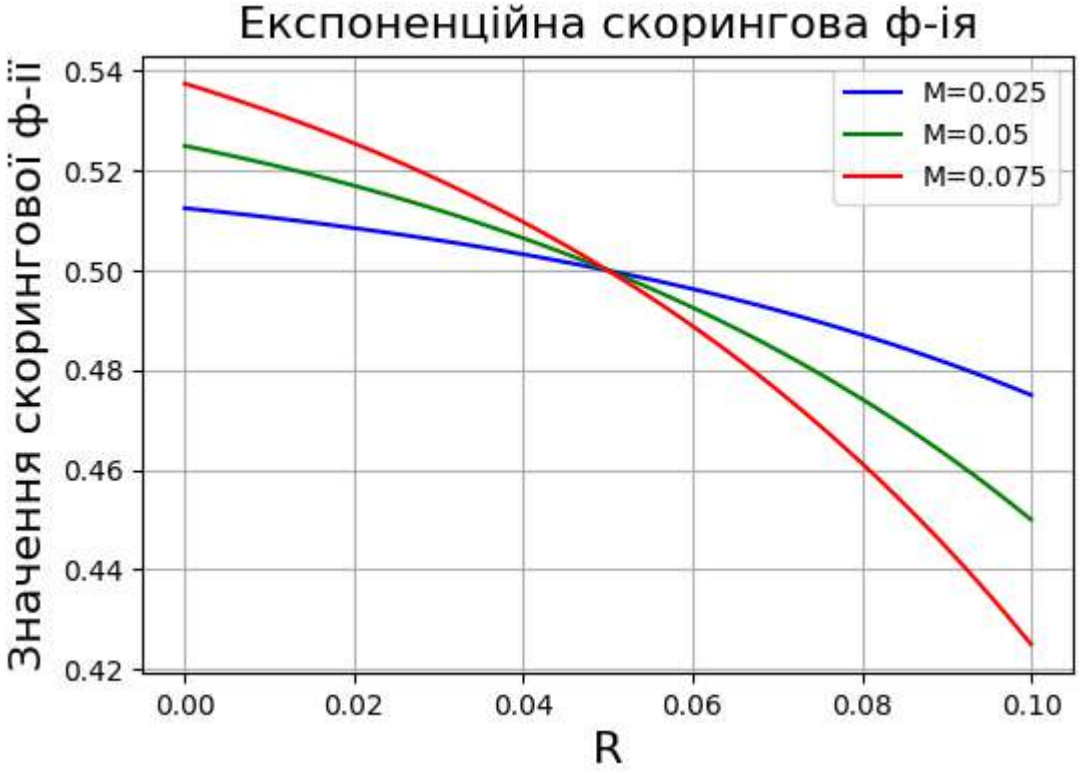


Рисунок 4.8 – Експоненційна скорингова функція

Проводиться всебічний аналіз шляхом зміни значень T і M і спостереження за впливом на стабільність моделі. Найкраща модель для кожної функції оцінювання ідентифікується на основі її продуктивності у завданнях класифікації або регресії.

На рисунку 4.10 зображено середнє RAD над найкращими моделями з усіх наборів даних, вибраними різними функціями оцінювання та їх параметрами.

Аналіз виявляє цікаві висновки щодо поведінки функцій оцінювання:

– лінійна функція оцінювання має незначний вплив порівняно з підходом вибору максимального значення CV, що свідчить про її обмежену ефективність у штрафуванні за перенавчання;

– для класифікаційних завдань експоненціальна функція оцінювання дає найстабільніші моделі, тоді як для завдань регресії більш ефективною виявляється квадратична експоненціальна функція;

– збільшення винагороди (M) здається корисним для знаходження більш стабільних моделей, навіть коли абсолютне значення M перевищує поріг толерантності (T).

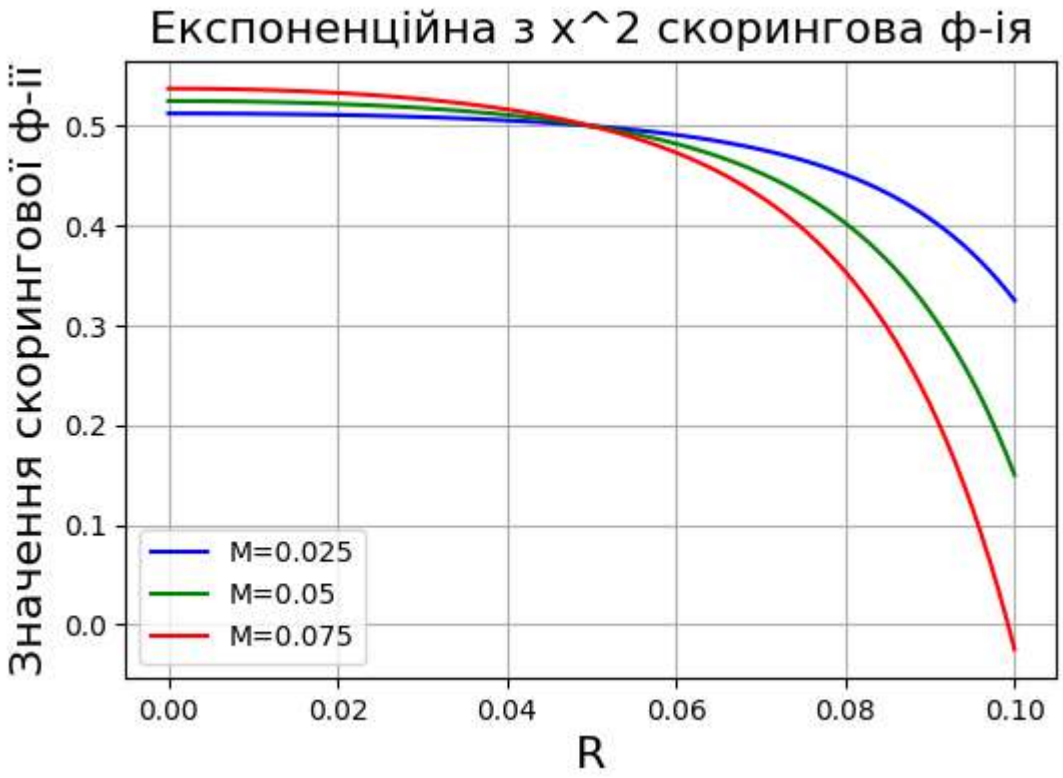


Рисунок 4.9 – Експоненційна з x^2скорингова функція

Однак важливо зазначити, що цей аналіз зосереджений виключно на відносній продуктивності різних функцій оцінювання з точки зору стабільності моделі. Він не враховує абсолютні значення метрик самі по собі. В результаті, хоча збільшення |M| > T може призвести до більш стабільних моделей, це

потенційно може вести до вибору простіших моделей зі стабільними, але поганими оцінками CV та тестовими результатами.

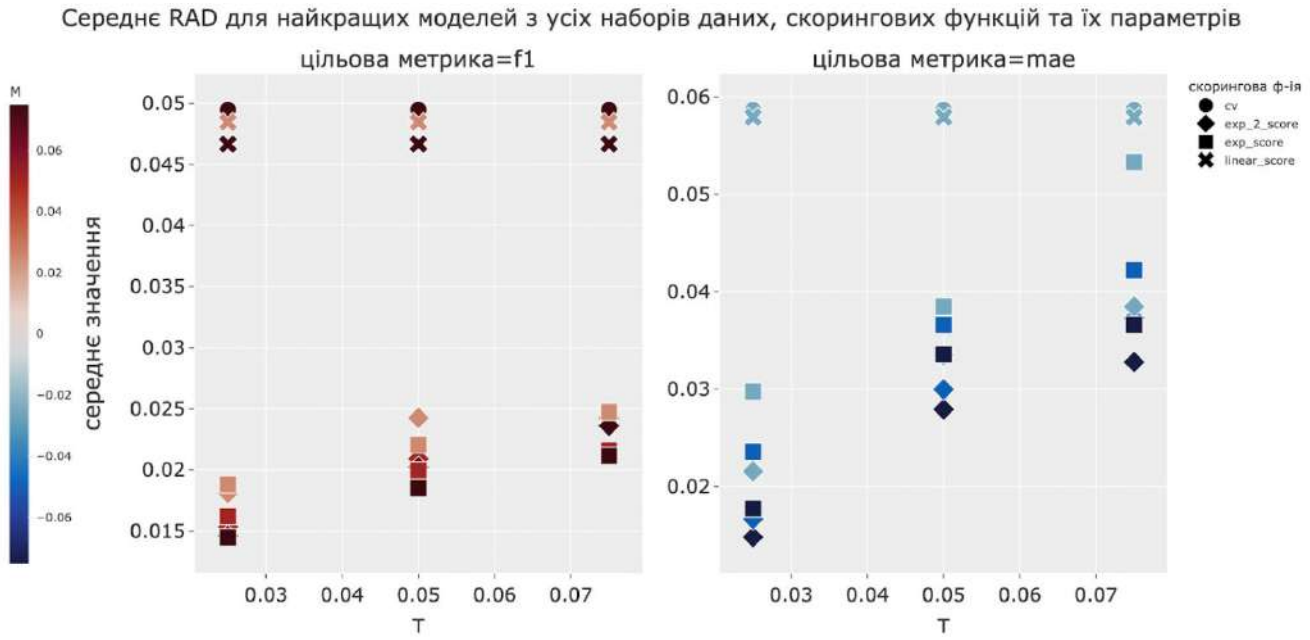


Рисунок 4.10 – Середнє RAD над найкращими моделями з усіх наборів даних, вибраними різними функціями оцінювання та їх параметрами

4.5 Оцінка впливу скорингових функцій на вибір моделей в AutoML пайплайні

У пошуках оптимальних стратегій вибору моделей введення нових скорингових функцій виявилося обнадійливим у вирішенні проблем перенавчання. Цей розділ досліджує вплив цих скорингових функцій на частоту, з якою різні сімейства моделей обираються як найкращі в класифікаційних та регресійних завданнях.

Аналіз використовує стовпчасті діаграми для порівняння частот, з якими різні сімейства моделей вибираються як найкращі різними алгоритмами, з особливим акцентом на задачах класифікації, оптимізованих для оцінки F1, та регресійних задачах, оптимізованих для середньої абсолютної помилки (MAE). Діаграми сегментовані залежно від порогу толерантності (T) та абсолютної

величини частки винагороди (M), що дозволяє детально вивчити їхній вплив на вибір моделей.

Рисунки 4.11 та 4.12 виявляють помітну зміну в уподобаннях вибору моделей при застосуванні скорингових функцій.

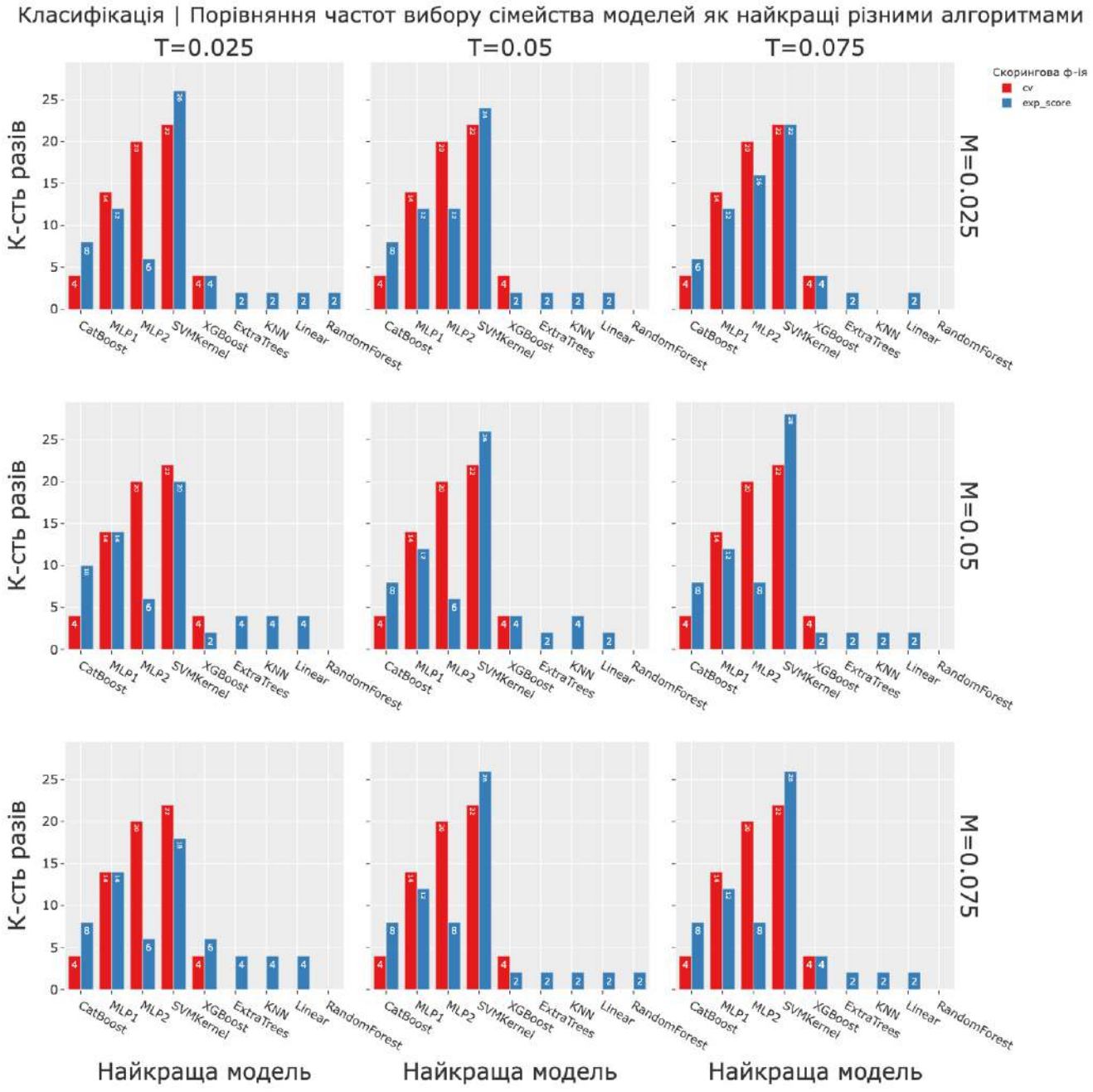


Рисунок 4.11 – Порівняння частот, із якими сімейство моделей обирається як найкраще різними алгоритмами для класифікацій

У контексті класифікації частота вибору MLP2 як найкращої моделі значно знижується, з відповідним збільшенням вибору CatBoost. Це свідчить про те, що

MLP2 може схильна до перенавчання, що призводить до високої варіативності її оцінок CV та тестових результатів, в той час як CatBoost виявляється більш стабільним і бажаним варіантом.

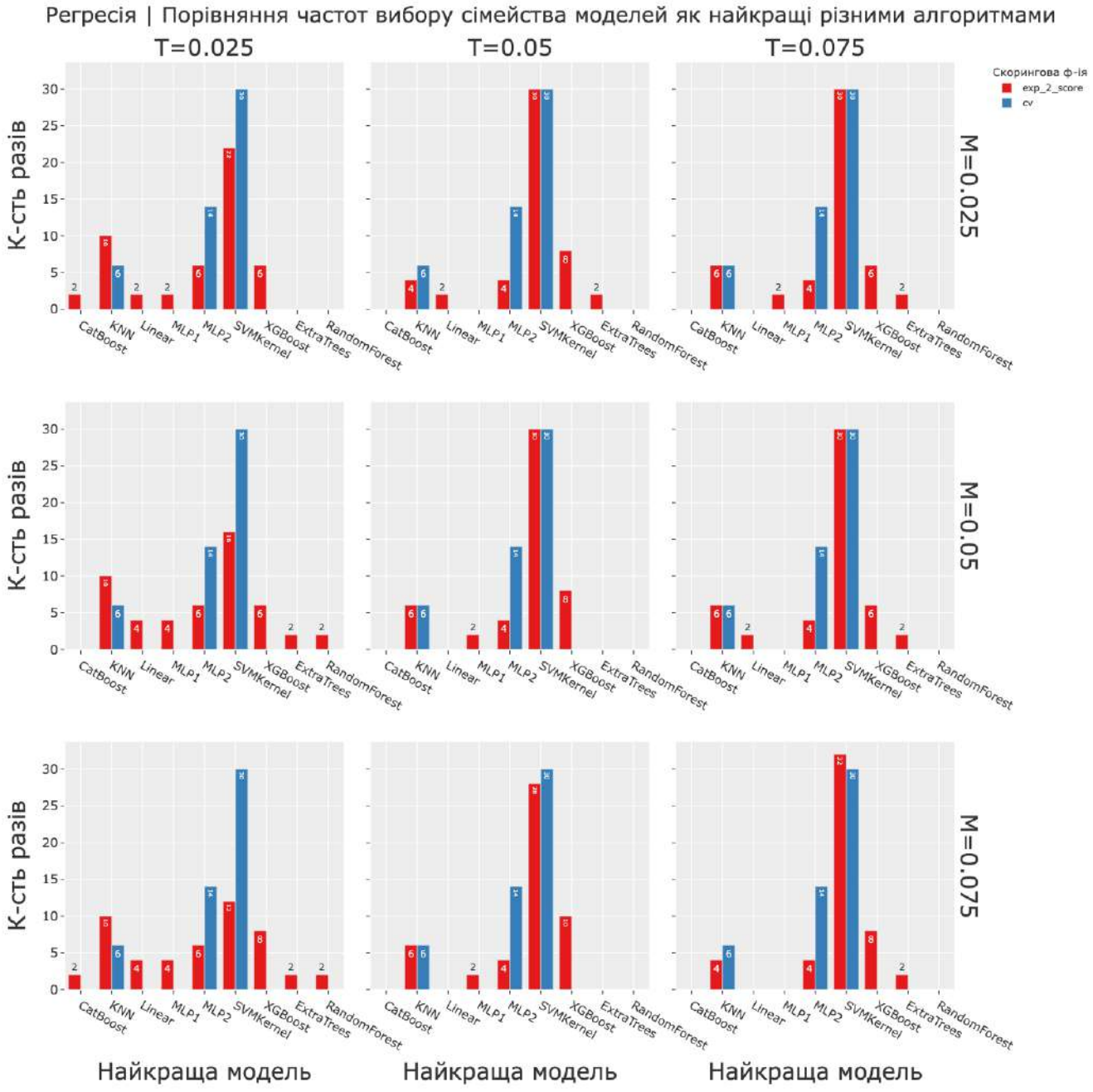


Рисунок 4.12 – Порівняння частот, із якими сімейство моделей обирається як найкраще різними алгоритмами для регресій

Аналогічно, у регресійних завданнях спостерігається помітне зниження вибору MLP2, з ростом популярності XGBoost як переважної моделі. Це вказує на

те, що XGBoost пропонує більш міцну та послідовну продуктивність у регресійних налаштуваннях порівняно з MLP2.

Аналіз також натякає на потенційний недолік використання частки винагороди (|M|) більше, ніж поріг толерантності (T). Такий підхід, здається, віддає перевагу простішим моделям, як-от KNN та лінійні моделі, які, незважаючи на їх стабільність, можуть не демонструвати високу продуктивність. Ймовірно, вони показуватимуть схожу посередню продуктивність як на CV, так і на тестових наборах, що призводить до їх збільшеної частоти вибору. Це відбувається за рахунок більш складних моделей, як-от SVM, які відомі своєю міцністю та вищою продуктивністю.

4.6 Аналіз експериментів і рекомендації щодо параметризації скорингових функцій

Цей розділ зосереджується на аналізі порівняння продуктивності моделей, обраних на основі оцінок крос-валідації (CV) та моделей, вибраних за допомогою експоненціальної (exp_score для класифікації) та квадратичної експоненціальної (exp_2_score для регресії) функції. Основна увага приділена розумінню впливу різних значень частки винагороди (M) на вибір моделі. Для цього візуалізовано відносні різниці в метриках CV (по осі x) і тесту (по осі y) моделей, обраних різними скоринговими функціями, у вигляді точкової діаграми, де різні кольори представляють різні значення M. На рисунку 4.13 зображено відносну різницю метрик CV (вісь x) та тесту (вісь y) F1 для моделей, обраних як найкращі експоненціальною функцією оцінювання.

Точкова діаграма виявляє декілька ключових спостережень.

Зменшення перенавчання. Функція exp_score, особливо з M=0.025, схильна вибирати моделі, які показують незначне зниження оцінок CV, але значне покращення тестових оцінок. Це підтверджується поширеністю точок у регіоні, де відносна різниця оцінок CV (RD_CV) є негативною, а відносна різниця тестових оцінок (RD_TEST) позитивною.

У деяких випадках, як-от у наборі даних CYP_Inh_3A4, найкраща модель, обрана на основі оцінок CV, перевершує модель, вибрану за допомогою exp_score, як у CV, так і у тестових оцінках. Це відбувається, коли винагорода за послідовність (M=0.05) є надмірно високою, дозволяючи моделі з нижчими загальними оцінками, але мінімальною різницею між оцінками CV і тесту, перевищити модель з вищими оцінками. Зменшення M до 0.025 пом'якшує цю проблему, хоча і не усуває її повністюНа точковій діаграмі менше точок розташовано нижче горизонтальної осі для M=0.025 (сині точки) порівняно з M=0.05 (червоні точки), що вказує на те, що зменшення M знижує ймовірність вибору моделей з нижчими тестовими оцінками.

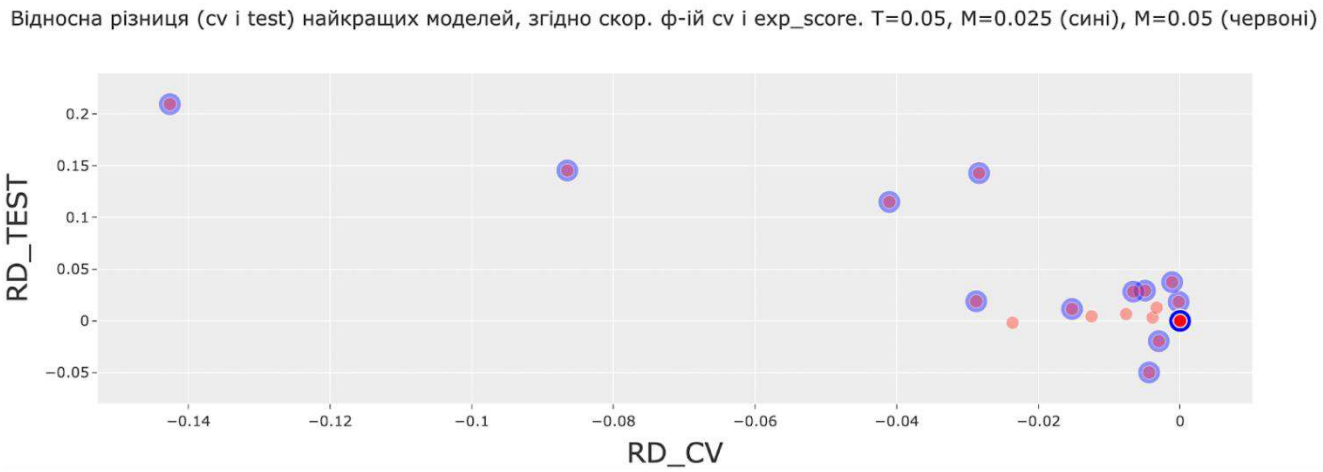


Рисунок 4.13 – Тип задачі класифікації. Відносна різниця метрик CV (вісь x) та тесту (вісь y) F1 для моделей, обраних як найкращі експоненціальною функцією оцінювання. T=0.05, M=0.025 (синій), M=0.05 (червоний)

Загалом аналіз підкреслює потенціал функції exp_score у боротьбі з перенавчанням у класифікаційних завданнях. Ретельно регулюючи частку винагороди (M), можна знайти баланс між винагородою за послідовність та забезпеченням високої продуктивності. Для класифікаційних проблем рекомендується використовувати функцію exp_score з порогом толерантності (T) 0.05 та часткою винагороди (M) 0.025. Цей підхід ефективно знижує середню

відносну абсолютну різницю між оцінками CV і тесту з майже 5% до 2.1%, демонструючи значне покращення узагальнюваності моделі.

На рисунку 4.14 зображено відносна різницю метрик CV (вісь x) і тесту (вісь y) MAE для моделей, обраних як найкращі квадратичною експоненціальною функцією оцінювання.

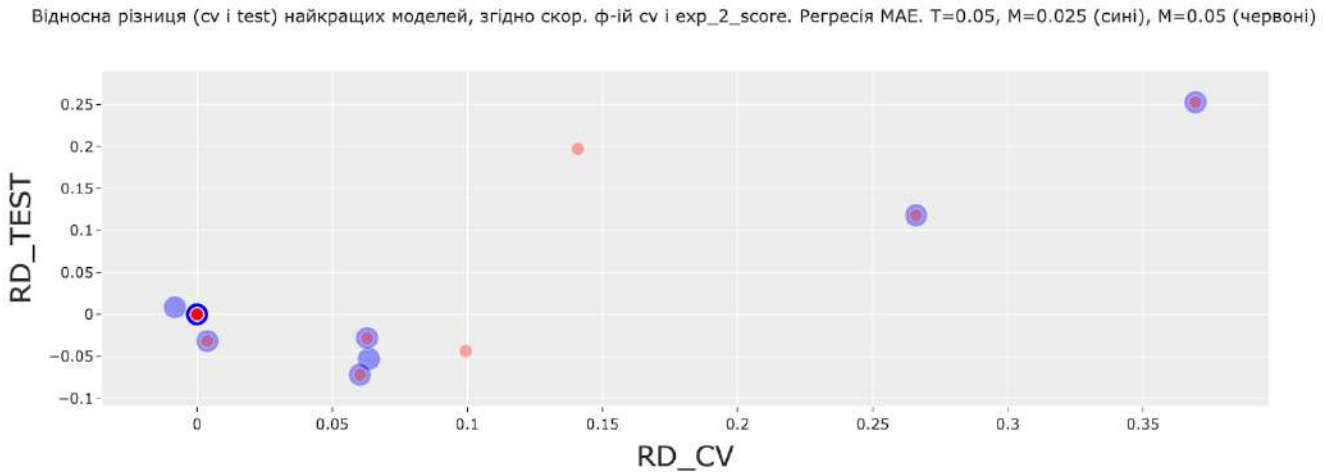


Рисунок 4.14 – Тип завдання регресії. Відносна різниця метрик CV (вісь x) і тесту (вісь y) MAE для моделей, обраних як найкращі квадратичною експоненціальною функцією оцінювання. T=0.05, M=0.025 (синій), M=0.05 (червоний)

Що стосується регресійних завдань, аналіз виявляє кілька важливих моментів.

Функція exp_2_score, особливо з M=0.025, схильна вибирати моделі, які показують невелике зниження оцінок CV, але помітне збільшення тестових оцінок. Ця тенденція ілюструється поширеністю точок у регіоні, де відносна різниця в оцінках CV (RD_CV) є позитивною, а відносна різниця в тестових оцінках (RD_TEST) негативною.

Зменшення M до 0.025 знижує частоту випадків, коли модель з нижчими оцінками CV і тесту обирається над моделлю з вищими оцінками виключно через її послідовність між оцінками CV і тесту.

Розподіл точок: на точковій діаграмі спостерігається менше точок вище горизонтальної осі для M=0.025 (сині точки) порівняно з M=0.05 (червоні точки),

що вказує на те, що нижче значення M знижує ймовірність вибору моделей з нижчими тестовими оцінками.

На завершення, функція exp_2_score ефективна у боротьбі з перенавчанням. Регулюючи частку винагороди (M) до 0.025, процес вибору сприяє моделям, які не тільки добре виступають на наборі CV, але й зберігають свою продуктивність на тестовому наборі. Для регресійних завдань радять використовувати функцію exp_2_score з порогом толерантності (T) 0.05 та часткою винагороди (M) 0.025. Цей підхід призводить до значного зниження середньої відносної абсолютної різниці між оцінками CV і тесту, з майже 6% до 3.3%, що вказує на покращення узагальнюваності моделі.

4.7 Висновки

Це дослідження було спрямоване на розробку та оцінку пайплайну AutoML, спеціально адаптованого для навчання моделей машинного навчання в області ADME-Tox. Було проведено всеосяжний аналіз 38 різноманітних наборів даних для оцінки ефективності пайплайну.

Однією з критичних проблем, виявлених під час цього дослідження, була схильність до перенавчання на наборах даних крос-валідації (CV), що є поширеною пасткою в стандартних алгоритмах автоматизованого вибору моделі. Вплив цього перенавчання був ретельно проаналізований для розуміння його наслідків для узагальнюваності моделі.

Дослідження також охопило продуктивність десяти широко використовуваних алгоритмів машинного навчання, і було виявлено, що машини опорних векторів (SVM) послідовно виявлялися найбільш міцними та продуктивними алгоритмами для малих та середніх молекулярних наборів даних. Алгоритми на основі дерев підсилення, такі як XGBoost, також продемонстрували надійну продуктивність. Однак архітектури нейронних мереж, ймовірно через їхню складність, виявилися більш схильними до перенавчання, що підкреслює необхідність уважного вибору та валідації моделі.

Для вирішення виявлених проблем був введений новий підхід, який включав три скорингові функції: лінійну, експоненційну та квадратичну експоненційну. Ці функції були ретельно випробувані для визначення їхньої ефективності у зменшенні перенавчання та покращенні вибору моделі.

Для завдань регресії квадратична експоненційна функція оцінювання з порогом толерантності (T) 0.05 та часткою винагороди (M) 0.025 була визначена як оптимальний вибір. Ця конфігурація значно знизила середню відносну абсолютну різницю між оцінками CV та тестовими результатами з приблизно 6% до 3.3%, свідчачи про покращення стабільності та узагальнюваності моделі.

У контексті завдань класифікації експоненційна функція оцінювання виявилася найкращим варіантом, з тим самим порогом толерантності та часткою винагороди, як у завданнях регресії. Цей підхід ефективно зменшив середню відносну абсолютну різницю між оцінками CV та тестовими результатами з майже 5% до 2.1%, додатково підтверджуючи користь запропонованих скорингових функцій у покращенні вибору та продуктивності моделі.

Загалом це дослідження вносить цінні ідеї та методології у галузь комп'ютерного відкриття ліків, пропонуючи міцний каркас для розробки надійних та узагальнюваних моделей машинного навчання для точок ADME-Tox.

ВИСНОВКИ

На основі проведених досліджень було розроблено 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 матиматичну модель основних компонентів пайплайну. Розглянуто принцип роботи таких алгоритмів класичного машинного начання, 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-Cloud деплойменту на Kubernetes кластерах хмарних провайдерів. Також наводиться порівняння різних технологій для реалізації поставленої задачі і обгрунтовується фінальне рішення.

У четвертому розділі міститься опис створення скорингових функцій для вибору найкращої моделі отриманої автоматичним навчанням. Проведено ретельне порівняння запропонованих сімейств скорингових функцій для задач бінарної класифікації та регресії, та обрано найкращі скорингові функції та значення їх параметрів для задач регресії та класифікації по критерії максимальної робастності натренованих моделей.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Krieger, M. T., Torreno, O., Trelles, O., Kranzlmüller, D. Building an open source cloud environment with auto-scaling resources for executing bioinformatics and biomedical workflows. *Future Generation Computer Systems*. 2020. № 67. P. 329-340.

2. Augustyn, D. R., Wyciślik, Ł., Mrozek, D. Perspectives of using Cloud computing in integrative analysis of multi-omics data. *Briefings in Functional Genomics*. 2021. № 20. P. 198-206.

3. Sheffield, N. C., Bonazzi, V. R., Bourne, P. E., Burdett, T., Clark, T., Grossman, R. L., Yates, A. D. From biomedical cloud platforms to microservices: next steps in FAIR data and analysis. *Scientific Data*. 2022. № 9. P. 553.

4. Rusinko, A., Rezaei, M., Friedrich, L., Buchstaller, H. P., Kuhn, D., Ghogare, A. Empowering Drug Discovery with AI/ML and CADD Tools in a Secure, Web-Based SaaS Platform. *Journal of Chemical Information and Modeling*. 2023. № 113. P. 190-202.

5. Spjuth, O., Frid, J., Hellander, A. The machine learning life cycle and the cloud: implications for drug discovery. *Expert Opinion on Drug Discovery*. 2021. № 16. P. 1071-1079.

6. Kumar, S. A., Ananda Kumar, T. D., Beeraka, N. M., Pujar, G. V., Singh, M., Narayana Akshatha, H. S., Bhagyalalitha, M. Machine learning and deep learning in data-driven decision making of drug discovery and challenges in high-quality data acquisition in the pharmaceutical industry. *Future Medicinal Chemistry*. 2022. № 14. P. 245-270.

7. Dall'Alba, Gabriel, et al. A survey of biological data in a big data perspective. *Big Data*. 2022. № 10. P. 279-297.

8. Selvaraj, Gurudeeban, et al. Application of artificial intelligence in drug repurposing: A mini-review. *Current Chinese Science*. 2021. № 10. P. 333-345.

9. Mitra, Debasis, et al. Evolution of Bioinformatics and its impact on modern bio-science in the twenty-first century: Special attention to pharmacology, plant science

and drug discovery. *Computational Toxicology*. 2022. № 10. P. 100-108. URL: https://www.sciencedirect.com/science/article/abs/pii/S2468111322000366

10. Chen, Yingjie, et al. An integrated data management and informatics framework for continuous drug product manufacturing processes: A case study on two pilot plants. *International Journal of Pharmaceutics*. 2023. № 12. P. 123-132. URL: https://www.sciencedirect.com/science/article/abs/pii/S0378517323005069

11. Ferdousi, Rahatara. Digital Twin Disease Diagnosis Using Machine Learning. Diss. Université d'Ottawa/University of Ottawa, 2021.

12. Manogaran, Gunasekaran, Chandu Thota, and Daphne Lopez. Human-computer interaction with big data analytics. *Research Anthology on Big Data Analytics, Architectures, and Applications*. 2022. P. 1578-1596. URL: https://www.igi-global.com/chapter/human-computer-interaction-with-big-data-analytics/291053

13. Mustafee, Navonil, et al. Co-citation analysis of literature in e-science and e-infrastructures. *Concurrency and Computation: Practice and Experience*. 2020. № 32. P. 5620.

14. Dixon, Thomas A., and Isak S. Pretorius. Drawing on the past to shape the future of synthetic yeast research. *International Journal of Molecular Sciences*. 2020. № 21. P. 7156.

15. Hurtado Requena, Sandro Jose. Integration and analysis of biomedical data from multiple clinical trials. URL: https://riuma.uma.es/xmlui/handle/10630/26156

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

17. 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.

18. 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.

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

20. 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.

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

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

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

24. 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

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

26. 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.

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

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

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

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

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

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

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

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

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

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

37. 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

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

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

40. 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

41. 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

42. 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

43. 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

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

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

46. 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.

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

48. 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

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

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

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

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

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

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

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

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

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

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

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

60. 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-Cloud деплойменту на Kubernetes кластерах хмарних провайдерів».

```
import kfp
from kfp import dsl
from kubernetes import client as k8s_client
from kfp.aws import use_aws_secret
from kubernetes.client.models import V1EnvVar


def init_mlflow_experiment_op(
        EXPERIMENT_NAME: str,
        RUN_NAME: str,
        MLFLOW_DB_URL: str,
        MLFLOW_BUCKET: str = 'receptor-mlflow-artifacts',
        vcpu: str = '4'
):
    op = dsl.ContainerOp(name='init_mlflow_op',

image='amazonaws.com/pipelines/general_blocks/init_mlflow_experiment
:latest',
                         container_kwargs={'env': [
                             V1EnvVar('MLFLOW_DB_URL',
MLFLOW_DB_URL),
                             V1EnvVar('MLFLOW_BUCKET',
MLFLOW_BUCKET),
                             V1EnvVar('EXPERIMENT_NAME',
EXPERIMENT_NAME),
                             V1EnvVar('RUN_NAME', RUN_NAME),
                         ]},
                         ).apply(use_aws_secret('aws-secret'))
```

```
    op.container.set_image_pull_policy('Always')
    op.set_cpu_request(vcpu)
    op.set_retry(num_retries=3)

    return op


def generate_descriptors_op(
        EXPERIMENT_NAME: str,
        RUN_NAME: str,
        DATASETS_BUCKET: str,
        DATASET_FOLDER: str,
        DATASET_FILE: str,
        DESCRIPTORS_FILE: str,
        DESCRIPTORS_TYPES: str,
        MLFLOW_DB_URL: str,
        vcpu: str = '4'
):
    op = dsl.ContainerOp(name='generate_descriptors',

image='amazonaws.com/pipelines/automl/generate_descriptors:latest',
                         container_kwargs={'env': [
                             V1EnvVar('EXPERIMENT_NAME',
EXPERIMENT_NAME),
                             V1EnvVar('RUN_NAME', RUN_NAME),
                             V1EnvVar('DATASETS_BUCKET',
DATASETS_BUCKET),
                             V1EnvVar('DATASET_FOLDER',
DATASET_FOLDER),
                             V1EnvVar('DATASET_FILE', DATASET_FILE),
                             V1EnvVar('DESCRIPTORS_FILE',
DESCRIPTORS_FILE),
                             V1EnvVar('DESCRIPTORS_TYPES',
DESCRIPTORS_TYPES),
```

```
                            V1EnvVar('MLFLOW_DB_URL',
MLFLOW_DB_URL)
                        ]},
                        ).apply(use_aws_secret('aws-secret'))

    op.container.set_image_pull_policy('Always')
    op.set_cpu_request(vcpu)
    op.set_retry(num_retries=3)

    return op


def select_descriptors_op(
        EXPERIMENT_NAME: str,
        RUN_NAME: str,
        DATASETS_BUCKET: str,
        DATASET_FOLDER: str,
        DESCRIPTORS_FILE: str,
        TARGET_FILE: str,
        SELECTED_DESCRIPTORS_FILE: str,
        MLFLOW_DB_URL: str,
        TEST_SIZE: str,
        vcpu: str = '4'
):
    op = dsl.ContainerOp(name='select_descriptors',

image='amazonaws.com/pipelines/automl/select_descriptors:latest',
                        container_kwargs={'env': [
                            V1EnvVar('EXPERIMENT_NAME',
EXPERIMENT_NAME),
                            V1EnvVar('RUN_NAME', RUN_NAME),
                            V1EnvVar('DATASETS_BUCKET',
DATASETS_BUCKET),
                            V1EnvVar('DATASET_FOLDER',
DATASET_FOLDER),
```

```
                                V1EnvVar('DESCRIPTORS_FILE',
DESCRIPTORS_FILE),
                                V1EnvVar('TARGET_FILE', TARGET_FILE),
                                V1EnvVar('SELECTED_DESCRIPTORS_FILE',
SELECTED_DESCRIPTORS_FILE),
                                V1EnvVar('MLFLOW_DB_URL',
MLFLOW_DB_URL),
                                V1EnvVar('TEST_SIZE', TEST_SIZE)
                            ]},
                            ).apply(use_aws_secret('aws-secret'))

    op.container.set_image_pull_policy('Always')
    op.set_cpu_request(vcpu)
    op.set_retry(num_retries=3)

    return op


def automl_op(
        i: int,
        EXPERIMENT_NAME: str,
        RUN_NAME: str,
        DATASETS_BUCKET: str,
        DATASET_FOLDER: str,
        DESCRIPTORS_FILE: str,
        SELECTED_DESCRIPTORS_FILE: str,
        TARGET_FILE: str,
        PROBLEM_TYPE: str,
        MODEL_FAMILIES: '[]',
        OPTIMIZATION_METRIC: str,
        SCORE_THRESHOLD: str,
        TIMEOUT_BASE: str,
        LEVEL: str,
        OPTUNA_DB_URL: str,
        TEST_SIZE: str,
```

```
        vcpu: str = '4'
):
    op = dsl.ContainerOp(name=f'automl_base_{i}',

image='amazonaws.com/pipelines/automl/automl:latest',
                         container_kwargs={'env': [
                             V1EnvVar('EXPERIMENT_NAME',
EXPERIMENT_NAME),
                             V1EnvVar('RUN_NAME', RUN_NAME),
                             V1EnvVar('DATASETS_BUCKET',
DATASETS_BUCKET),
                             V1EnvVar('DATASET_FOLDER',
DATASET_FOLDER),
                             V1EnvVar('DESCRIPTORS_FILE',
DESCRIPTORS_FILE),
                             V1EnvVar('SELECTED_DESCRIPTORS_FILE',
SELECTED_DESCRIPTORS_FILE),
                             V1EnvVar('TARGET_FILE', TARGET_FILE),
                             V1EnvVar('PROBLEM_TYPE', PROBLEM_TYPE),
                             V1EnvVar('MODEL_FAMILIES',
MODEL_FAMILIES),
                             V1EnvVar('OPTIMIZATION_METRIC',
OPTIMIZATION_METRIC),
                             V1EnvVar('SCORE_THRESHOLD',
SCORE_THRESHOLD),
                             V1EnvVar('TIMEOUT_BASE', TIMEOUT_BASE),
                             V1EnvVar('TEST_SIZE', TEST_SIZE),
                             V1EnvVar('LEVEL', LEVEL),
                             V1EnvVar('OPTUNA_DB_URL',
OPTUNA_DB_URL),
                             V1EnvVar('TIMEOUT_META', '0'),
                         ]},
                         ).apply(use_aws_secret('aws-secret'))

    op.container.set_image_pull_policy('Always')
```

```
    op.set_cpu_request(vcpu)
    op.set_retry(num_retries=3)

    return op


def log_automl_to_mlflow_op(
        EXPERIMENT_NAME: str,
        RUN_NAME: str,
        MLFLOW_BUCKET: str,
        DATASETS_BUCKET: str,
        DATASET_FOLDER: str,
        DESCRIPTORS_FILE: str,
        SELECTED_DESCRIPTORS_FILE: str,
        TARGET_FILE: str,
        PROBLEM_TYPE: str,
        MODEL_FAMILIES: str,
        OPTIMIZATION_METRIC: str,
        SCORE_THRESHOLD: str,
        OPTUNA_DB_URL: str,
        MLFLOW_DB_URL: str,
        TEST_SIZE: str,
        vcpu: str = '4'
):
    op = dsl.ContainerOp(name='log_automl_to_mlflow',

image='amazonaws.com/pipelines/automl/log_automl_to_mlflow:latest',
                         container_kwargs={'env': [
                             V1EnvVar('EXPERIMENT_NAME',
EXPERIMENT_NAME),
                             V1EnvVar('RUN_NAME', RUN_NAME),
                             V1EnvVar('MLFLOW_BUCKET',
MLFLOW_BUCKET),
                             V1EnvVar('DATASETS_BUCKET',
DATASETS_BUCKET),
```

```
                                V1EnvVar('DATASET_FOLDER',
DATASET_FOLDER),
                                V1EnvVar('DESCRIPTORS_FILE',
DESCRIPTORS_FILE),
                                V1EnvVar('SELECTED_DESCRIPTORS_FILE',
SELECTED_DESCRIPTORS_FILE),
                                V1EnvVar('TARGET_FILE', TARGET_FILE),
                                V1EnvVar('PROBLEM_TYPE', PROBLEM_TYPE),
                                V1EnvVar('MODEL_FAMILIES',
MODEL_FAMILIES),
                                V1EnvVar('OPTIMIZATION_METRIC',
OPTIMIZATION_METRIC),
                                V1EnvVar('SCORE_THRESHOLD',
SCORE_THRESHOLD),
                                V1EnvVar('OPTUNA_DB_URL',
OPTUNA_DB_URL),
                                V1EnvVar('MLFLOW_DB_URL',
MLFLOW_DB_URL),
                                V1EnvVar('TEST_SIZE', TEST_SIZE)
                            ]},
                            ).apply(use_aws_secret('aws-secret'))

    op.container.set_image_pull_policy('Always')
    op.set_cpu_request(vcpu)
    op.set_retry(num_retries=3)

    return op


@dsl.pipeline(
  name='AutoML-pipeline-base-5',
  description='AutoML Pipeline with only base level'
)
def AutoML_pipeline(
        MLFLOW_BUCKET: str,
```

```
        DATASETS_BUCKET: str,
        EXPERIMENT_NAME: str,
        RUN_NAME: str,
        DATASET_FOLDER: str,
        DATASET_FILE: str,
        DESCRIPTORS_FILE: str,
        DESCRIPTORS_TYPES: str,
        TARGET_FILE: str,
        SELECTED_DESCRIPTORS_FILE: str,
        PROBLEM_TYPE: str,
        MODEL_FAMILIES: str,
        OPTIMIZATION_METRIC: str,
        SCORE_THRESHOLD: str,
        TIMEOUT_BASE: str,
        TEST_SIZE: str,
        MLFLOW_DB_URL: str,
        OPTUNA_DB_URL: str,
):
    """A pipeline function describing the orchestration of the
workflow."""
    N_PARALLEL = 5
    vcpu: str = '15'

    init_mlflow_experiment = init_mlflow_experiment_op(
        EXPERIMENT_NAME=EXPERIMENT_NAME,
        RUN_NAME=RUN_NAME,
        MLFLOW_DB_URL=MLFLOW_DB_URL,
        MLFLOW_BUCKET=MLFLOW_BUCKET,
        vcpu=vcpu
    )
    generate_descriptors = generate_descriptors_op(
        EXPERIMENT_NAME=EXPERIMENT_NAME,
        RUN_NAME=RUN_NAME,
        DATASETS_BUCKET=DATASETS_BUCKET,
        DATASET_FOLDER=DATASET_FOLDER,
```

```
        DATASET_FILE=DATASET_FILE,
        DESCRIPTORS_FILE=DESCRIPTORS_FILE,
        DESCRIPTORS_TYPES=DESCRIPTORS_TYPES,
        MLFLOW_DB_URL=MLFLOW_DB_URL,
        vcpu=vcpu
    )
    select_descriptors = select_descriptors_op(
        EXPERIMENT_NAME=EXPERIMENT_NAME,
        RUN_NAME=RUN_NAME,
        DATASETS_BUCKET=DATASETS_BUCKET,
        DATASET_FOLDER=DATASET_FOLDER,
        DESCRIPTORS_FILE=DESCRIPTORS_FILE,
        TARGET_FILE=TARGET_FILE,
        SELECTED_DESCRIPTORS_FILE=SELECTED_DESCRIPTORS_FILE,
        TEST_SIZE=TEST_SIZE,
        MLFLOW_DB_URL=MLFLOW_DB_URL,
        vcpu=vcpu
    )
    automl_blocks = [automl_op(i=i,
                               EXPERIMENT_NAME=EXPERIMENT_NAME,
                               RUN_NAME=RUN_NAME,
                               DATASETS_BUCKET=DATASETS_BUCKET,
                               DATASET_FOLDER=DATASET_FOLDER,
                               DESCRIPTORS_FILE=DESCRIPTORS_FILE,

SELECTED_DESCRIPTORS_FILE=SELECTED_DESCRIPTORS_FILE,
                               TARGET_FILE=TARGET_FILE,
                               PROBLEM_TYPE=PROBLEM_TYPE,
                               MODEL_FAMILIES=MODEL_FAMILIES,

OPTIMIZATION_METRIC=OPTIMIZATION_METRIC,
                               SCORE_THRESHOLD=SCORE_THRESHOLD,
                               TIMEOUT_BASE=TIMEOUT_BASE,
                               TEST_SIZE=TEST_SIZE,
                               LEVEL='base',
```

```
                                    OPTUNA_DB_URL=OPTUNA_DB_URL,
                                    vcpu=vcpu)
                        for i in range(N_PARALLEL)]
    log_automl_to_mlflow =
log_automl_to_mlflow_op(EXPERIMENT_NAME=EXPERIMENT_NAME,

RUN_NAME=RUN_NAME,

MLFLOW_BUCKET=MLFLOW_BUCKET,

DATASETS_BUCKET=DATASETS_BUCKET,

DATASET_FOLDER=DATASET_FOLDER,

DESCRIPTORS_FILE=DESCRIPTORS_FILE,

SELECTED_DESCRIPTORS_FILE=SELECTED_DESCRIPTORS_FILE,

TARGET_FILE=TARGET_FILE,

PROBLEM_TYPE=PROBLEM_TYPE,

MODEL_FAMILIES=MODEL_FAMILIES,

OPTIMIZATION_METRIC=OPTIMIZATION_METRIC,

SCORE_THRESHOLD=SCORE_THRESHOLD,

TEST_SIZE=TEST_SIZE,

OPTUNA_DB_URL=OPTUNA_DB_URL,

MLFLOW_DB_URL=MLFLOW_DB_URL,
                                                    vcpu=vcpu)
```

```
    generate_descriptors.after(init_mlflow_experiment)
    select_descriptors.after(generate_descriptors)
    for automl_block in automl_blocks:
        automl_block.after(select_descriptors)
        log_automl_to_mlflow.after(automl_block)

    # Call set_image_pull_secrets after get_pipeline_conf().
    dsl.get_pipeline_conf() \

.set_image_pull_secrets([k8s_client.V1ObjectReference(name="ecr-
cred")])


kfp.compiler.Compiler().compile(
    pipeline_func=AutoML_pipeline,
    package_path='yaml_files/AutoML_pipeline_base_5.yaml')
```

ДОДАТОК Б
(обов'язковий)

КОПІЯ ПУБЛІКАЦІЇ У ФАХОВОМУ НАУКОВОМУ ВИДАННІ

RSC Advances

ROYAL SOCIETY OF CHEMISTRY

PAPER

View Article Online
View Journal | View Issue

Cite this: *RSC Adv.*, 2024, **14**, 1341

Augmenting a training dataset of the generative diffusion model for molecular docking with artificial binding pockets†

Taras Voitsitskyi, *[ad] Volodymyr Bdzhola, [b] Roman Stratiichuk,[ae] Ihor Koleiev,[ad] Zakhar Ostrovsky,[a] Volodymyr Vozniak,[a] Ivan Khropachov,[a] Pavlo Henitsoi,[a] Leonid Popryho,[a] Roman Zhytar,[a] Semen Yesylevskyy, [acdf] Alan Nafiiev[a] and Serhii Starosyla [e]

This study introduces the PocketCFDM generative diffusion model, aimed at improving the prediction of small molecule poses in the protein binding pockets. The model utilizes a novel data augmentation technique, involving the creation of numerous artificial binding pockets that mimic the statistical patterns of non-bond interactions found in actual protein–ligand complexes. An algorithmic method was developed to assess and replicate these interaction patterns in the artificial binding pockets built around small molecule conformers. It is shown that the integration of artificial binding pockets into the training process significantly enhanced the model's performance. Notably, PocketCFDM surpassed DiffDock in terms of non-bond interaction and steric clash numbers, and the inference speed. Future developments and optimizations of the model are discussed. The inference code and final model weights of PocketCFDM are accessible publicly *via* the GitHub repository: https://github.com/vtarasv/pocket-cfdm.git.

Received 28th November 2023
Accepted 21st December 2023

DOI: 10.1039/d3ra08147h

rsc.li/rsc-advances

Introduction

Molecular docking plays a central role in modern computational drug discovery. Until recently docking was the only available method of predicting the poses of small molecules in the binding pockets of target proteins fast enough to be useful in large-throughput virtual screening projects. Despite its ultimate importance, there is a notable stagnation in improving the docking versatility, accuracy, computing cost, and predictive power.[1,2] Being based on inevitably simplified empirical potentials of intermolecular interactions and lacking explicit solvent, the docking scoring functions have arguably reached a plateau of practical accuracy. Despite a number of recent developments in the field, all of them are incremental improvements and domain-specific tuning rather than technological breakthroughs.

However, recent advancements in deep learning methodologies for predicting ligand poses in the protein binding pockets provide a promising alternative to docking algorithms. These techniques can be categorized into two primary groups.

The models from the first group utilize a one-shot inference regression-based approach.[3,4] They are developed with the aim of very fast inference, which is superior to traditional docking simulations. The geometric vector perceptrons (GVP) and equivariant graph neural networks (EGNN) are the most popular architectures for those types of models.[5–7] Such techniques as EquiBind[4] and TANKBind[3] demonstrated efficacy in predicting protein–ligand binding structure without prior knowledge about the binding pocket (also known as blind docking) while being faster than traditional docking techniques by several orders of magnitude. However, they still suffer from unrealistic ligand conformations and numerous sterical clashes in predicted complexes, which puts them behind the docking approaches in terms of structure quality and reliability. A possible reason for this is a mismatch between the objectives of molecular docking and the regression paradigm. Particularly, the accuracy metrics in molecular docking are based on structural similarity, rather than a regression loss.

The second and currently state-of-the-art approach uses generative AI models, which aim to be on par or better than

[a]*Receptor.AI Inc., 20-22 Wenlock Road, London N1 7GU, UK*
[b]*Institute of Molecular Biology and Genetics of The National Academy of Sciences of Ukraine, 150 Zabolotnogo Str., Kyiv 03143, Ukraine*
[c]*Institute of Organic Chemistry and Biochemistry, Czech Academy of Sciences, Prague 6 CZ-166 10, Czech Republic*
[d]*Department of Physics of Biological Systems, Institute of Physics of The National Academy of Sciences of Ukraine, 46 Nauky Ave., Kyiv 03038, Ukraine*
[e]*Department of Biophysics and Medical Informatics, Educational and Scientific Centre "Institute of Biology and Medicine", Taras Shevchenko Kyiv National University, 64 Volodymyrska Str., Kyiv 01601, Ukraine*
[f]*Department of Physical Chemistry, Faculty of Science, Palacký University Olomouc, 17 listopadu 12, Olomouc 771 46, Czech Republic*

† Electronic supplementary information (ESI) available. See DOI: https://doi.org/10.1039/d3ra08147h

classic docking techniques in terms of accuracy and structure quality. The DiffDock,[8] a current leader in the field, demonstrates superior performance in comparison to some conventional docking techniques and the previous ML models in the blind docking scenarios. It produces much less steric clashes than its rivals and generates realistic ligand conformations. However, the enhanced precision of DiffDock comes at the cost of a substantial computational burden, which is on par with that of traditional docking methods. Since there is a significant potential for improvement in terms of inference speed, it makes generative models the most promising in the field at the moment of writing.

The major bottleneck, which hampers further improvement of the generative ligand pose prediction models, is the inherently limited amount of the training data. The overall number of experimentally determined protein–ligand complexes resolved by X-ray, Cryo-EM, or NMR techniques is now below 20 000. The PDBbind database,[9,10] which is a primary dataset for machine learning in protein binding site prediction[11,12] and ligand pose prediction,[3,4,8] contains 19 443 distinct protein–ligand complexes with the binding activity annotations (version 2020). Out of this, 2709 entries involve peptide ligands or multiple molecules in a single binding pocket; 3827 have an insufficient resolution (2.5 angstroms and more); 3523 contain weak binders ($K_d/K_i/IC_{50}$ > 10 μM) and 216 lack confident activity measurements. Thus there are only 10 270 high-quality complexes, which could be used for model training

Another issue of the experimental complexes is ligand data sparsity. There are 12 815 distinct small molecule ligands in PDBbind, of which only 1655 appear in two or more entries (Fig. 1). It indicates that ML-based approaches are mostly presented with a particular ligand bound to a single protein without any information about the possible variability of its binding modes. This might lead to overfitting to a single ligand pose, which wouldn't be the case when working with more dense data as was demonstrated for the affinity prediction models.[13]

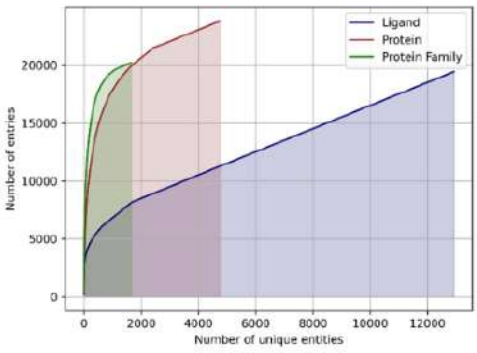


Fig. 1 The cumulative sums of entries in the PDBbind database assigned to a unique ligand, unique protein, or unique protein family. Note that the count of proteins and families may exceed the total number of complexes in PDBbind because a single PDB structure may be assigned to multiple entries.

In addition, the proteins are represented very unevenly in available data. The protein data bank (PDB) identifiers of 19 443 protein–ligand complexes are associated with 4749 unique UniProt[14] identifiers, but 1.500 most frequently occurring identifiers accounting for ~80% of the total number of complexes (Fig. 1). The same is true for the protein families: a total of 1646 distinct protein families and superfamilies are present but the top 100 families account for almost 60% of complexes (Fig. 1). Thus, despite a significant overall variety of proteins, there is an obvious overrepresentation of some proteins and protein families, which may lead to model overfitting and imbalance.

There is little doubt that an insufficient overall number of samples, limited ligand diversity, and protein representation imbalance in the available data for model training are impairing the accuracy of ML-based approaches to the ligand pose prediction. These limitations are especially noticeable when comparing the amount and quality of the training data with a requirement for the model to operate on arbitrary protein targets and arbitrary ligands from an immense chemical space.[15]

It is clear that the dataset of experimentally resolved structures will not grow fast enough to satisfy the demands of the exploding field of ML ligand–protein binding prediction thus other approaches are needed for overcoming the lack of training data.

In this study, we develop an approach of augmenting the training set of the protein–ligand complexes with artificial data, which mimics real protein binding pockets in a number of structural characteristics. The statistical distributions of artificial pockets' parameters are fitted to the respective distributions of real protein–ligand structures so that both types of data could be used together seamlessly.

The idea of our approach is based on the assumption that the number of favorable interaction geometries between the protein amino acids and the chemical groups of the ligands is finite and is represented sufficiently well in the available experimental data. What is presumably lacking, is the sampling of all possible combinations of such interactions within the binding pocket. In other words, we assume that available experimental structures provide decent statistics about the preferable chemical identity of interacting atom pairs, distances between the atoms, and orientations of the corresponding chemical groups. However, only a very small fraction of all possible combinations of such pairs is observed in real proteins.

If one generates a large number of "artificial binding pockets", which follow the same statistical distribution of the interacting atom pairs as the real ones, but sample a much larger variety of their combinations, it might be possible to overcome the undersampling and to train the model on a more complete set of data.

Although we do not have a strong independent proof of our hypothesis, we decided to validate it experimentally by developing an algorithm for generating artificial pockets, compiling a dataset consisting of artificial and real pockets, and measuring the performance of the diffusion generative model,

which is inspired by DiffDock, and is trained on such augmented data – PocketCFDM (Pocket Conformation Fitting Diffusion Model). We show that PocketCFDM outperforms DiffDock, which is a recent breakthrough technology in the field of ML-based docking, in terms of generated ligand poses correctness (less steric clashes and more favorable non-covalent interactions). We also discuss future prospects of our methodology in terms of improving its predictive power and the speed of inference.

Methods

Protein and ligand preprocessing

The Python API of the RDKit v. 2021.03 was utilized for loading, processing, and feature generation of small molecules. A custom protein processing module was developed to extract protein data from the PDB files and to generate the necessary features for model training. This module utilizes the PDB atom names to obtain atom-level graph features, rather than relying on a third-party software to infer them. This approach decreases the exclusion rate for processed proteins due to inevitable inconsistencies in the PDB files.

In order to assess the protein–ligand interactions, we employed the SMILES arbitrary target specification (SMARTS) substructure search to classify the ligand atoms or chemical groups into the following categories: hydrophobic, aromatic, amide, donor, acceptor, cation, anion, or halogen. The protein atom assignment was conducted using a predefined mapping of the standard PDB atom names (Table S1†).

Prior to the assessment of non-covalent interactions, proteins and ligands were protonated (including any implicit hydrogens). The protein protonation was performed in a similar manner to EquiBind and DiffDock by using the reduce software in order to account for hydrogen bonding correctly. Additionally, we considered possible alternative positions of hydrogens, such as within hydroxyl or amine groups. In this work, we considered only amino acids as the entities interacting with the ligands, while the water molecules, ions and metal atoms, which are present in the experimental structures, are disregarded. This limitation could be addressed in the next versions of our technique as detailed in the Discussion. The source code of the preprocessing module is available: **https://github.com/vtarasv/rai-chem.git**.

The choice of non-bond interactions

The hydrophobic and electrostatic interactions as well as the hydrogen bonding (favorable non-bond interactions) were assessed between the protein and the ligand. We also accounted for unfavorable interactions, such as donor–donor atom pairs in close proximity, to improve the overall quality of artificial binding pockets by omitting such interactions. The summary of all used non-bond interactions is shown in Table S4.† The choice of included interactions is based on a compromise between the multiple approaches in the literature,[16–19] commonly used cheminformatics software[20–22] and widely-used molecular modeling tool BIOVIA Discovery Studio Visualizer.

Ligand–protein interaction statistics

The protein–ligand complexes with known 3D structures were taken from the PDBbind dataset.[9,10] Only the entries that satisfy the following criteria were used: the presence of a single small molecule ligand, resolution below 2.5 Å, and an activity/affinity less than 10 μM. A total of 10 270 protein–ligand complexes were selected. Among them 1805 ligand files were found to be unreadable, resulting in a final count of 8465 complexes that were used in this work.

The following statistical information was extracted:

- The probability of a specific ligand substructure (particular atom type, aromatic ring, or amide group) to participate in a protein–ligand interaction.
- The distribution of the number of the binding pocket substructures that are connected to ligand substructures through a specific interaction type.
- The distribution of amino acids involved in particular interaction types.
- The distributions of distances and angles involved in particular interaction types.

Artificial pockets generation

We utilized the PeptideBuilder package[23] to produce a collection of 20 amino acids in the PDB format, as well as 400 dipeptides representing each possible permutation of two amino acids. The amino acids and dipeptides were flanked by GLY residues on both sides and served as the basic building blocks for artificial pockets. The inclusion of flanking GLY residues helps in the generation of the correct peptide conformers, which are restrained by the peptide bonds on each side of the building blocks.

Artificial pocket generation is an iterative process of placing the building blocks around a small molecule in order to form a realistic network of non-covalent interactions. A total of 13 potential interactions, both directed and undirected, were taken into account during the pocket construction (Table 1).

Table 1 Non-bonded interactions taken into account during artificial pocket construction

#	Pocket feature	Ligand feature	Interaction type
1	Aromatic ring	Aromatic ring	Pi stacking
2	Amide group	Aromatic ring	Amide–pi
3	Aromatic ring	Amide group	Amide–pi
4	Aromatic ring	Cationic atom	Cation–pi
5	Hydrogen bond donor	Hydrogen bond acceptor	Hydrogen bond
6	Hydrogen bond acceptor	Hydrogen bond donor	Hydrogen bond
7	Hydrogen bond acceptor	Halogen atom	Halogen bond
8	Cationic atom	Anionic atom	Electrostatic
9	Anionic atom	Cationic atom	Electrostatic
10	Cationic atom	Aromatic ring	Cation–pi
11	C or S atom	F atom	Hydrophobic
12	C or S atom	Cl, Br or I atom	Hydrophobic
13	C or S atom	C or S atom	Hydrophobic

The pocket construction starts from the particular small molecule conformer (referred as ligand hereafter). For each of the 13 interactions listed in Table 1, the following steps are performed:

1. Find all the features of ligand L, which are compatible with the current interaction type i. Each such feature is denoted as F_{Li}.

2. Given experimental probability P of finding F_{Li} among all interactions of type i and the random number p, determine whether the new interaction should be added if $p < P$.

3. Select the peptide building block B to be placed by taking all building blocks with the features compatible with i and randomly selecting one of them according to the experimentally determined probability of the corresponding residue to participate in the interaction i.

4. For the chosen building block B, generate a random conformer taking into account the peptide bonds to flanking GLY residues. Then delete the flanking GLY residues.

5. Randomly sample the distance d between the F_{Li} and the matching feature of B from experimentally determined distributions and place the building block at a determined location.

6. Repeat the following steps until no steric clashes or unfavorable interactions are found between the building block and the ligand and between the building blocks:

a. Randomly rotate and translate the building block B preserving the distance d.

b. Determine whether the angular criteria of interaction i are met, if applicable.

c. If the maximal number of tries (2000 by default) is reached, the building block is skipped.

Using this algorithm 8465 artificial pockets were generated (one for each ligand from the PDBbind database). The distributions of the non-bond interactions of the generated pockets were computed and compared to the experimental distributions. Due to a good match of the distributions, no further tuning of the algorithm was required.

The examples of randomly selected artificial pockets can be found in the ESI (Fig. S1†).

Model training and testing datasets

In order to cover the maximal diversity of the ligands we employed the ZINC20 database of commercially available chemicals widely used for virtual screening.[24,25] The "In-Stock" category of chemicals was chosen, resulting in a collection of 13 million molecules represented by SMILES. The compounds were standardized using the ChEMBL Structure Pipeline[26] (https://github.com/chembl/ChEMBL_Structure_Pipeline). Particularly, we eliminated duplicates, compounds with less than seven heavy atoms, and large molecules with a molecular weight exceeding 750 g mol^{-1}. This resulted in 9 041 707 compounds. A subset of compounds (size depends on the model training settings) was randomly selected and used for artificial pockets generation. The pocket generation was repeated for each model training epoch.

In addition to the artificial pockets, the real binding pockets from the PDBbind database were added to the training set. These were defined as the residues with at least one heavy atom within a 5 Å from any heavy atom of the ligand. We compared the model training results achieved with the artificial pockets only, experimental pockets only, and a combination of both.

A subset of PDBbind complexes, which had been previously used in the analysis of DiffDock technique,[8] was used for model testing. We additionally removed all complexes containing the ligands with more than 100 heavy atoms in order to concentrate on the drug-like molecules.

Model performance metrics

In a manner consistent with the EquiBind,[4] TANKBind,[3] and DiffDock[8] we used 25th, 50th, and 75th percentiles of symmetry-corrected[27] root mean square deviation (RMSD) between expected and predicted ligand pose, together with the percentage of predictions below 5 Å or 2 Å RMSD. The centroid distances between the expected and predicted positions of the ligands were tracked using the same metrics.

The following additional metrics of the non-bond interactions were used:

- The fraction of favorable contacts (F_{fav}) – total number of favorable interactions normalised by the number of ligand heavy atoms. The contribution of the hydrophobic interactions was accounted for with a weight of 1/6 due to their high relative abundance.
- The fraction of atoms involved in favorable contacts ($F_{fav\text{-}atoms}$) – a fraction of the heavy ligand atoms participating in at least one non-bond interaction.
- The fraction of unfavorable contacts (F_{unfav}) – total number of unfavorable interactions normalised by the number of ligand heavy atoms.

In order to evaluate the quality of predicted ligand poses, we determined the frequency of steric clashes between the ligand and protein atoms. The clash was defined as the distance between the atoms smaller than 70% of the sum of their respective van der Waals radii.

Model training and inference

The core architecture utilized in this study was the diffusion generative score model, which was adapted from DiffDock. This model is based on SE(3)-equivariant convolutional networks that operate on point clouds.[28,29] The model utilizes pocket and ligand graph representations and takes into account the spatial arrangement of the atoms. It produces SE(3)-equivariant vectors that describe the ligand's translations and rotations, along with SE(3)-invariant scalars per each compound's rotatable bond. The input position of a ligand is subsequently altered based on the model's output resulting in the final conformation of the molecule within a binding site. The torsion angles of the ligand are optimized as well as translations and rotations of the whole molecule. During the training phase, the position of each input ligand in a complex undergoes modifications caused by translational, rotational, and torsional noise, which can be referred to as forward diffusion. The model then learns how to reverse the diffusion process. This method enables the generation of numerous alternative ligand poses during the inference process.[8]

We adjust the DiffDock model inputs and architecture as follows:

- The atomic-level pocket graph is used instead of the residue-level graph of the whole protein.
- The categorical feature space of the ligand nodes was decreased significantly by narrowing down the number of atom types, number of neighbor heavy atoms, and atomic charges to those expected in the typical screening databases of small molecules.

The learning rate was set to 0.0125 based on initial tuning. During each epoch, the model is trained for 10 000 iterations with a batch size of 4. In our experimental conditions, the batch always consists of identical pocket and ligand components, whereas the level of diffusion noise varies between individual samples.

The models were trained for 25 epochs. During the training based on artificial data, 10 000 ligands are randomly sampled from the preprocessed ZINC dataset at each epoch. After that, the artificial pockets are created for each ligand. Thus, 250 000 unique artificial complexes were generated for the model training. Each epoch of training on experimental protein–ligand complexes is performed with 10 000 randomly sampled PDBbind entries (out of 16 379 complexes in the train split). The training on a combined dataset was performed with a 4 : 1 ratio of artificial and experimental data with 200 000 unique artificial complexes. The final production model was trained for 80 epochs using a combined dataset with 640 000 unique artificial complexes. The model training, which is a GPU-intensive task, and pocket generation, which is a CPU-intensive task, were separated into distinct parallel workflows.

Given the generative nature of the model, it is possible to produce an infinite number of alternate ligand poses. The real-world model performance is thus sensitive to the scoring function, which is used for the pose ranking and selection. The developers of DiffDock employed a confidence model that takes into account all protein atoms and produces a confidence score of the ligand pose. In contrast, we employed a non-covalent interaction score in the binding pocket, which reduces the inference cost significantly. Our scoring function S is computed as follows:

$$S = F_{\text{fav}} + F_{\text{fav-atoms}} - 2F_{\text{unfav}} - 2F_{\text{unfav-atoms}} - 10D_{\text{clash}}$$

where $F_{\text{unfav-atoms}}$ is a fraction of the ligand heavy atoms participating in at least one unfavorable interaction, D_{clash} is a sum of all distances, which are below the steric clashes threshold. The coefficients of the scoring function were adjusted empirically by visual inspection of the predicted protein–ligand complexes.

Results and discussion

Statistics of non-bond interactions in experimental and artificial protein–ligand complexes

Analysis of the high-quality PDBbind protein–ligand dataset revealed a total of 343 784 intermolecular non-covalent interactions. As anticipated, the hydrophobic contacts were the predominant kind of interaction, with a total of 267 774 atom pairs observed. The overwhelming majority of these interactions occurred between hydrophobic carbon or sulfur atoms. The hydrogen bonds were the second most prevalent form of interaction, occurring around once for every five hydrophobic pairs. The remaining contacts constituted less than 3% of the overall count (Fig. 2 and Table S2†).

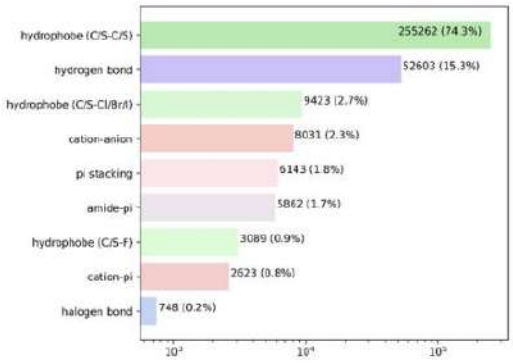


Fig. 2 Probability distribution of the non-bond protein–ligand interactions in the PDBbind complexes. Every bar is labeled with the total number of interactions and its fraction. Note the log scale of the X-axis.

The total number of non-bond interaction entries in the artificial pockets involving the same ligands amounted to 453 593. This significantly larger amount, in comparison to the real ones, is caused by formations of "unintended" interactions (mainly hydrophobic) during the placement of the pocket building blocks in the close proximity to the ligand. The pocket generation algorithm underestimates the solvent exposure contribution to the experimental interaction statistics because the ligand–water interactions are not taken into account during the pocket generation. This is also likely to produce additional interactions with the protein, which partially substitute the ligand–solvent interactions.

We compared the distributions of the parameters for each type of interaction between real and artificial pockets. For each interaction type, we also computed the probability of the involvement of particular amino acids in the binding pocket.

Fig. 3 shows the statistics of hydrophobic interactions in real and artificial binding pockets.

There is a reasonably good correspondence between the distributions, but the distributions of distances are systematically smoother and more monotonous for artificial pockets in comparison to real ones. This is an artifact caused by the frequent formation of "unintended" hydrophobic contacts while generating other types of interactions due to the abundance of hydrophobe atoms in both amino acids and small molecules. The distributions of interactions among amino acids are very similar except the notably increased involvement of PHE and TRP in artificial pockets. This is explained by the overlap between hydrophobic and pi stacking interactions, which is not checked in the algorithm for the sake of computational efficiency.

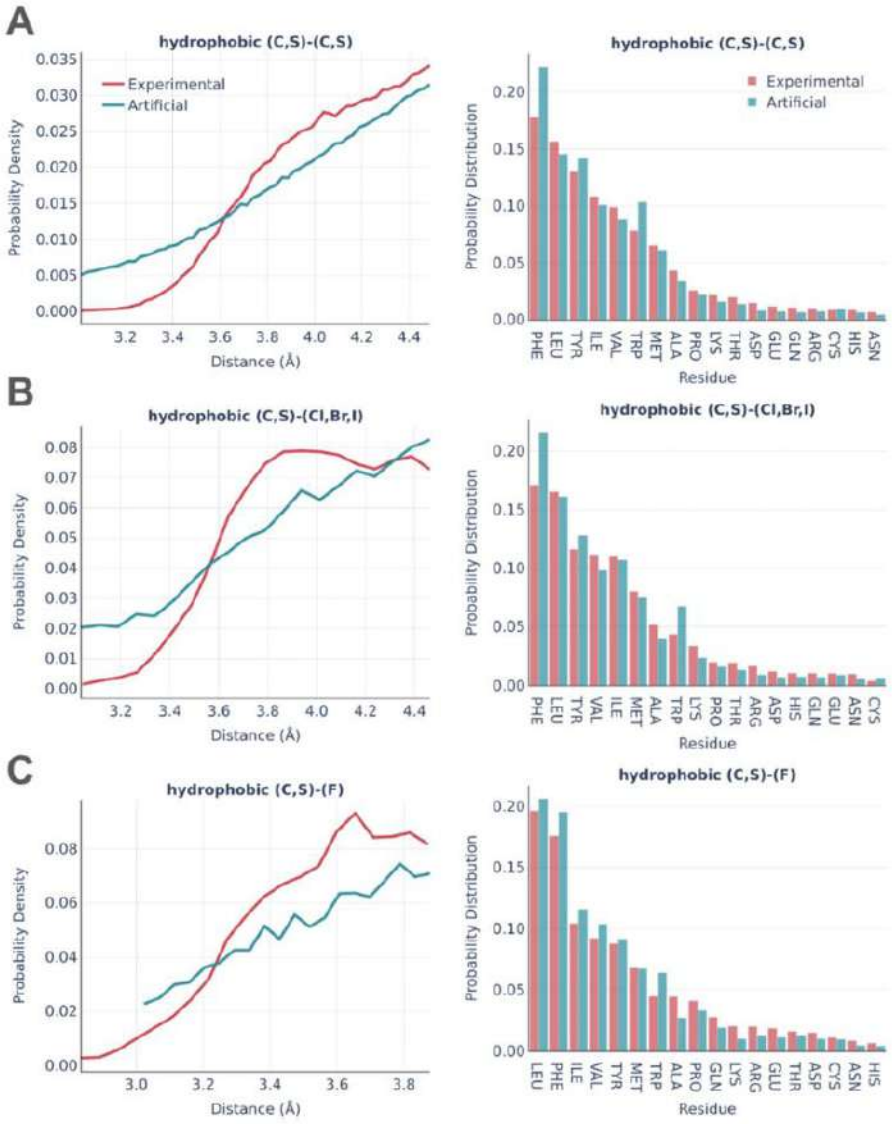


Fig. 3 Statistics of hydrophobic interactions in the PDBbind and artificial pockets involving (A) C or S atoms of the protein and the ligand (B) Br, Cl or I atoms of the ligand (C) F atoms of the ligand. The distance distributions are shown in the left columns and the pocket residues occurrence ratio is shown in the right column.

Fig. 4 shows the statistics of pi stacking interactions. There are two types of these interactions: parallel (true pi stacking) and T-shaped (aromatic–pi interactions). The distance distributions of both types of interactions are remarkably similar in real and artificial pockets as well as the involvements of different aromatic amino acids. However, the theta angle distributions of parallel interactions for artificial pockets are shifted toward 90° by 10–15° in comparison to experimental ones since the generation algorithm only checks if the aromatic ring has an angle within a given range.

Fig. 5 shows the statistics of amide–pi interactions, which could also be classified into parallel and T-shaped.

Similarly to pi stacking interactions, the distance distributions are very similar between real and artificial pockets, while the theta angles in artificial pockets are somewhat shifted towards 90°. The primary residues serving as donors of the amide group were found to be glycine, which exposes the peptide bond due to the lack of a side chain, as well as asparagine and glutamine, which possess an amide group at the terminus of their side chains. The predominant residue bearing

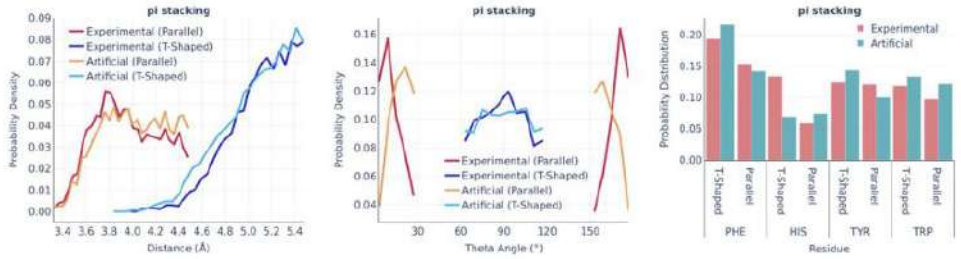


Fig. 4 Pi stacking: the distance distribution (left), theta angle distribution (middle), and pocket residues frequency (right) in the PDBbind and artificial pockets.

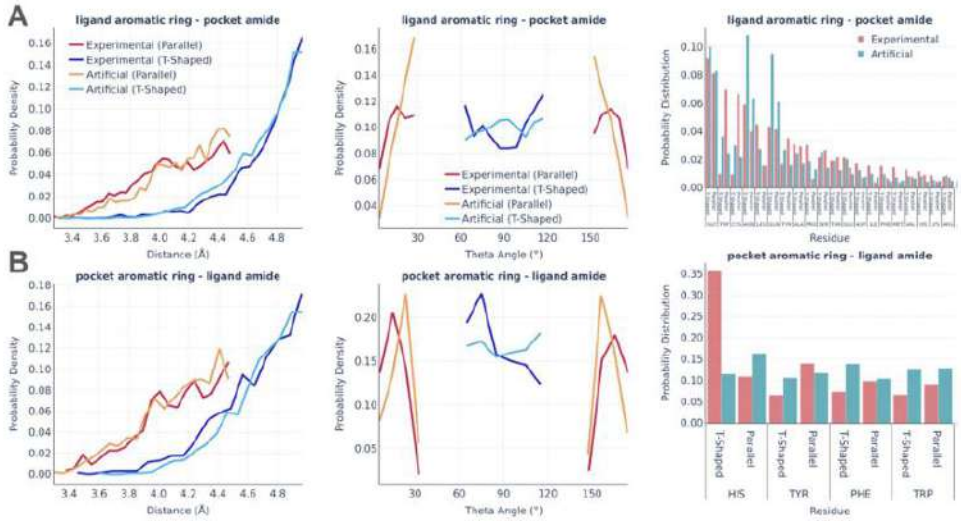


Fig. 5 Amide–pi interactions: the distance distribution (left), theta angle distribution (middle), and pocket residues frequency (right) in the PDBbind and artificial pockets for the ligand aromatic ring – pocket amide contacts (A) and pocket aromatic ring – ligand amide contacts (B).

an aromatic ring in the amide–pi linkages is HIS, which accounts for the majority of experimentally observed T-shaped interactions. There are significantly less HIS contacts in artificial pockets because unfavorable donor–donor contacts between the nitrogens of the HIS ring and amide group were omitted during the pocket generation.

The distributions of the hydrogen bonds are shown in Fig. 6. Hydrogen bonding imposes the biggest challenge in terms of artificial pocket generation. It is clearly seen there are the biggest discrepancies between real and artificial pockets in terms of the hydrogen bonds' parameters, which are especially visible for the angle distributions. In general, our algorithm tends to underestimate the D–H–A angles (the significance of the angle distribution was deliberately reduced to speed up the algorithm) and doesn't capture the distance peak at 2.8–2.9 Å, which is observed in real pockets, while generating significantly more long h-bonds (short h-bonds tend to be discarded more often during the generation as they often lead to steric clashes between the atoms not participating in the interaction). These compromises allow us to keep the algorithm fast enough for routine practical usage. At the same time, the involvement of different amino acids is reproduced remarkably well in artificial pockets.

The halogen bonds are the least frequent type of interaction in real binding pockets. Their distributions are shown in Fig. 7. Taking into account the small number of observed interactions of this kind, the experimental and artificial distributions are sufficiently similar to each other.

The electrostatic interactions were analyzed separately for the "ligand anion – pocket cation" and "pocket anion – ligand cation" pairs (Fig. 8).

The experimentally observed involvement of charged amino acids is reproduced almost ideally in artificial pockets, while the distance distributions are somewhat different in real and artificial pockets. Artificial pockets possess more iterating pairs at larger distances than real ones, which could be explained by the

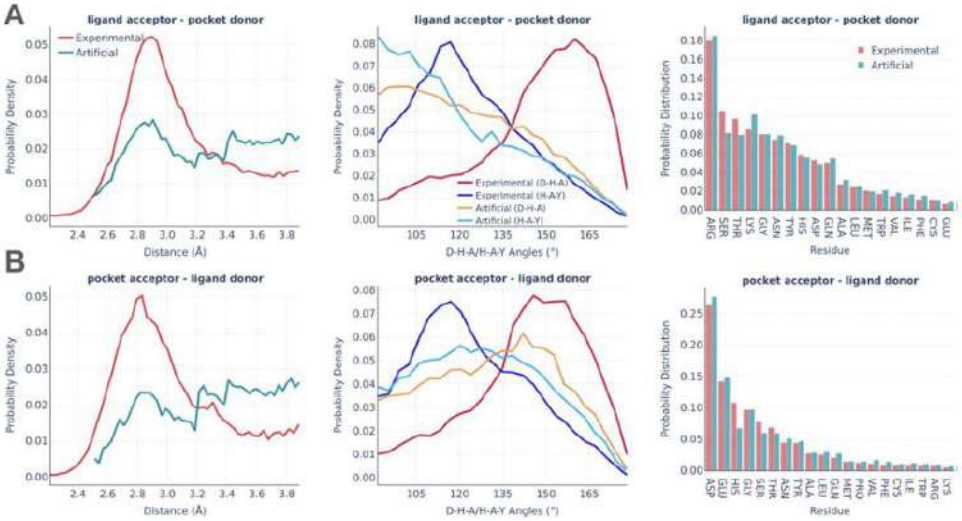


Fig. 6 Hydrogen bonds: the distance distribution (left), D–H–A/H–A–Y angles distribution (middle), and pocket residues frequency (right) in the PDBbind and artificial pockets for the ligand acceptor – pocket donor contacts (A) and pocket acceptor – ligand donor contacts (B). In the D–H–A/H–A–Y angles, D is the donor atom covalently bound to the hydrogen; H is a hydrogen atom; A is an acceptor atom; Y is a heavy covalently bound neighbor of the acceptor atom.

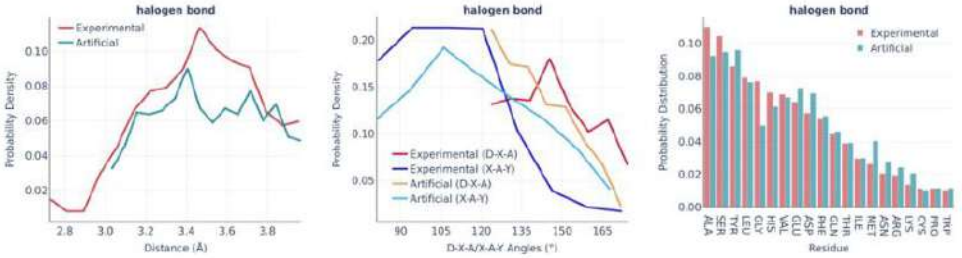


Fig. 7 Halogen bonds: the distance distribution (left), D–X–A/X–A–Y angles distribution (middle), and pocket residue frequencies (right) in the PDBbind and artificial pockets. In the D–X–A/X–A–Y angles, D is the donor atom covalently bound to the halogen; X is a halogen atom; A is the halogen bond acceptor atom; Y is a heavy covalently bound neighbor of the acceptor atom.

increased complexity of fitting pocket residues in the close proximity of a small molecule. Such placement often leads to steric clashes and thus is often discarded by the algorithm.

The last type of interaction is cation–pi pairs (Fig. 9). They are rather minor and the distributions of their parameters are very similar in real and artificial pockets without any significant feature worth commenting on.

Impact of artificial data on model performance

To evaluate the potential influence of data augmentation using artificial protein–ligand complexes on model performance, we compared three models. The first model was exclusively trained on experimental protein–ligand complexes, the second model was trained on the artificially generated pockets with corresponding small molecules, and the third model was trained on the combination of both. The metrics are reported for the top-1 predicted complex and the best of the top-5 predicted complexes based on custom scoring function S (see the Methods section).

Table 2 illustrates that the inclusion of artificial or combined data in the training process led to enhancements in both the RMSD and centroid distance metrics, as compared to the model trained exclusively on experimental complexes. Unexpectedly, the training of the artificial dataset is superior to the combined dataset in certain metrics, particularly when the generation of 40 poses for each pocket was used. We hypothesize that this might be caused by ignoring the solvent exposure contribution in the pocket generation algorithm, which results in an exaggerated number of the protein–ligand interactions and the tighter spatial constraints in the final ligand pose. In addition, we noticed multiple experimental complexes with the steric clashes (*e.g.* 1gj4, 5yr6) or even covalent bonds (*e.g.* 3s3q, 6eyz)

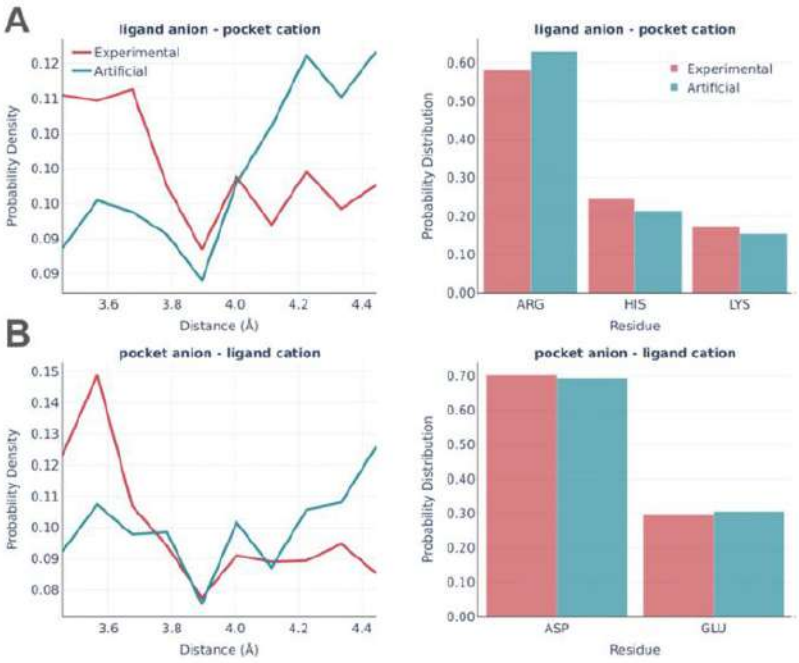


Fig. 8 Electrostatic interactions: the distance distribution (left) and pocket residues frequency (right) in the PDBbind and artificial pockets for the ligand anion – pocket cation (A) and pocket anion – ligand cation (B).

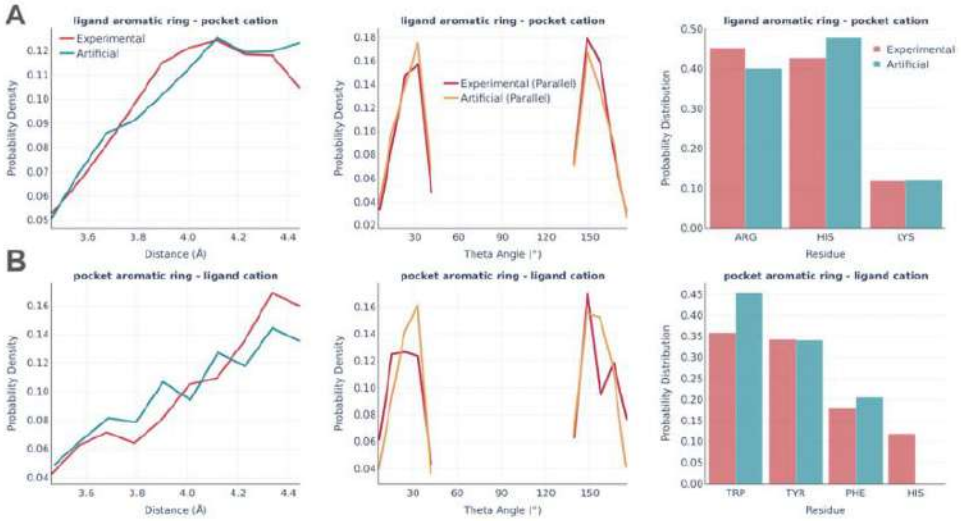


Fig. 9 Cation–pi interactions: the distance distribution (left), theta angle distribution (middle), and pocket residues frequency (right) in the PDBbind and artificial pockets for the ligand aromatic ring – pocket cation (A) and pocket aromatic ring – ligand cation (B).

between the protein and the ligand. Such artifacts constitute nearly 5% of the PDBBind training data, which may contribute to confusing the model concerning valid physical distance constraints.

The combined dataset of protein–ligand complexes exhibited superior performance compared to the other two datasets in terms of both favorable and unfavorable non-covalent interactions, as illustrated in Table 3. Additionally, it was observed that

Table 2 The symmetry-corrected root mean square deviation (RMSD) and centroid distance metrics between predicted and real ligand positions in the PDBbind test complexes. The models generated either 10 or 40 ligand poses. The best of top 5 poses is picked based on the lowest RMSD. ↑ – means the higher the better; ↓ – means the lower the better

	RMSD					Centroid distance				
	Percentiles ↓			% Below threshold ↑		Percentiles ↓			% Below threshold ↑	
Dataset	25th	50th	75th	5 Å	2 Å	25th	50th	75th	5 Å	2 Å
10 samples, top 1										
Experimental	4.08	5.51	7.45	40.93	5.02	1.02	1.75	2.7	94.59	57.92
Artificial	3.69	5.75	7.39	40.15	5.02	1	1.46	2.19	97.68	69.88
Combined	3.76	5.13	7.12	45.95	7.34	1.05	1.69	2.33	98.07	63.32
10 samples, top 5										
Experimental	2.68	3.72	4.64	82.24	11.2	0.95	1.37	1.97	99.61	75.29
Artificial	2.49	3.47	4.6	81.08	16.99	0.78	1.24	1.86	98.84	80.31
Combined	2.4	3.31	4.55	85.33	17.37	0.81	1.29	1.88	99.23	77.99
40 samples, top 1										
Experimental	3.89	5.54	7.36	40.93	4.63	1.01	1.73	2.48	96.14	57.92
Artificial	3.47	5.46	7.25	47.1	7.72	1	1.39	2.09	97.68	72.59
Combined	3.86	5.2	7.16	45.95	6.18	1.05	1.57	2.42	97.68	64.09
40 samples, top 5										
Experimental	2.37	3.35	4.6	82.24	15.06	0.85	1.19	1.69	99.61	81.08
Artificial	1.99	3.08	4.3	85.33	25.1	0.73	1.11	1.69	99.61	82.63
Combined	2.09	3.3	4.6	82.24	23.94	0.8	1.26	1.99	98.84	75.29

Table 3 The favorable rate, favorable rate uniq, and unfavorable rate for the predicted PDBbind test complexes. The models generated either 10 or 40 samples. The best of the top 5 candidates are picked based on the lowest RMSD. ↑ – means the higher the better; ↓ – means the lower the better

	Favorable rate[a]			Favorable rate uniq[b]			Unfavorable rate[c]		
	Percentiles ↑			Percentiles ↑			Percentiles ↓		
Dataset	25th	50th	75th	25th	50th	75th	25th	50th	75th
10 samples, top 1									
Experimental	0.27	0.37	0.51	0.38	0.53	0.67	0	0.08	0.17
Artificial	0.27	0.39	0.53	0.38	0.53	0.67	0	0.07	0.15
Combined	0.27	0.38	0.54	0.41	0.54	0.68	0	0.07	0.15
10 samples, top 5									
Experimental	0.28	0.38	0.52	0.35	0.53	0.66	0.04	0.1	0.23
Artificial	0.3	0.39	0.52	0.38	0.52	0.68	0.04	0.1	0.21
Combined	0.3	0.4	0.56	0.39	0.55	0.67	0.03	0.09	0.18
40 samples, top 1									
Experimental	0.29	0.37	0.54	0.4	0.54	0.71	0	0.06	0.14
Artificial	0.29	0.41	0.58	0.42	0.58	0.71	0	0.04	0.12
Combined	0.31	0.43	0.59	0.42	0.58	0.71	0	0.05	0.12
40 samples, top 5									
Experimental	0.26	0.4	0.54	0.39	0.53	0.69	0.03	0.09	0.18
Artificial	0.29	0.41	0.55	0.39	0.54	0.7	0	0.07	0.14
Combined	0.32	0.43	0.55	0.41	0.56	0.71	0	0.06	0.14

[a] Total number of favorable interactions normalized by the number of ligand heavy atoms. [b] A fraction of the heavy ligand atoms participating in at least one non-bond interaction. [c] Total number of unfavorable interactions normalized by the number of ligand heavy atoms.

the integration of the combined data for training purposes led to a decrease in the proportion of samples exhibiting steric clashes, as shown in Table 4.

A positive correlation was identified between the amount of small molecule heavy atoms and the metrics such as RMSD and steric clashes. Additionally, a negative correlation was found between the count of atoms and favorable non-bond interaction rates, as depicted in Fig. S2.† This indicates that the model performance decreases with the increase of ligand size.

The final production PocketCFDM model was trained for 80 epochs using high-quality experimental data (used for the non-bond interactions statistics retrieval) and artificial samples, similar to the aforementioned combined dataset settings. A significant reduction in the occurrence of protein–ligand steric clashes was observed, with percentages of 19.31% and 13.90% for the top-1 and best of top-5 samples, respectively. This is a nearly 10% decrease compared to the most optimal model trained for 25 epochs (Table 4). There has been no substantial improvement in other metrics.

Table 4 The percentage of samples within the predicted PDBbind test complexes exhibiting at least one steric clash. The models generated 40 ligand poses. The best of top-5 candidates are picked based on the lowest number of clashes

	Steric clashes	
Dataset	Top 1	Top 5
Experimental	33.59%	30.12%
Artificial	30.50%	26.64%
Combined	28.19%	23.94%

Table 5 Comparison between the DiffDock and PocketCFDM on the test PDBbind dataset

Metric	PocketCFDM	DiffDock
Avg. inference time (s); 40 samples	49	90
RMSD; median; 40 samples; top-1	5.14	2.8
RMSD; median; 40 samples; top-5	3.02	2.17
Centroid distance; median; 40 samples; top-1	1.4	1.01
Centroid distance; median; 40 samples; top-5	1.08	0.81
Favourable rate; median; 40 samples; top-1	0.43	0.37
Favourable rate; median; 40 samples; top-5	0.42	0.39
Favourable rate uniq; median; 40 samples; top-1	0.58	0.47
Favourable rate uniq; median; 40 samples; top-5	0.56	0.5
Unfavourable rate; median; 40 samples; top-1	0.02	0.07
Unfavourable rate; median; 40 samples; top-5	0.07	0.08
Steric clashes; 40 samples; top-1	19.31%	46.51%
Steric clashes; 40 samples; top-5	13.90%	25.97%

Comparison with other methods

We also performed a detailed comparison between PocketCFDM and DiffDock, which is currently considered as the most accurate AI technique for predicting ligand binding poses (Table 5).

DiffDock exhibited superior accuracy in terms of RMSD and centroid distance. In contrast, PocketCFDM exhibited superior outcomes in terms of favorable and unfavorable non-covalent interactions, as well as a notably lower incidence of steric clashes. It was expected due to the divergence in the scoring algorithms employed to evaluate and rank the generated samples. The DiffDock approach utilized a confidence model that was trained to prioritize samples with lower RMSD to the actual ligand pose. However, in our context, the scoring approach was more focused on the number of non-covalent contacts and the absence of steric clashes, without considering the specific conformation of the ligand pose. Another notable difference pertaining to the disparity in scoring functions is the diversity observed within anticipated poses. The median RMSD values between the alternative top-5 samples in the PocketCFDM and DiffDock methods were 2.59 and 1.29 respectively. Additionally, it was noted that the mean inference time for the PocketCFDM was approximately 1.8 times quicker. This is attributed mostly to the reduced size of the protein graph and the decrease in the graph's node feature space.

PocketCFDM differs from DiffDock in both the scoring function and the training dataset. DiffDock is trained on the residue-level full protein graphs, while PocketCFDM utilizes the atomic-level graphs limited to the binding pockets (real or artificial, see the "Model training and inference" section above). The training dataset for PocketCFDM is augmented by artificially generated pockets, while DiffDock is trained on the experimentally resolved protein–ligand complexes only. Thus observed better performance of PocketCFDM originates from both of these factors.

Additionally, we compared PocketCFDM performance to conventional molecular docking methods such as Autodock Vina,[30] QuickVina-W,[31] GNINA,[32] SMINA,[33] and GLIDE.[34] The results of traditional docking programs on the PDBbind dataset were reported by the DiffDock authors.[8] According to Table 6, PocketCFDM performs comparably to the conventional docking methods. It is worth mentioning that the ML-based methods don't guarantee the absence of inter-/intramolecular clashes in

Table 6 Comparison between the PocketCFDM, DiffDock, and conventional docking techniques on the test PDBbind dataset. The ML-based method results are reported for the best of the top 5 pose candidates among 40 predicted samples. ↑ – means the higher the better; ↓ – means the lower the better

	RMSD					Centroid distance				
	Percentiles ↓			% Below threshold ↑		Percentiles ↓			% Below threshold ↑	
Method	25th	50th	75th	5 Å	2 Å	25th	50th	75th	5 Å	2 Å
Autodock Vina (top-1)	5.7	10.7	21.4	21.2	5.5	1.9	6.2	20.1	47.1	26.5
QuickVina-W (top-1)	2.5	7.7	23.7	40.2	20.9	0.9	3.7	22.9	54.6	41
GNINA (top-1)	2.4	7.7	17.9	40.8	22.9	0.8	3.7	23.1	53.6	40.2
SMINA (top-1)	3.1	7.1	17.9	38	18.7	1	2.6	16.1	59.8	41.6
GLIDE (top-1)	2.6	9.3	28.1	33.6	21.8	0.8	5.6	26.9	48.7	36.1
GNINA (top-5)	1.6	4.5	11.8	52.8	29.3	0.6	2	8.2	66.8	49.7
SMINA (top-5)	1.7	4.6	9.7	53.1	29.3	0.6	1.85	6.2	72.9	50.8
DiffDock	1.2	2.4	5	75.5	44.7	0.4	0.9	1.9	88	76.7
PocketCFDM	2.1	3.3	4.6	82.2	23.9	0.8	1.3	2.0	98.8	75.3

the docked complexes while the traditional approaches mostly provide physically valid poses.

Limitations and perspectives

The primary area of enhancement of the PocketCFDM model lies in inference time, which is currently still too large for efficient practical deployment. The mean time required to predict 40 ligand poses is around ~50 seconds at 24 GB NVIDIA L4 GPU. The inference time could be improved significantly by replacing computationally intensive EGNN blocks with more efficient alternatives like GVP. The inclusion of the artificial pockets into the training of even simpler regression-based techniques, such as EquiBind and TANKBind, could also be beneficial since these architectures are generally more sensitive to the number of distinct training samples. It is possible to explore the usage of multiple augmentations of the same input instead of directly including specific symmetries and equivariances into graph neural network (GNN) designs. The utilization of such augmentation is popular in convolutional neural networks (CNNs). Moreover, it demonstrated promising outcomes in the domain of geometric graph learning.[35] This strategy may potentially increase the model training time while resulting in reduced inference time.

Another notable drawback of the present proof-of-the-principle implementation is the possibility of ligand self-intersections (intramolecular clashes), which have to be filtered out during the post-processing steps. Incorporating intramolecular and intermolecular clashes into the loss function during the training process could potentially address this problem.

Also, we believe that incorporating larger ligands into the training process will address the challenges encountered with relatively large compounds. The ZINC20 dataset, which was utilized in this work, has only 2.5% of molecules larger than 40 heavy atoms, which makes them underrepresented during the model training. Although the median size of the ligands in this dataset is 25 heavy atoms, which is quite common for the datasets of drug-like molecules, a higher percentage of large ligands may enhance the inference capabilities for larger compounds while maintaining the same level of performance for smaller molecules.

It should be pointed out that the current iteration of the pocket generation algorithm doesn't consider water molecules, ions, and metal atoms, which are known to be important mediators of interactions in a significant amount of protein–ligand complexes. Thus currently our technique should be used with caution in the cases when the involvement of water, ions, and the metal atoms in the ligand binding is anticipated. We assume that the inclusion of these components into the artificial pockets in the next iterations of our technique should further improve the model performance and universality.

Another shortcoming of the current pocket generation algorithm is the frequent formation of "unintended" contacts (mainly hydrophobic) while generating other types of interactions, which is evident from the disbalance between the number of non-bond interactions found in the experimental and artificial pockets. When the pocket building block is added, additional interactions could be formed accidently apart from the intended contact pair. Additional checks could be added in future versions of the algorithm to minimize the amount of such unwanted random interactions.

Although the present work concentrates on predicting the ligand poses, it is necessary to note that the pose prediction is only a part of the accurate prediction of the protein–ligand binding. Two other important components are the accurate scoring of the binding poses in terms of affinity or activity and the accounting for protein flexibility and dynamics. The former is currently being addressed not only by traditional docking force fields but also by ML-based pose rescoring techniques.[36] The latter problem could be tackled by multiple approaches including flexible and ensemble docking. Accounting for the protein flexibility and the ensembles of protein conformations are among the future directions of improvement for our data augmentation technique.

Conclusions

In this work, we introduced the PocketCFDM generative diffusion model for predicting the poses of small molecules in the protein binding pockets. The model is trained using an innovative approach of data augmentation, which involves the construction of a large number of artificial binding pockets that follow the same statistical patterns of non-bond interactions as the real ones. In order to construct such artificial pockets we thoroughly evaluated the statistical characteristics of non-bond interactions in the real protein–ligand complexes and designed an algorithmic approach that reproduces them in artificial pockets, which are built around given small molecule conformers. The performance of the models trained on experimental data only, artificial data only, and a combination of both was evaluated and compared to the currently most promising ML model for binding pose prediction DiffDock. It is shown that the inclusion of artificial binding pockets into the model training resulted in a significant increase of model performance. Particularly, PocketCFDM outperforms DiffDock in terms of the non-bond interaction counts, the number of steric clashes, and the inference speed. The prospects of further improvements of PocketCFDM are discussed. The inference code, final model weights, and model prediction examples are publicly available in the GitHub repository (**https://github.com/vtarasv/pocket-cfdm.git**).

Author contributions

VB, SS and AN designed the study and supervised data quality of generated pocket as well as model development and testing. SY coordinated the work and participated in results interpretation. TV developed the modules for pocket and features generation, model training and testing. TV, IK, and RS researched existing methods for machine-learning-based molecular docking. LP, ZO, and RZ participated in consultations about the best practices and strategies for model tuning and efficient training. IK,

PH, and VV performed and supervised the tuning, training and testing process. The manuscript was written by TV and SY.

Conflicts of interest

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

Acknowledgements

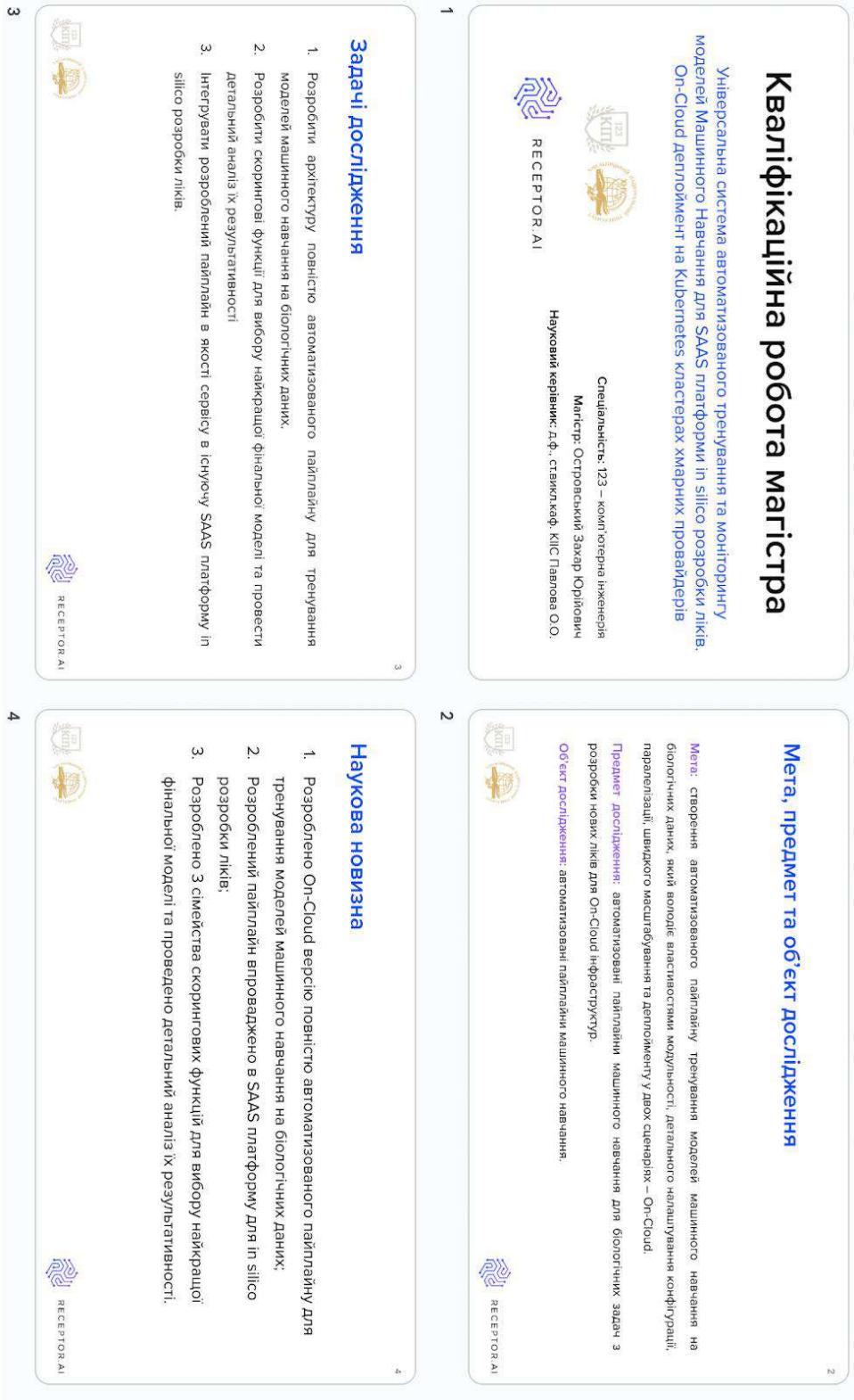
SY was supported through the MSCA4Ukraine project, which is funded by the European Union.

References

1 X. Li, Y. Li, T. Cheng, Z. Liu and R. Wang, *J. Comput. Chem.*, 2010, **31**, 2109–2125.
2 J. B. Ghasemi, A. Abdolmaleki and F. Shiri, in *Pharmaceutical Sciences: Breakthroughs in Research and Practice*, IGI Global, 2017, pp. 770–794.
3 W. Lu, Q. Wu, J. Zhang, J. Rao, C. Li and S. Zheng, *bioRxiv*, 2022, preprint, DOI: **10.1101/2022.06.06.495043**.
4 H. Stärk, O.-E. Ganea, L. Pattanaik, R. Barzilay and T. Jaakkola, *arXiv*, 2022, preprint, arXiv:2202.05146, DOI: **10.48550/arXiv.2202.05146**.
5 B. Jing, S. Eismann, P. Suriana, R. J. L. Townshend and R. Dror, *arXiv*, 2021, preprint, arXiv:2009.01411, DOI: **10.48550/arXiv.2009.01411**.
6 B. Jing, S. Eismann, P. N. Soni and R. O. Dror, *arXiv*, 2021, preprint, arXiv:2106.03843, DOI: **10.48550/arXiv.2106.03843**.
7 O.-E. Ganea, X. Huang, C. Bunne, Y. Bian, R. Barzilay, T. Jaakkola and A. Krause, *arXiv*, 2022, preprint, arXiv:2111.07786, DOI: **10.48550/arXiv.2111.07786**.
8 G. Corso, H. Stärk, B. Jing, R. Barzilay and T. Jaakkola, *arXiv*, 2023, preprint, arXiv:2210.01776, DOI: **10.48550/arXiv.2210.01776**.
9 R. Wang, X. Fang, Y. Lu and S. Wang, *J. Med. Chem.*, 2004, **47**, 2977–2980.
10 Z. Liu, Y. Li, L. Han, J. Li, J. Liu, Z. Zhao, W. Nie, Y. Liu and R. Wang, *Bioinformatics*, 2015, **31**, 405–412.
11 Ž. H. Petrovski, B. Hribar-Lee and Z. Bosnić, *Pharmaceutics*, 2022, **15**, 119.
12 J. Kandel, H. Tayara and K. T. Chong, *J. Cheminf.*, 2021, **13**, 65.
13 M. Volkov, J.-A. Turk, N. Drizard, N. Martin, B. Hoffmann, Y. Gaston-Mathé and D. Rognan, *J. Med. Chem.*, 2022, **65**, 7946–7958.
14 The UniProt Consortium, *Nucleic Acids Res.*, 2019, **47**, D506–D515.
15 J.-L. Reymond, R. Van Deursen, L. C. Blum and L. Ruddigkeit, *MedChemComm*, 2010, **1**, 30.
16 C. Bissantz, B. Kuhn and M. Stahl, *J. Med. Chem.*, 2010, **53**, 5061–5084.
17 R. Ferreira De Freitas and M. Schapira, *MedChemComm*, 2017, **8**, 1970–1981.
18 R. Wilcken, M. O. Zimmermann, A. Lange, A. C. Joerger and F. M. Boeckler, *J. Med. Chem.*, 2013, **56**, 1363–1388.
19 B. Kuhn, E. Gilberg, R. Taylor, J. Cole and O. Korb, *J. Med. Chem.*, 2019, **62**, 10441–10455.
20 M. Wójcikowski, P. Zielenkiewicz and P. Siedlecki, *J. Cheminf.*, 2015, **7**, 26.
21 H. C. Jubb, A. P. Higueruelo, B. Ochoa-Montaño, W. R. Pitt, D. B. Ascher and T. L. Blundell, *J. Mol. Biol.*, 2017, **429**, 365–371.
22 M. F. Adasme, K. L. Linnemann, S. N. Bolz, F. Kaiser, S. Salentin, V. J. Haupt and M. Schroeder, *Nucleic Acids Res.*, 2021, **49**, W530–W534.
23 M. Z. Tien, D. K. Sydykova, A. G. Meyer and C. O. Wilke, *PeerJ*, 2013, **1**, e80.
24 J. J. Irwin, K. G. Tang, J. Young, C. Dandarchuluun, B. R. Wong, M. Khurelbaatar, Y. S. Moroz, J. Mayfield and R. A. Sayle, *J. Chem. Inf. Model.*, 2020, **60**, 6065–6073.
25 T. Sterling and J. J. Irwin, *J. Chem. Inf. Model.*, 2015, **55**, 2324–2337.
26 A. P. Bento, A. Hersey, E. Félix, G. Landrum, A. Gaulton, F. Atkinson, L. J. Bellis, M. De Veij and A. R. Leach, *J. Cheminf.*, 2020, **12**, 51.
27 R. Meli and P. C. Biggin, *J. Cheminf.*, 2020, **12**, 49.
28 N. Thomas, T. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff and P. Riley, *arXiv*, 2018, preprint, arXiv:1802.08219, DOI: **10.48550/arXiv.1802.08219**.
29 M. Geiger and T. Smidt, *arXiv*, 2022, preprint, arXiv.2207.09453, DOI: **10.48550/arXiv.2207.09453**.
30 O. Trott and A. J. Olson, *J. Comput. Chem.*, 2010, **31**, 455–461.
31 N. M. Hassan, A. A. Alhossary, Y. Mu and C.-K. Kwoh, *Sci. Rep.*, 2017, **7**, 15451.
32 A. T. McNutt, P. Francoeur, R. Aggarwal, T. Masuda, R. Meli, M. Ragoza, J. Sunseri and D. R. Koes, *J. Cheminf.*, 2021, **13**, 43.
33 D. R. Koes, M. P. Baumgartner and C. J. Camacho, *J. Chem. Inf. Model.*, 2013, **53**, 1893–1904.
34 T. A. Halgren, R. B. Murphy, R. A. Friesner, H. S. Beard, L. L. Frye, W. T. Pollard and J. L. Banks, *J. Med. Chem.*, 2004, **47**, 1750–1759.
35 FAENet, *Frame Averaging Equivariant GNN for Materials Modeling | OpenReview*, **https://openreview.net/forum?id=HRDRZNxQXc**, accessed August 16, 2023.
36 C. Yang, E. A. Chen and Y. Zhang, *Molecules*, 2022, **27**, 4568.

ДОДАТОК В
(обов'язковий)

ПРЕЗЕНТАЦІЯ ДО ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ



5

Практична цінність

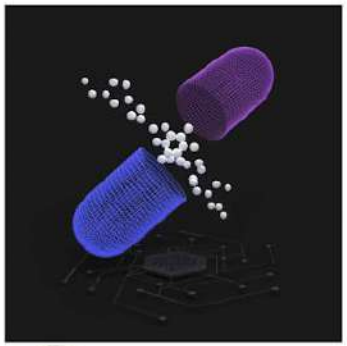
Практична значимість отриманих результатів полягає у можливості автоматичного тренування моделей машинного навчання на молекулярних датасетах і подальшого їх використання у SAAS платформі.

RECEPTOR.AI

5

6

Вступ



Сучасний світ даних та технологій постійно розвивається, пропонуючи все нові виклики та можливості у сфері машинного навчання та штучного інтелекту. В контексті стрімкого прогресу, важливість ефективного та швидкого впровадження машинного навчання у різноманітні промислові та дослідницькі процеси не може бути переоцінена.

Особлива увага у цьому контексті приділяється підходам автоматизації машинного навчання, які відомі під загальною назвою AutoML.

RECEPTOR.AI

6

Розділ 1. Огляд існуючих рішень

AIDDISON

Переваги:

- автоматизований пайплайн для тренування ADME-Tox моделей;
- гнучкість налаштування параметрів тренування.

Недоліки:

- відсутність функціоналу для аналізу результатів натренованих моделей.

Bioconductor

Переваги:

- використання оптимальних хмарних сервісів для тренування моделей;
- візуалізація результатів.

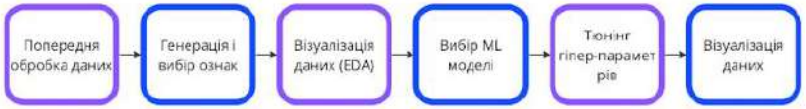
Недоліки:

- складність використання.

RECEPTOR.AI

7

Розділ 1. Основні складові пайплайну автоматичного тренування моделей машинного навчання



RECEPTOR.AI

8

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

9

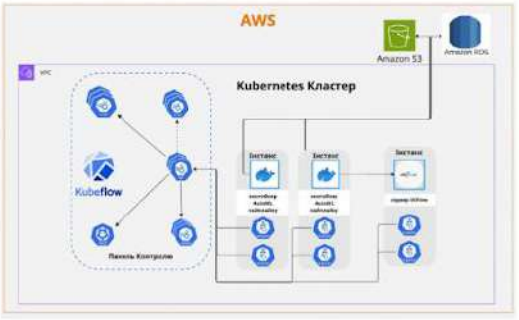
Розділ 3. Дослідження і вибір технології On-Cloud оркестрації

Параметр	KubeFlow	Argo
Сфера застосування	**Спеціалізовано для ML**	Загальна оркестрація робочих процесів
Інтеграція	**Глибока інтеграція з ML інструментами**	Проста інтеграція з Kubernetes
Гнучкість	**Висока в контексті ML**	Висока в контексті загальної оркестрації
Функціональність	**Висока для ML пайплайнів**	Середня для ML пайплайнів

RECEPTOR.AI

10

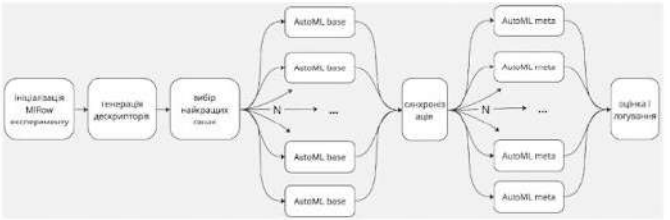
Розділ 3. Архітектура системи у середовищі Kubernetes з оркестрацією KubeFlow



RECEPTOR.AI

11

Розділ 3. Архітектура AutoML пайплайну



RECEPTOR.AI

12

Розділ 4. Створення скорингових функцій для етапу вибору найкращої моделі

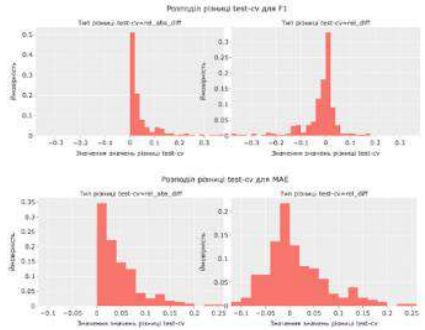
$$rel_{diff} = \frac{test - cv}{cv} \qquad rel_{abs_{diff}} = \frac{|test - cv|}{cv}$$

де rel_{diff} – відносна різниця;

$rel_{abs_{diff}}$ – відносна абсолютна різниця;

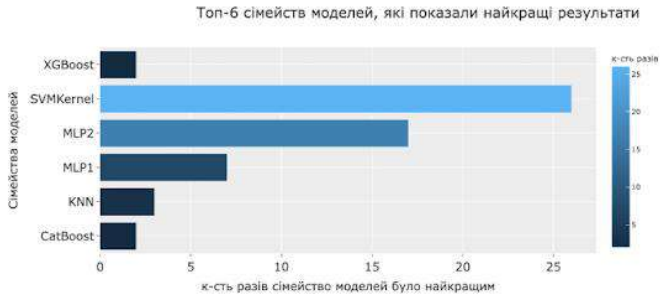
$test$ – значення метрики на тестовій вибірці;

cv – значення метрики на крос-валідаційній вибірці.



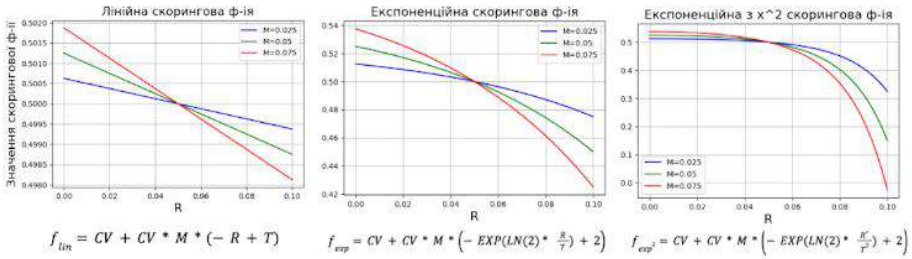
RECEPTOR.AI

Розділ 4. Топ-6 сімейств моделей



RECEPTOR.AI

Розділ 4. Запропоновані сімейства скорингових функцій



$$f_{lin} = CV + CV * M * (-R + T)$$

$$f_{exp} = CV + CV * M * \left(-EXP(LN(2) * \frac{R}{T}) + 2\right)$$

$$f_{exp^2} = CV + CV * M * \left(-EXP(LN(2) * \frac{R^2}{T^2}) + 2\right)$$

де CV – оцінка CV за метрикою оптимізації;

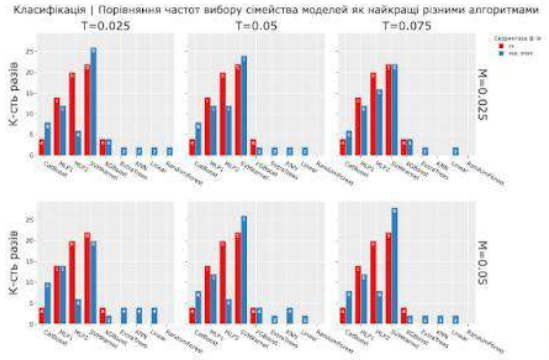
R – відносна абсолютна різниця між оцінками CV та тестовими оцінками;

T – поріг толерантності, що представляє максимально допустиму величину R;

M – частка оцінки CV, що додасться як бонус у ідеальному випадку, коли $R = 0$.

RECEPTOR.AI

Розділ 4. Оцінка впливу скорингових функцій на вибір моделей в AutoML пайплайні



RECEPTOR.AI

17

Наукова публікація


RSC Advances

PAPER

Augmenting a training dataset of the generative diffusion model for molecular docking with: artificial binding pockets†

Introduction

2024 | *cited in Scopus* | *Q1 journal*

Augmenting A Training Dataset Of The Generative Diffusion Model For Molecular Docking With Artificial Binding Pockets

Article | *RSC advances (Q1) 14 (2), 1341-1353*

T. Voitsitskyi, V. Bdzhola, R. Stratiichuk, I. Koleiev, **Z. Ostrovsky**, V. Vozniak et al.

RECEPTOR.AI

17

18

Висновки

- На основі проведених досліджень було розроблено архітектура і компоненти програмного забезпечення, яке інтегровано в існуючу SAAS платформу з розробки ліків.
- 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 начання, 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-Cloud деплойменту на Kubernetes кластерах хмарних провайдерів. Також наводиться порівняння різних технологій для реалізації поставленої задачі і обгрунтовується фінальне рішення.
- У четвертому розділі міститься опис створення скорингових функцій для вибору найкращої моделі отриманої автоматичним навчанням. Проведено ретельне порівняння запропонованих сімейств скорингових функцій для задач бінарної класифікації та регресії, та обрано найкращі скорингові функції та значення їх параметрів для задач регресії та класифікації по критерії максимальної робастності натренованих моделей.

RECEPTOR.AI

18

19

Дякую за увагу!

RECEPTOR.AI

19

UNICHECK
by Turnitin

Ім'я користувача:
Кафедра КІ

ID перевірки:
1016257541

Дата перевірки:
16.05.2024 21:40:37 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
16.05.2024 21:44:04 EEST

ID користувача:
100005591

Назва документа: Островський_Універсальна система автоматизованого тренування та моніторингу моделей...

Кількість сторінок: 97 **Кількість слів:** 15740 **Кількість символів:** 128881 **Розмір файлу:** 5.79 MB **ID файлу:** 1016045122

1.62%
Схожість

Найбільша схожість: 0.53% з джерелом з Бібліотеки (ID файлу: 1010865386)

1.41% Джерела з Інтернету	70	Сторінка 99
0.79% Джерела з Бібліотеки	58	Сторінка 99

0.08% Цитат

Цитати	1	Сторінка 100
Посилання	1	Сторінка 100

0%
Вилучень

Немає вилучених джерел

Модифікації

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

Замінені символи	20

Thu May 16 20:23:44 EEST 2024, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 7.0%

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

	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
ID: 126452 Назва: МКР Універсальна система автоматизованого тренування та моніторингу моделей Машинного Навчання для SAAS платформи in silico розробки ліків. On-Cloud деплоймент на Kubernetes кластерах хмарних провайдерів Додано в БД: 2024-05-16 Автора: Островський З.Ю. Керівники: Павлова О.О. Консультанти: Опоненти:	102260	959	7228 (7%)	104 (11%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Островський Захар Юрійович

Тема: Універсальна система автоматизованого тренування та моніторингу моделей Машинного Навчання для SAAS платформи in silico розробки ліків. On-Cloud деплоймент на Kubernetes кластерах хмарних провайдерів.

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг дипломної роботи:

Кількість сторінок записки 84 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-Cloud деплойменту на Kubernetes кластерах хмарних провайдерів. У четвертому розділі Інтеграція пайлайну в SAAS платформу для розробки ліків. Наукова новизна отриманих результатів полягає у тому, що вперше розроблено On-Cloud 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) _____

Вормак Олександр Володимирович, д.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.

Островського Захара Юрійовича
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.

23 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-Cloud деплоймент на Kubernetes кластерах хмарних провайдерів

Автор: Островський Захар Юрійович

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 1.62% і адресується до 128 першоджерел; та системою Anti-Plagiarism складає 7%, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ О. О. Павлова

Гарант ОП _____ О. С. Савенко

Завідувач кафедри КПС _____ Т. О. Говорущенко