НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра математичного моделювання та аналізу даних

«До захисту допущено»
В.о. завідувача кафедри
_____ І.М. Терещенко
«__» _____ 2024 р.

Дипломна робота

на здобуття ступеня бакалавра

зі спеціальності: 113 Прикладна математика
на тему: **«Порівняння багатошарового перцептрону та $(1+\lambda)$-еволюційного алгоритму для задач класифікації»**

Виконав: студент 4 курсу, групи ФІ-02
Харь Дмитро Федорович

Керівник: асистент кафедри ММАД Яворський О.А. _____

Рецензент: звання, степінь, посада Прізвище І.П. _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Київ — 2024

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра математичного моделювання та аналізу даних

Рівень вищої освіти — перший (бакалаврський)
Спеціальність (освітня програма) — 113 Прикладна математика,
ОПП «Математичні методи моделювання, розпізнавання образів та комп'ютерного зору»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ І.М. Терещенко
«__» _____ 2024 р.

ЗАВДАННЯ
на дипломну роботу

Студент: Харь Дмитро Федорович

1. Тема роботи: «*Порівняння багатошарового перцептрону та* $(1+\lambda)$*-еволюційного алгоритму для задач класифікації*»,

керівник: асистент кафедри ММАД Яворський О.А.,

затверджені наказом по університету №__ від «__» _____ 2024 р.

2. Термін подання студентом роботи: «__» _____ 2024 р.

3. Вихідні дані до роботи:

4. Зміст роботи: *Порівняльний аналіз багатошарового перцептрону (англ. MLP, Multilayer Perceptron) з оптимізаційними алгоритмами в основі яких градієнтний спуск, MLP з оптимізаційним алгоритмом в основі якого одноточкова мутація та* $(1+\lambda)$*-еволюційного алгоритму з кодуванням генетичного програмування (англ.* $(1+\lambda)$*-EA with GP encoding,* $(1+\lambda)$*-evolutionary algorithm with genetic programming encoding), на прикладі задач бінарної та мультикласової класифікації табличних даних та картинок.*

5. Перелік ілюстративного матеріалу: *«Презентація доповіді»*

6. Дата видачі завдання: 10 грудня 2023 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання	Примітка
1	Узгодження теми роботи із науковим керівником	листопад-грудень 2023 р.	Виконано
2	Огляд та опрацювання опублікованих джерел за тематикою дослідження	грудень 2023 р - лютий 2024 р.	Виконано
3	Написання програмного забезпечення та проведення дослідження	березень-квітень 2024 р.	Виконано
4	Оформлення та опис результатів	травень 2024 р.	Виконано
5	Написання та оформлення дипломної роботи	травень-червень 2024 р.	Виконано
6	Отримання рекомендації до захисту	08.06.2024	Виконано

Студент _____ Харь Д.Ф.

Керівник _____ Яворський О.А.

РЕФЕРАТ

Кваліфікаційна робота містить: ??? стор., ??? рисунки, ??? таблиць, ??? джерел.

У даній роботі розглядаються методи для вирішення задач класифікації, а саме: MLP, який використовує оптимізаційні алгоритми в основі яких градієнтний спуск, MLP, який використовує оптимізаційний алгоритм на основі одноточкової мутації та $(1 + \lambda)$-EA with GP encoding. Ці методи порівнювались в задачах бінарної та мультикласової класифікації табличних даних та картинок.

У ході дослідження було встановлено, що всі три методи здатні досягти однакових метрик у всіх задачах. Найшвидшу сходимість до цих метрик продемонстрував MLP з використанням градієнтного спуску. Тим не менш, $(1 + \lambda)$-EA with GP encoding виділився завдяки здатності легко адаптуватись до умов задачі. Цей метод дозволяє вибирати кількість нащадків і тип мутацій, що надає можливість зосередити пошук рішень у конкретних областях простору рішень. Такий підхід є особливо корисним, коли потрібно зосередитися на важливих регіонах пошуку для вдосконалення рішень.

МАШИННЕ НАВЧАННЯ, ЕВОЛЮЦІЙНІ АЛГОРИТМИ, ГЕНЕТИЧНЕ ПРОГРАМУВАННЯ, МЕТОДИ ОПТИМІЗАЦІЇ, ЕКСПРЕСИВНІ КОДУВАННЯ

ABSTRACT

This paper considers methods for solving classification problems, namely: MLP, which uses optimization algorithms based on gradient descent, MLP, which uses an optimization algorithm based on one-point mutation, and $(1+\lambda)$-EA with GP encoding. These methods were compared in the tasks of binary and multiclass classification of tabular data and pictures.

During the research, it was established that all three methods are able to achieve the same metrics in all tasks. The fastest convergence to these metrics was demonstrated by MLP using gradient descent. Nevertheless, the $(1 + \lambda)$-EA with GP encoding stood out due to its ability to easily adapt to the task conditions. This method allows you to choose the number of offspring and the type of mutations, which makes it possible to focus the search for solutions in specific regions of the solution space. This approach is particularly useful when focusing on important search regions to improve solutions.

MACHINE LEARNING, EVOLUTIONARY ALGORITHMS, GENETIC PROGRAMMING, OPTIMIZATION METHODS, EXPRESSIVE ENCODINGS

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ML — машинне навчання (англ. Machine Learning)

MLP — багатошаровий перцептрон (англ. Multilayer Perceptron)

EA — еволюційний алгоритм (англ. Evolutionary Algorithm)

GP — генетичне програмування (англ. Genetic Programming)

Adam — адаптивна оцінка моменту (англ. Adaptive Moment Estimation)

$(1 + \lambda)$-EA with GP encodings — еволюційний алгоритм, який може використовуватися для вирішення задач класифікації (англ. $(1 + \lambda)$-Evolutionary Algorithm with Genetic Programming encodings).

MLP with gradient descent — багатошаровий перцептрон, який використовує метод на основі градієнтного спуску, в якості оптимізаційного алгоритму.

MLP with single-point mutation — багатошаровий перцептрон, який використовує одноточкову мутацію, в якості оптимізаційного алгоритму.

Фітнес-функція — це функція $F : \mathcal{S} \to \mathbb{R}$, яка відображає представлення рішення S на дійсне число f.

Індивід — I в еволюційних алгоритмах визначається як кортеж $I = (S, f)$, де S є представленням рішення в просторі рішень $\mathcal{S}$, природа S залежить від конкретної проблеми та може варіюватися в широких межах, від двійкових рядків, дійсних векторів, дерев до більш складних структур даних, f — це значення фітнес-функції, пов'язане з індивідом, кількісно оцінюючи якість індивіда як рішення цільової проблеми.

Популяція — P визначається як множина індивідів $P = \{I_1, I_2, \ldots, I_N\}$, де кожен окремий I_i є варіантом вирішення розв'язуваної проблеми.

Кросинговер — C, є бінарною функцією, яка бере два індивіди з

популяції як вхідні дані та створює одне або більше нащадків, потенційно включаючи генетичний матеріал від обох батьків. Формально це можна виразити так: $C : (I_i, I_j) \to (I_{i'}, I_{j'})$, де I_i і I_j є батьківськими індивідами, кожен з яких містить представлення рішення та значення фітнес-функції, $I_{i'}$ і $I_{j'}$ є особинами-нащадками, отриманими в результаті операції кросинговеру.

Мутація — це функція $M : I \to I'$, де I — оригінальний індивід, I' є мутованою особиною з потенційно зміненим представленням рішення S' і відповідним новим значенням фітнес-функції f', яка застосовує стохастичну модифікацію до індивіду, що потенційно призводить до появи нового варіанту рішення.

Контрольованість у контексті $(1 + \lambda)$-еволюційного алгоритму з генетичним програмуванням — визначається як здатність алгоритму дозволяти користувачу точно регулювати його параметри (наприклад, кількість нащадків λ і стратегії мутації), щоб оптимізувати процес пошуку рішення та адаптувати його під специфічні умови задачі.

Precision — це метрика, яка визначає відношення кількості правильно класифікованих позитивних прикладів до загальної кількості прикладів, що були класифіковані як позитивні.

Recall — це метрика, яка визначає відношення кількості правильно класифікованих позитивних прикладів до загальної кількості справді позитивних прикладів.

F1-score — це гармонійне середнє між precision та recall.

Expressive Encodings — виразне кодування.

ВСТУП

Актуальність дослідження. Використання класифікаційних задач має широкий спектр застосування в різних сферах наукових досліджень та практичних областях. Наприклад, важливо класифікувати, чи особа є носієм певного захворювання, спираючись на ретгенівські знімки або аналізи крові, що дозволяє з високою точністю визначати наявність патологій. З цього випливає необхідність дослідження алгоритмів, які вирішують задачу класифікації, для того, щоб мати більшу гнучкість у налаштуванні процесу пошуку по певним областям простору рішень. Відповідно до цього актуальність дослідження полягає у порівнянні алгоритимів MLP та $(1 + \lambda)$-EA with GP encodings, щоб перевірити чи надає останній можливість контролювати гіперпараметри для більш детального пошуку та можливість краще адаптуватися до поточної задачі.

Метою дослідження є пошук оптимального методу класифікації серед методів MLP with gradient descent, MLP with single-point mutation, $(1 + \lambda)$-EA with GP encodings, для дослідження контрольованості (див. означення у розділі перелік умовних позначень, скорочень і термінів) у розв'язанні задач бінарної та мультикласової класифікації табличних даних та картинок.

Об'єктом дослідження є якісна поведінка MLP та $(1 + \lambda)$-еволюційних алгоритмів для задачі бінарної та мультикласової класифікації.

Предметом дослідження є особливості контролювання алгоритмів на прикладі MLP with gradient descent, MLP with single-point mutation, $(1 + \lambda)$-EA with GP encodings на прикладі застосування до задач бінарної та мультикласової класифікації табличних даних та картинок.

Наукова новизна полягає в дослідженні та порівнянні алгоритмів MLP with gradient descent, MLP with single-point mutation, $(1 + \lambda)$-EA with

GP encodings на прикладі задач бінарної та мультикласової класифікації.

Практичне значення результатів полягає в використанні перелічених вище методів, для задачі класифікації, для покращення контрольованості і збереженню такої ж точності та швидкості, як і в класичних методах.

1 МЕТОДИ ТА ПІДХОДИ ВИРІШЕННЯ ЗАДАЧ КЛАСИФІКАЦІЇ

В даному розділі будуть основні теоретичні відомості про об'єкт дослідження та огляд суміжних робіт в даній сфері.

1.1 Задача класифікації: визначення, види

Класифікація – це процес віднесення об'єкту до певної категорії або класу на основі його характеристик, серед заздалегідь встановленого набору категорій. Класифікація може бути бінарною, багатокласовою, багатомітковою, ієрархічною та інші. Бінарна класифікація – це класифікація, коли кожному об'єкту обирається група з наперед визначеної множини груп в якій знаходиться рівно дві групи; багатокласова класифікація – це класифікація, коли кожному об'єкту обирається група з наперед визначеної множини груп в якій може знаходиться довільна кількість груп. В поточній роботі ми зосередимося на бінарній та багатокласовій класифікації.

Задача класифікації зустрічається в багатьох сферах, наприклад: медицина (діагностика раку на основі зображень МРТ), фінанси (класифікація позичальників як „надійних“ чи „ризикованих“ на основі їхньої кредитної історії), роздрібна торгівля (класифікація покупців за типами покупок для надання персоналізованих знижок), транспорт (розрізнення між легковими авто, вантажівками та мотоциклами на дорозі), освіта (ідентифікація студентів, яким потрібна додаткова допомога в певних предметах), безпека (класифікація електронних листів як „безпечні,“ „спам“ або „фішинг“), біотехнології (розпізнавання мутацій, що спричиняють хвороби).

1.2 Способи вирішення задачі класифікації

Існує декілька способів для вирішення задачі класифікації: класичні статистичні методи (наприклад, логістична регресія [24]), алгоритми ML (наприклад, метод k-найближчих сусідів [9]), глибинне навчання (за допомогою нейронних мереж), а також задачу класифікації можна вирішувати за допомогою генетичних алгоритмів.

На початку розглянемо методи глибинного навчання для вирішення задач класифікації. Обчислювальним об'єктом в глибинному навчанні є нейронна мережа. Існують різноманітні типи нейронних мереж, але ми будемо їх розглядати на прикладі MLP [27], оскільки саме його ми використовуємо для експериментів. MLP [27] складається з шарів нейронів. Кожен нейрон в шарі, приймає вхідні дані з попереднього шару та обчислює вихідний сигнал, який передається наступному шару. Формально штучний нейрон можна описати наступним чином:

$$a = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{1.1}$$

де $x_1, x_2, \ldots, x_n$ — вхідні сигнали до нейрону; $w_1, w_2, \ldots, w_n$ — ваги, що призначені для кожного вхідного сигналу; b — зсув (англ. bias), що додається до суми зважених вхідних сигналів; f — активаційна функція, яка має наступні властивості: нелінійність, дииференційовність, неперервність, монотонність.

В нейронній мережі може бути довільна кількість шарів та в кожному шарі може бути довільна кількість нейронів. Усі вони працюють за вище наведеним принципом: на вхід кожному нейрону в кожному шарі приходить сигнал з попереднього шару і кожний нейрон генерує вихід, якщо це перший шар, то на вхід подаються самі дані. Загалом схема нейронної мережі може виглядати наступним чином (рисунок 1.1).

Далі розглянемо генетичні алгоритми для вирішення задач класифікації. Обчислювальним об'єктом в генетичних алгоритмах є

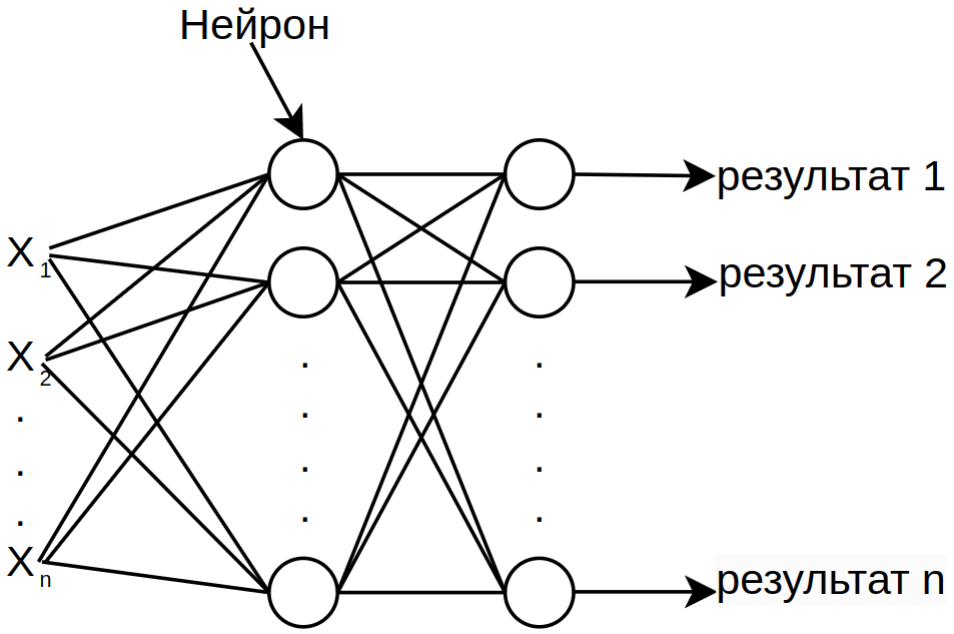


Рисунок 1.1 – Загальна архітектура повнозв'язної нейронної мережі

індивід (див. означення в переліку умовних позначень, скорочень і термінів). Індивід може бути представлений різними способами, але ми будемо розглядати представлення, яке найчастіше використовується в GP, а саме – дерево (приклад дерева – рисунок 1.2). В якості внутрішніх вузлів в дереві можуть бути функції будь якої арності, а в якості листків – ознаки (англ. features) вхідних даних, константи або змінні.

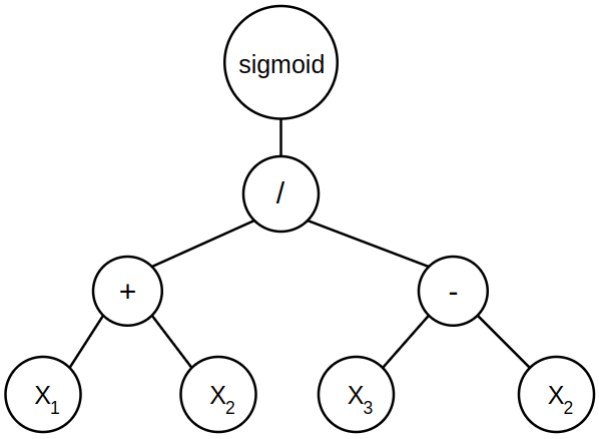


Рисунок 1.2 – Представлення індивіду у вигляді дерева, який отримує на вхід три фічі: X_1, X_2, X_3 та обраховує наступну функцію, яка залежить від цих фіч - $sigmoid\left(\frac{X_1+X_2}{X_3-X_2}\right)$, де $sigmoid(x) = \frac{1}{1+\exp(-x)}$

1.3 Метрики оцінки якості моделей та функції втрат, для задач класифікації

Існують різноманітні метрики для оцінювання якості моделі, наприклад точність (англ. accuracy) [20], precision [20], recall [28], f1-score [32]. Формально ці метрики можна записати наступним чином (таблиця 1.1).

Таблиця 1.1 – Формули основних метрик якості класифікаційних моделей

Метрика	Формула
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
F1-Score	$2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision}+\text{Recall}}$

Основна метрика, що використовується для загальної оцінки якості моделі, — це accuracy [20]. Ця метрика є найбільш інформативною, коли класи в даних розподілені рівномірно. Проте, в умовах сильного дизбалансу класів accuracy [20] може давати занадто оптимістичну картину ефективності моделі, оскільки вона враховує лише загальну кількість правильних передбачень.

Precision [20] краще використовувати, коли важливіше знизити кількість помилково позитивних результатів. Наприклад, у медичних тестах або в системах, де вартість помилки дуже висока.

Recall [28] є ключовою метрикою, коли важливо виявити всі можливі позитивні випадки. Це критично для ситуацій, де пропуск позитивних результатів може мати серйозні наслідки, наприклад, в системах раннього виявлення захворювань.

F1-score [32] використовується для оцінки балансу між precision [20]

та recall [28]. Ця метрика особливо корисна, коли потрібно врахувати обидві ці характеристики одночасно, наприклад, у контексті інформаційного пошуку та класифікації текстів, де немає явної переваги між помилково позитивними та помилково негативними результатами.

Вибір метрики для конкретної задачі залежить від самої задачі, але гарною практикою є розрахунок одразу декількох метрик, для того, щоб бачити повну картину.

Функції втрат використовуються під час навчання моделей, щоб оптимізувати параметри моделі з метою мінімізації розбіжності між прогнозованими результатами та дійсними даними. Такі функції кількісно оцінюють помилки моделі та на основі значень такої функції оновлюються параметри моделі. Функцій втрат також існує велика кількість, але ми наведемо приклад двох функцій, одна з яких використовується для бінарної класифікації – бінарна крос ентропія [30], а інша для багатокласової класифікації – крос ентропія [19]. Бінарна крос ентропія [30] та крос ентропія [19] виражаються наступними формулами:

$$\text{binary cross entropy loss} = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log(p_i) + (1-y_i)\log(1-p_i)\right) \quad (1.2)$$

$$\text{cross entropy loss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij}) \quad (1.3)$$

де N – кількість спостережень у наборі даних, M – кількість можливих класів, y_i – фактична мітка класу для і-го спостереження, y_{ij} – бінарний індикатор, який показує чи належить i-те спостереження до класу j, p_i – прогнозована ймовірність, що спостереження i належить до класу з міткою 1, p_{ij} – прогнозована ймовірність, що i-те спостереження належиить до класу j, log – натуральний логарифм.

1.4 Процес навчання моделей для задач класифікації

В цьому підрозділі ми розглянемо, як відбувається процес навчання MLP [27] та генетичного алгоритму.

Почнемо розгляд з процесу навчання MLP [27]. Перед початком навчання ініціалізуються ваги мережі. Після того, як ваги були ініціалізовані, мережі подаються на вхід дані. Перший шар нейронів розраховує вихідний сигнал, де кожен нейрон розраховує його за формулою 1.1, далі цей сигнал передається наступному шару, наступний шар розраховує вихідний сигнал, передає його далі і так це повторюється стільки разів, скільки в мережі шарів, останній шар розраховує вихідний сигнал, у випадку бінарної класифікації це буде одне значення для кожного вхідного прикладу з даних, яке буде відображати ймовірність того, що поточний приклад належить до класу 1, у випадку багатокласової класифікації, для кожного вхідного прикладу будуть розраховуватись n - значень, де n – це кількість можливих класів, які будуть відображати ймовірність того, що поточний приклад належить до класу i. Після того, як були розраховані ймовірності належності прикладів до класу/класів, використовуючи значення цих ймовірностей розраховуються функції втрат за формулами 1.2, 1.3 для бінарної та багатокласової класифікації відповідно. Наступний крок є ключовим у навчанні MLP [27] – зворотнє поширення помилки. Зворотнє поширення помилки полягає в обчисленні градієнтів функції втрат по відношенню до кожного вагового коефіцієнту в мережі, використовуючи правило ланцюгового диференціювання. Обчисленні градієнти використовуються для оновлення ваг у напрямку, що зменшує помилку (зазвичай за допомогою методу градієнтного спуску, або його варіантів, наприклад, Adam [15]). Описаний вище процес повторюється ітеративно певну кількість ітерацій, або поки не буде виконана умова завершення.

Тепер розглянемо процес навчання генетичного алгоритму. Першим

кроком ініціалізується популяція (див. означення в переліку умовних позначень, скорочень і термінів) індивідів. Ініціалізація індивідів може відбуватися як повністю випадковим чином, так і з наперед заданими конкретними структурами індивідів. Для кожного індивіду в популяції розраховується фітнес-функція (див. означення в переліку умовних позначень, скорочень і термінів), яка вимірює як добре особина вирішує поставлену задачу. Для задач бінарної та багатокласової класифікації в якості фітнес функції використовуються формули 1.2 та 1.3, в цьому випадку, чим менше буде значення фітнес-функцій тим краще індивід буде пристосований до поточної задачі. Після того, як для кожного індивіду були розраховані фітнес-функції, за допомогою методу відбору обираються індивіди, які будуть брати участь у створені наступного покоління. Існує багато методів відбору, ось декілька прикладів: рулетковий відбір [16], турнірний відбір [7] (випадковим чином обирається групка індивідів з усієї популяції і з цієї групки для репродукції обирається той індивід у якого найкраща фітнес функція), стабільний відбір (англ. Steady State Selection) [6]. Далі для генерації наступного покоління, до відібраних індивідів застосовується операція кросинговеру (див. означення в переліку умовних позначень, скорочень і термінів). Обрані індивіди розбиваються на пари і обмінюютьсся частинами своїх хромосом для створення нових індивідів. Цей процес є стохастичним, тобто які саме частини хромосом будуть обмінюватись визначається випадковим чином. До поширених методів кросинговеру відносяться одноточковий кросинговер [26] та багатоточковий кросинговер [34]. При одноточковому кросинговері [26] геноми обох батьків розділяються в одній випадково обраній точці, а потім їх сегменти обмінюються місцями. Багатоточковий кросинговер [34] включає декілька таких точок, що дозволяє формувати потомство з ще більшою генетичною різноманітністю. Після кросинговеру, гени нових особин можуть випадково змінюватись з певною, зазвичай низькою, ймовірністю – цей процес називається мутацією (див. означення в переліку умовних

позначень, скорочень і термінів). Мутація запобігає можливій стагнації всієї популяції в локальному оптимумі, вносячи нові варіанти в генетичний матеріал. Створені особини заміщують деякі, або всі особини в поточній популяції, в залежності від методу відбору. Описаний вище процес повторюється певну кількість поколінь, або поки не буде досягнуто задовільне значення фітнес-функції.

Важливим кроком під час навчання моделей є розділення даних на тренувальну та тестувальну вибірки. Тренувальна вибірка використовується для оновлення ваг моделі, у випадку MLP [27], та генерацію нових поколінь, у випадку генетичного алгоритму. Градієнти, у випадку MLP [27], розраховуються використовуючи значення функції втрат, яка була отримана в результаті роботи MLP [27], який отримав на вході тренувальну вибірку. Фітнес-функції для індивідів, у випадку генетичного алгоритму, також розраховуються використовуючи тільки тренувальну вибірку. Тестувальна вибірка використовується для оцінки якості моделі під час навчання, для того, щоб можна було відслідковувати в який момент почнеться перенавчання (англ. overfit) і використовувати ті параметри моделі, які вона мала до початку перенавчання, це значно покращить узагальнювальну здатність моделей. Важливо зазначити, що тестувальна вибірка ніяким чином не впливає на оновлення параметрів моделей.

1.5 Огляд суміжних робіт

Як вже було описано в розділі 1.2 існують наступні методи вирішення задач класифікації: класичні статистичні методи, методи ML, методи глибинного навчання та генетичні алгоритми.

Статистичні методи добре підходять в тих випадках, коли нам важлива інтерпретованість результатів. Прикладом такого методу може бути логістична регресія [24]. Логістична регресія [24] – це статистичний метод, який використовується для задачі бінарної класифікації. Метод

базується на логістичній функції, яка оцінює ймовірності приналежності спостережень до однієї з двох категорій. Основною перевагою логістичної регресії [24] є її здатність працювати з даними, де таргетна зміна обмежена інтервалом [0,1], що робить її ідеальною для задач бінарної класифікації. Крім того, модель легко інтерпретувати, оскільки коефіцієнти моделі можуть бути представлені у вигляді шансів (англ. odds ratios). Застосування логістичної регресії [24] виявилося ефективним у багатьох областях, включаючи медицину для діагностики захворювань, в банківській справі для оцінки кредитного ризику, а також в соціальних науках для аналізу виборчих перегонів. Однією з найважливіших областей застосування статистичних методів у медицині є прогнозування серцево-судинних захворювань. Логістична регресія [24] часто використовується для аналізу ймовірності розвитку цих захворювань на основі різних ризикових факторів, таких як вік, кров'яний тиск, холестерин, куріння, сімейний анамнез та інші. В роботі Hosmer і Lemeshow [13], метод логістичної регресії [24] було застосовано для визначення ймовірності настання серцевих нападів у пацієнтів на основі їхнього медичного анамнезу. Модель включала незалежні змінні, які були вибрані на основі клінічного досвіду та попередніх досліджень. Кожен з цих факторів був оцінений на його зв'язок з ризиком розвитку хвороби, і коефіцієнти моделі були інтерпретовані через шансові співвідношення, що дозволило медичним працівникам краще розуміти ризики. Зокрема, було встановлено, що високий кров'яний тиск та високий рівень холестерину значно підвищують шанси на розвиток серцевих захворювань, в той час як регулярні фізичні вправи та здоровий раціон харчування зменшують ці шанси. Ці висновки допомагають лікарям формулювати профілактичні рекомендації та лікувальні стратегії для пацієнтів з підвищеним ризиком. Такий підхід підкреслює значення логістичної регресії [24] не тільки як аналітичного інструменту, але й як засобу для підтримки клінічних рішень у медицині.

Методи ML, такі як k-найближчих сусідів [9] (англ. k-Nearest

Neighbors, knn), залишаються одними з найпопулярніших через їхню простоту та ефективність у багатьох випадках. У статті Guo і співавторів [10] досліджено модифікований підхід до методу knn [9], який використовується для класифікації даних у складних застосуваннях, таких як веб-майнінг. У цій статті автори фокусуються на застосуванні цього методу для класифікації великих наборів даних, де традиційні методи knn [9] часто зазнають труднощів через велику обчислювальну складність і залежність від вибору оптимального значення параметра k. Автори пропонують новий метод – kNNModel [10], який автоматизує вибір k і використовує передбачувальну модель для підвищення ефективності класифікації. Цей підхід передбачає попереднє моделювання даних за допомогою визначення представників кожної класифікаційної категорії на основі тренувального набору даних. Представники, визначені методом, є центрами кластерів, що представляють групи схожих за характеристиками екземплярів. Свій підхід автори тестують на даних з репозиторію UCI. Використовуючи kNNModel [10], автори провели експерименти, які показали значне покращення в точності класифікації порівняно зі стандартним методом knn [9], особливо в умовах, де потрібно ефективно обробляти великі обсяги даних. Одним з ключових результатів експерименту є те, що застосування моделі kNNModel [10] дозволило значно скоротити час обчислень, необхідний для класифікації нових екземплярів, завдяки використанню попередньо підготовлених представників замість повторного обчислення відстаней до всіх точок даних.

Дослідження Soliman та Abd-elaziem [33] розглядає використання MLP [27] для спеціалізованої задачі класифікації зірок за їхніми спектральними характеристиками. MLP [27], варіант нейронної мережі, є винятково підходящим для обробки нелінійних завдань, як-от аналіз астрофізичних даних, де потрібно розпізнавати складні взаємозв'язки між характеристиками. У цьому конкретному дослідженні використовувались дані з понад 100,000 спостережень, кожне з яких

містить 18 ознак, таких як інтенсивність на різних довжинах хвиль. Ці особливості були використані для тренування MLP [27] з метою класифікації об'єктів на галактики та зорі. MLP [27], яке було застосоване в дослідженні, містило кілька прихованих шарів, що дозволяло моделі ефективно вивчити складні патерни у даних. Ефективність класифікації, яку продемонструвала модель, склала 97%. Такий високий показник точності підкреслює здатність MLP [27] ефективно обробляти великі обсяги складних даних та виділяти критично важливі особливості для розпізнавання патернів. Для оптимізації процесу тренування та досягнення максимальної точності було випробувано декілька оптимізаторів, серед яких Adagrad [5] показав найкращі результати з найвищою валідаційною точністю. Ці результати не тільки демонструють потенціал MLP [27] для вирішення астрофізичних задач класифікації, але й вказують на можливість його застосування в інших сферах, де потрібне швидке та ефективне рішення аналогічних задач. Завдяки такому дослідженню, можливо підвищити ефективність використання астрономічних даних та покращити розуміння структури та еволюції космосу.

Дослідження Robu та Holban [29] демонструє застосування генетичних алгоритмів у завданнях класифікації, де використовуються класичні набори даних: Car, Zoo та Mushrooms. В рамках цього дослідження автори впровадили новий підхід до фітнес-функції, який враховує точність прогнозування та інтерпретованість правил. Фітнес-функція, запропонована в їхній роботі, включає в себе вагові коєфіцієнти, які дозволяють регулювати значимість точності прогнозування та інтерпретованості правиил. Це важливо, оскільки в генетичних алгоритмах не тільки важлива здатність правил точно класифікувати дані, але й можливість інтерпретувати ці правила. Такий підхід дозволяє створювати правила, які не тільки ефективні, але й інтерпретовані, що є критично важливим для застосувань, де необхідно пояснення моделі, наприклад, в медичних діагностиках чи у фінансовому

секторі. Експериментальні результати, представлені в дослідженні, показали, що генетичні алгоритми можуть бути порівнянно ефективними з традиційними методами ML, такими як Наївний Баєс [37] та J48 [2], які також були застосовані до тих же даних. Це свідчить про великий потенціал генетичних алгоритмів в завданнях класифікації, особливо коли необхідно знайти баланс між точністю та інтерпретованістю результатів.

Висновки до розділу 1

В цьому розділі було розглянуто теоретичні відомості про об'єкт та предмет дослідження, а саме про задачі бінарної та багатокласової класифікації та існуючі методи вирішення цих задач. Було здійснено короткиий огляд суміжних розділів, таких як класичні статистичні методи, методи ML, глибинне навчання та генетитичні алгоритми. Було оглянуто процеси навчання моделей, які вирішують задачі класифікації та метрики, що оцінюють якість роботи моделей.

Ми також вказали на важливості використання методів, які легко інтерпретувати та контролювати для сфер, де пояснення моделі є критично важливим.

2 ПІДГОТОВКА ДО ПРОВЕДЕННЯ ДОСЛІДЖЕННЯ

В даному розділі знаходиться огляд основних інструментів та методів аналізу та попередньої обробки даних, також ми зазначимо використані інструменти та ресурси для моделювання.

2.1 Використані інструменти та ресурси

В якості мови програмування було вибрано Python v3.11 [36], це ефективна та гнучка мова програмування, для розв'язання задач ML, для якої створено велику кількість бібіліотек та ресурсів, які дозволяють ефективно розв'язувати задачі, включаючи задачі бінарної та багатокласової класифікації табличних даних та картинок. Основними бібліотеками для створення моделей були бібліотеки Deap v1.4 [8] та scikit-learn v1.4 [23]. Обидві бібліотеки надають документацію, невеликі навчальні посібники та приклади для пришвидшення побудови моделей.

Бібліотека Deap [8] — це спеціалізована бібліотека для створення EA. Ця бібліотека має реалізовані рішення для різних задач, таких як GP, еволюційні стратегії, генетичні алгоритми та багато інших. Вона забезпечує зручний інтерфейс для налаштування та запуску еволюційних експериментів, надаючи широкий набір інструментів для маніпуляції популяціями, відбору, кросинговеру та мутацій. Основними елементами бібліотеки Deap [8] є індивідуми, популяції, фітнес-функції, оператори генетичних алгоритмів, такі як, відбір, кросинговер та мутація. Ця бібліотека також дозволяє налаштовувати багато параметрів, таких як розмір популяції, кількість поколінь, ймовірності мутацій та кросинговеру, що робить її дуже гнучкою для різних задач. Вона підтримує паралельні обчислення, що значно прискорює процес еволюційного пошуку оптимальних рішень. В даному дослідженні буде

використовуватись бібліотека Deap [8] для реалізації $(1 + \lambda)$-EA with GP encodings, що дозволяє досліджувати ефективність та керованість цього алгоритму в контексті задачі класифікації. Зокрема, ми будемо використовувати такі оператори, як турнірний відбір та одноточкову мутацію. Крім того, буде проведено аналіз впливу різних гіперпараметрів, таких як, значення λ та глибина дерева, а також множини термінальних та внутрішніх вузлів, на якість розв'язків та швидкість конвергенції алгоритму.

Бібліотека scikit-learn [23] — це популярна бібліотека для ML, яка надає великий набір інструментів для задач класифікації, регресії, кластеризації, зниження розмірності та попередньої обробки даних. Вона забезпечує простий і уніфікований інтерфейс для побудови та оцінки моделей ML, що дозволяє швидко розробляти і тестувати різні алгоритми. Основні компоненти бібліотеки scikit-learn [23] включають реалізовані алгоритми для класифікації, регресії, кластеризації та зниження розмірності, а також методи для попередньої обробки даних. В даному дослідженні бібліотека scikit-learn [23] буде використовуватись для підготовки даних, вибору ознак, побудови та оцінки моделей класифікації. Зокрема, ми будемо використовувати стандартні підходи до попередньої обробки даних, такі як масштабування ознак, зниження розмірності та розділення даних на тренувальну та тестову вибірки. Побудова моделей буде здійснюватись з використанням алгоритму MLP [27]. Результати класифікації будуть оцінюватись за допомогою метрик, таких як accuracy [20], precision [20], recall [28] та f1-score [32]. Це дозволить порівняти ефективність різних підходів та обрати найкращий алгоритм для задачі класифікації.

Також були використані наступні бібліотеки: pandas [35] – для завантаження та попередньої обробки даних, optuna [1] – для оптимізації гіперпараметрів моделей, torch [22] та torchvision [18] – для обробки картинок та створення ембідінгів з моделей.

Проаналізувавши різноманітні сервіси, які надають доступ до

даних, в якості вебресурсу з даними ми використовуємо вебсайт https://www.kaggle.com/datasets. Kaggle – це платформа для змагань з ML, яка також надає великий каталог відкритих наборів даних для різноманітних задач, включаючи класифікацію, регресію та кластеризацію. Набори даних на kaggle часто добре документовані та попередньо оброблені, що дозволяє швидко приступити до експериментів.

2.2 Опис використаних наборів даних

В даній роботі використовувалися наступні набори даниих:

– Pima Indians Diabetes Database [25] – це набір даних, який використовується для задач бінарної класифікації табличних даних в області біомедичних досліджень. Цей датасет був зібраний Національним інститутом діабету, шлункових і ниркових захворювань США. Набір даних містить інформацію про жінок з племені Піма, що проживають в Арізоні, та включає показники здоров'я, які можуть впливати на розвиток діабету. Датасет складається з 768 зразків, кожен з яких має 8 вхідних ознак і два вихідних класи, які вказують на наявність або відсутність діабету. Всі ознаки числові, що дозволяє легко використовувати їх у ML. Датасет складається з наступних ознак: Pregnancies – кількість вагітностей у жінки; Glucose – рівень глюкози у плазмі крові через 2 години після навантажувального тесту; Blood Pressure – діастолічний артеріальний тиск; Skin Thickness – товщина шкірної складки трицепса; Insulin – рівень інсуліну у сироватці крові; BMI – індекс маси тіла; Diabetes Pedigree Function – функція родоводу діабету (враховує генетичну спадковість); Age – вік пацієнта; цільова змінна: Outcome – наявність діабету (0 - відсутній, 1 - наявний). Цей датасет має збалансовані класи.

– Human Activity Recognition with Smartphones [3] – це набір даних, який використовується для задач багатокласової класифікації табличних даних в області розпізнавання людської діяльності. Цей датасет був

зібраний з допомогою вбудованих акселерометрів та гіроскопів смартфонів, що носили на поясі 30 учасників. Дані записувалися під час виконання різних фізичних активностей, включаючи ходьбу, підйом та спуск по сходах, сидіння, стояння та лежання. Датасет складається з 10 299 зразків, кожен з яких має 562 вхідні ознаки, що представляють різні статистичні та перетворені значення з сигналів акселерометра і гіроскопа, такі як середнє значення, стандартне відхилення, максимальні та мінімальні значення, а також частотні перетворення. Всі ознаки числові. Датасет складається з наступних ознак: Body Acceleration – лінійне прискорення тіла в осях X, Y та Z; Total Acceleration – загальне прискорення тіла в осях X, Y та Z; Body Gyroscope – кутова швидкість тіла в осях X, Y та Z; Jerk Signals – похідні лінійного прискорення та кутової швидкості; Magnitude of these three-dimensional signals – величина сигналів прискорення та гіроскопа; Frequency domain features – перетворені у частотну область сигнали за допомогою швидкого перетворення Фур'є; цільова змінна: Activity – тип фізичної активності, виконуваної учасником (наприклад, walking, walking upstairs, walking downstairs, sitting, standing, laying). Цей датасет є збалансованим і добре підходить для задач класифікації, оскільки містить різноманітні фізичні активності.

– Chest X-Ray Images (Pneumonia) [14] – це набір даних, який використовується для задач бінарної та багатокласової класифікації зображень у медичних дослідженнях. Цей датасет містить рентгенівські знімки грудної клітки пацієнтів з пневмонією (вірусною або бактеріальною) та без неї. Дані були зібрані для сприяння розвитку моделей ML, здатних автоматично виявляти пневмонію на рентгенівських знімках. Датасет складається з 5 840 зображень, які розділені на дві категорії: Train та Test. Кожна категорія містить зображення, позначені як „Pneumonia“ або „Normal,“ або якщо розглядати задачу, як багатокласову класифікацію, то „Pneumonia“ розділяється на „Virus“ та „Bacteria“. Цей датасет надає великі можливості для досліджень у сфері

медичної діагностики за допомогою глибокого навчання, дозволяючи розробляти моделі, які можуть автоматично ідентифікувати захворювання на основі рентгенівських знімків.

Ці три набори даних представляють різноманітні задачі класифікації, включаючи бінарну класифікацію табличних даних, багатокласову класифікацію табличних даних, бінарну класифікацію зображень та багатокласову класифікацію зображень. Це дозволяє комплексно оцінити ефективність методів і моделей у різних доменах застосування ML.

2.3 Вплив Expressive Encodings на ефективність класифікаційних алгоритмів

У цьому розділі ми розглянемо вплив виразних кодувань на ефективність класифікаційних алгоритмів. Основою для даного аналізу є стаття „Simple Genetic Operators are Universal Approximators of Probability Distributions (and other Advantages of Expressive Encodings)“ авторів Elliot Meyerson, Xin Qiu та Risto Miikkulainen [21], яка досліджує можливості генетичних алгоритмів завдяки їхнім expressive encodings (за формальним означенням можна звернутися до [21, Definition 1, стор. 3]). В цій статті описуються переваги використання expressive encodings у EA. Головна ідея полягає в тому, що такі encodings дозволяють EA ефективно знаходити рішення у складних середовищах, дозволяючи простим генетичним операторам [21, підрозділ 2.2.3, стор. 3] апроксимувати будь-який розподіл ймовірностей фенотипів потомства [21, розділ 2.1, стор. 2]. До переваг expressive encodings також відносяться: прискорення конвергенції, тобто такі encodings можуть забезпечити надекспоненційне прискорення конвергенції, а також зменшення потреби в ручному налаштуванні, завдяки expressive encodings немає потреби у складному ручному налаштуванні операторів для кожної окремої задачі. В даній

статті наводяться теореми 4.2 [21, Theorem 4.2] та 4.4 [21, Theorem 4.4], в яких йдеться про те, що GP є expressive encodings для одноточкової мутації та що MLP з сигмоїдною функцією активації також є expressive encodings для одноточкової мутації. Автори статті розглядають алгоритм $(1 + \lambda)$-EA with GP encodings (псевдокод алгоритму можна знайти [21, Appendix B, стор. 12]) для вирішення проблем DFC [21, Problem 1, стор. 6], RFC [21, Problem 2, стор. 7] та LBAP [21, Problem 3, стор. 7]. Саме цим дослідженням ми надихнулись та порівняли методи MLP with gradient descent, MLP with single-point mutation та $(1 + \lambda)$-EA with GP encodings для задач бінарної та багатокласової класифікації табличних даних та картинок.

Висновки до розділу 2

У цьому розділі було детально описано інструменти та ресурси, які були використані для підготовки та проведення дослідження. Розглянуто основні бібліотеки Python, які використовувалися для моделювання та аналізу даних, а також наведено опис трьох наборів даних, які були використані для задач класифікації. Було обрано Python v3.11 як мову програмування завдяки її ефективності та гнучкості. Основними бібліотеками, які використовувалися для створення моделей, були Deap v1.4 та scikit-learn v1.4. Бібліотека Deap забезпечує зручний інтерфейс для налаштування та запуску еволюційних експериментів, що дозволяє реалізувати EA, такі як $(1 + \lambda)$-EA with GP encodings. Бібліотека scikit-learn надає великий набір інструментів для попередньої обробки даних, побудови та оцінки моделей класифікації. Також використовувалися бібліотеки pandas, optuna, torch та torchvision для завантаження, обробки та оптимізації даних. Були використані три різні набори даних, які представляють різноманітні задачі класифікації: Pima Indians Diabetes Database – набір даних для бінарної класифікації табличних даних в області біомедичних досліджень; Human Activity

Recognition with Smartphones – набір даних для багатокласової класифікації табличних даних, зібраний за допомогою акселерометрів та гіроскопів смартфонів; Chest X-Ray Images (Pneumonia) – набір даних для бінарної та багатокласової класифікації зображень у медичних дослідженнях. Було розглянуто вплив expressive encodings на ефективність класифікаційних алгоритмів. Це дозволить здійснити порівняння та оцінку ефективності різних підходів у наступному розділі роботи.

3 ПОПЕРЕДНЯ ОБРОБКА ДАНИХ, ПОБУДОВА МОДЕЛЕЙ ТА ОЦІНКА МЕТОДІВ

В даному розділі описано попередню обробку даних, які використовувалися в дослідженні, та описано параметри побудованих моделей. Також описано процес навчання та підбору гіперпараметрів моделей. Вказана перевірка моделей на тестових даних, за допомогою метрик: accuracy, precision, recall та f1-score.

3.1 Попередня обробка даних

Як вже зазначалось в даній роботі використовується три набори даних, а саме: Pima Indians Diabetes Database [25], Human Activity Recognition with Smartphones [3] та Chest X-Ray Images (Pneumonia) [14]. Для кожного набору даних була застосована своя попередня обробка. Далі ми наведемо, які методи обробки були застосовані до кожного набору.

Почнемо розгляд з Pima Indians Diabetes Database (прочитати детальніше, про датасет можна у [25] або у розділі 2.2). Першим кроком попередньої обробки була заміна нульових значень, які зустрічаються у деяких змінних (Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age) на відсутні значення (nan). Це дозволяє уникнути впливу невірних даних на подальший аналіз. Далі, для кожної змінної, у якої були відсутні значення, було обчислено медіанні значення для груп з позитивним та негативним результатом по діабету (Outcome). Відсутні значення заповнювалися відповідно до медіанної величини для відповідної групи. Після цього було проведено обробку змінної Insulin для видалення викидів. Викиди визначалися за допомогою методу Interquartile Range Technique [11]. Наступним кроком

було виявлення та видалення викидів за допомогою методу Local Outlier Factor [4]. Цей метод використовує локальну щільність сусідів для визначення аномалій. Після розрахунку негативного фактору аномалії для кожного зразка, було визначено порогове значення, і видалено ті зразки, які мали значення нижче цього порогу. Після видалення аномалій дані були розділені на ознаки та мітки. Вибірки було поділено на тренувальний та тестовий набори даних у пропорції 80:20. Потім, для тренувального та тестового наборів даних було проведено стандартизацію ознак шляхом видалення середнього значення та масштабування до одиничної дисперсії. В результаті ми отримали оброблений датасет, який будемо використовувати для порівняння моделей для задачі бінарної класифікації табличних даних.

Наступну попередню обробку опишемо для набору даних Human Activity Recognition with Smartphones (прочитати детальніше, про датасет можна у [3] або у розділі 2.2). Для цього набору даних було застосовано наступні методи попередньої обробки. По-перше, дані було розділено на тренувальну та тестову вибірки, кожна з яких містила відповідні дані для тренування та тестування моделей. Після завантаження датасетів для тренування та тестування, було застосовано LabelEncoder для кодування міток активностей (Activity) у числовий формат. Наступним кроком було розділення даних на ознаки та мітки для тренувальних і тестових вибірок. Далі було проведено випадкове перемішування тренувальних та тестових даних для уникнення впливу можливого порядку даних на результати навчання моделей. Для покращення роботи моделей було проведено стандартизацію ознак шляхом видалення середнього значення та масштабування до одиничної дисперсії. Цей процес дозволяє моделі краще адаптуватися до даних, які мають різний масштаб. В результаті попередньої обробки ми отримали стандартизовані та перемішані тренувальні та тестові набори даних, готові до подальшого використання у моделюванні.

Нарешті, розглянемо попередню обробку даних для набору даних

Chest X-Ray Images (Pneumonia) (прочитати детальніше про датасет можна у [14] або у розділі 2.2). Для цього набору даних було застосовано наступні методи попередньої обробки. По-перше, дані були завантажені та організовані у вигляді класів, де кожне зображення має відповідну мітку (0 - нормальний, 1 - бактеріальна пневмонія, 2 - вірусна пневмонія). Як зазначалося ми використали цей датасет для бінарної та багатокласової класифікації, для бінарної, відповідно, класи бактеріальна та вірусна пневмонії були об'єднані в один клас – пневмонія, а для багатокласової використовувалися класи нормальний, бактеріальна пневмонія та вірусна пневмонія. Зображення були перетворені до розміру 224x224 пікселів і конвертовані в градації сірого для уніфікації формату. Для екстракції ознак було використано попередньо натреновану модель ResNet-50 [12] без останнього повнозв'язного шару. Модель була завантажена з збережених ваг (які ми самі натренували використовуючи цей же набір даних) та використана для отримання векторів ознак зображень. Далі, для обробки отриманих векторів ознак, було застосовано стандартизацію ознак шляхом видалення середнього значення та масштабування до одиничної дисперсії. Для зменшення розмірності та збереження 99% варіативності даних було застосовано метод Principal Component Analysis [17]. В результаті ми отримали оброблені вектори ознак, готові до подальшого використання у моделюванні.

3.2 Детальний опис моделей та підбір гіперпараметрів моделей

Як вже було зазначено в даній роботі ми сфокусувалися на тестуванні моделей MLP with gradient descent, MLP with single-point mutation та $(1 + \lambda)$-EA with GP encodings. Принцип тренування MLP with gradient descent ми описали в розділі 1.4, тому в цьому розділі ми опишемо детально процес навчання моделей MLP with single-point

mutation та $(1 + \lambda)$-EA with GP encodings.

MLP with single-point mutation використовує оптимізаційний алгоритм, який працює наступним чином: на кожній епосі випадковим чином обирається значення однієї ваги з усієї нейронної мережі та до нього додається значення випадкової величини, яка має нормальний розподіл з нульовим математичним сподіванням та певним значенням дисперсії, після цього розраховується функція втрат з оновленими вагами, якщо її значення стало менше, ніж було з початковими вагами на поточній епосі, то ваги зберігаються і процес переходить на наступну епоху, якщо функція втрат збільшилася, то повертаються ваги, які були до додавання значення випадкової величини і відбувається перехід на наступну епоху. Цей процес повторюється задану кількість епох.

Метод $(1 + \lambda)$-EA with GP encodings описується наступним чином. Першим кроком ініціалізується індивід, в нашому випадку індивідом є дерево у якого в якості внутрішніх вузлів – функції, які мають арність 2, а в якості листків – features датасету або константи з набору: $1, 0, -1, e, \pi$. Для цього індивіду розраховується фітнес-функція. Після того, як була розрахована фітнес-функція для створеного індивіду, він мутує λ разів, таким чином ми отримуємо λ нових індивідів. Мутація в даному випадку відбувається наступним чином: випадково обирається один вузол з усього індивіду і його значення замінюється на якесь інше випадкове, валідне значення (у випадку внутрішніх вузлів – значення замінюється на якусь іншу функцію, а у випадку листків на якусь іншу feature, або константу). Після цього ми розраховуємо фітнес-функції для усіх новостворених індивідів і обираємо індивід, який має найменше значення фітнес-функції. Цей індивід далі виступає в якості батька на наступних ітераціях.

Тепер після того, як ми описали, як в нашому випадку працюють алгоритми, перейдемо до процесу підбору оптимальних гіперпараметрів. Процес вибору гіперпараметрів є важливим кроком в тренуванні моделей, оскільки від них значно залежить швидкість конвергенції та якість моделей. Тому в цій частині ми опишемо, процес за яким відбувається

підбір гіперпараметрів для різних задач та моделей, а також які саме гіперпараметри виявилися найкращими і які ми використовуємо.

Почнемо розгляд з задач класифікації та моделі MLP with gradient descent. Для підбору гіперпараметрів для цієї моделі ми використовували бібліотеку optuna [1], яка дозволяє проводити байєсівську оптимізацію [31]. Для цього ми задали простір гіперпараметрів по якому проводився пошук оптимальних з них за 1000 ітерацій. Найкращі гіперпараметри для кожної задачі можна подивитися у таблиці 3.1.

Таблиця 3.1 – Найкращі гіперпараметри для моделі MLP with gradient descent

Гіперпараметри	Задачі			
	Бінарна класифікація табличних даних	Бінарна класифікація картинок	Багатокласова класифікація табличних даних	Багатокласова класифікація картинок
hidden_layer_sizes	(10, 15, 10)	(15, 20, 15)	(10, 10)	(10, 10)
activation	tanh	tanh	logistic	tanh
solver	sgd	sgd	sgd	sgd
alpha	0.0009	0.0054	0.0001	0.0001
learning_rate_init	0.004	0.002	0.008	0.001
learning_rate	adaptive	invscaling	adaptive	invscaling
batch_size	32	256	64	128
tol	0.00002	0.00033	0.00023	0.00003

Для моделі MLP with single-point mutation для пошуку оптимальних гіперпараметрів також було застосовану байєсівську оптимізацію [31] за 1000 ітерацій. В результаті ми отримали оптимальні гіперпараметри, які можна подивитися у таблиці 3.2.

Останньою моделлю для якої ми шукали оптимальні параметри є $(1 + \lambda)$-EA with GP encoding. Процес пошуку такий же самий як і для вище наведених моделей. Відповідні результати наводяться у таблиці 3.3.

Таким чином після того, як ми отримали оптимальні гіперпараметри для усіх моделей можна переходити до процесу тренування та аналізу результатів.

Таблиця 3.2 – Найкращі гіперпараметри для моделі MLP with single-point mutation

Гіперпараметри	Задачі			
	Бінарна класифікація табличних даних	Бінарна класифікація картинок	Багатокласова класифікація табличних даних	Багатокласова класифікація картинок
hidden_layer_sizes	(10, 15, 20, 15, 10)	(15, 20, 15)	()	(10, 10)
scale_for_mutation	0.5	0.1	0.1	0.1

Таблиця 3.3 – Найкращі гіперпараметри для моделі $(1+\lambda)$-EA with GP encoding

Гіперпараметри	Задачі			
	Бінарна класифікація табличних даних	Бінарна класифікація картинок	Багатокласова класифікація табличних даних	Багатокласова класифікація картинок
tree_depth	3	6	6	8
λ	3	5	3	5

3.3 Навчання моделей та порівняльний аналіз результатів

Для тренування моделей ми використовуємо функції 1.2 та 1.3 в якості функцій втрат для MLP та фітнес-функцій для $(1+\lambda)$-EA with GP encoding для бінарної та багатокласової класифікацій відповідно. Таким чином наші моделі вчаться мінімізувати ці функції, оскільки, як можна бачити, чим менше значення цих функцій тим більш правильний результат. Справді, підставивши в ці функції в якості y_i – 1 та в якості ймовірності p_i також 1 (тобто це той випадок, коли справжня мітка для поточного прикладу – 1 і модель на виході дає ймовірність того, що поточний приклад належить до класу 1 також 1), отримаємо: $1\log(1) + (1-1)\log(1-1) = 0$, а якщо підставити в якості y_i – 0, а в якості ймовірності p_i також 0 (тобто це той випадок, коли справжня мітка для поточного прикладу – 0 і модель на виході дає ймовірність того, що поточний приклад належить до класу 1 також 0), отримаємо:

$0 \log(0) + (1 - 0) \log(1 - 0) = 0$, що показує, що якщо модель правильно передбачила результат для прикладу, то значення функції втрат дорівнює 0. Для функції втрат для багатокласової класифікації подібна підстановка тільки з кількістю класів більше 2 також покаже, що функція втрат буде дорівнювати 0. Ще раз підсумовуючи, чим ближче функція втрат до 0, тим більш правильні передбачення робить модель, тобто під час навчання моделей стоїть задача саме мінімізувати функції втрат.

Для оцінки якості моделей ми використовували метрики зазначені в таблиці 1.1. Опис які метрики в яких випадках краще використовувати можна прочитати в розділі 1.3. Зазначимо, як видно з формул цих метрик чим ближче кожна з них до 1, тим більш якісніша модель. У випадку accuracy, якщо частина доданку в знаменнику, а саме $FP + FN$ буде дорівнювати 0, тобто наша модель не дасть жодного неправильного передбачення, то чисельник і знаменник будуть дорівнювати один одному, а отже значення accuracy буде 1, у випадку precision та recall, якщо частини доданків в знаменнику, а саме FP та FN будуть дорівнювати 0, тобто наша модель не дасть жодного неправильного результату, то чисельники та знаменники відповідних формул будуть рівні між собою, а отже і значення цих метрик буде дорівнювати 1 і остання метрика f1-score, підставивши в цю формулу $2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$ найкращі показники для precision та recall, а саме 1 та 1 отримаємо $2 \times \frac{1 \times 1}{1+1}$, що дорівнює 1.

Володіючи детальною інформацією про метрики та функції втрат перейдемо до тренування моделей. Почнемо розгляд з задачі бінарної класифікації табличних даних, як вже зазначалося в розділі 2.2 для цього ми використовували датасет Pima Indians Diabetes Database [25]. Оптимальні гіперпараметри для усіх трьох моделей можна знайти у попередньому розділі. Зазначимо, що MLP with gradient descent, MLP with single-point mutation та $(1 + \lambda)$-EA with GP encodings тренувалися протягом 150, 6000 та 50 епох відповідно. Після тренування ми отримали результати для моделі MLP with gradient descent, які можна подивитися в таблиці 3.4, для моделі MLP with single-point mutation – у таблиці 3.5, для

моделі $(1 + \lambda)$-EA with GP encodings – у таблиці 3.6. Результуючі метрики на найкращій ітерації для усіх трьох моделей знаходяться в таблиці 3.7. Графіки зміни функцій втрат для кожної моделі можна знайти на рисунку 3.1

Таблиця 3.4 – Результати моделі MLP with gradient descent для задачі бінарної класифікації табличних даних

Номер епохи	Час тренування, секунди	Функція втрат для тренувальної вибірки	Функція втрат для тестувальної вибірки
50	0.2187	0.3375	0.3399
100	0.4371	0.2797	0.3222
139	0.611	0.2517	0.3201
150	0.6601	0.246	0.3205

Таблиця 3.5 – Результати моделі MLP with single-point mutation для задачі бінарної класифікації табличних даних

Номер епохи	Час тренування, секунди	Функція втрат для тренувальної вибірки	Функція втрат для тестувальної вибірки
1000	1.6075	0.6145	0.6068
2000	3.1348	0.4003	0.3845
3000	4.6372	0.3476	0.37
4000	6.1426	0.3096	0.3476
5000	7.6502	0.2899	0.3331
5377	8.2214	0.2805	0.3203
6000	9.1574	0.2712	0.337

Як видно з цих даних, усі три моделі можуть досягти приблизно однакових метрик, найкращі результати моделі MLP with gradient descent, MLP with single-point mutation та $(1 + \lambda)$-EA with GP encodings мали після 139, 5377 та 44 епох відповідно, але якщо говорити в термінах часу, то на цих даних найкраще себе показали моделі MLP with gradient descent та $(1 + \lambda)$-EA with GP encodings, час роботи, яких склав 0.6110560894012451 та 0.641812801361084 секунд відповідно, в той час, як алгоритму MLP with single-point mutation знадобилось значно більше часу, щоб зійтися до таких же метрик – 8.22137975692749 секунд.

Тепер перейдемо до задачі бінарної класифікації картинок. Як ми

Таблиця 3.6 – Результати моделі $(1 + \lambda)$-EA with GP encodings для задачі бінарної класифікації табличних даних

Номер епохи	Час тренування, секунди	Функція втрат для тренувальної вибірки	Функція втрат для тестувальної вибірки
10	0.1342	0.5604605107576973	0.5854752970381429
20	0.2827446460723877	0.4759779039581297	0.47930999487854886
30	0.4323310852050781	0.4660168717459822	0.48669901348150507
40	0.5819759368896484	0.44926223768207857	0.4670636285658903
44	0.641812801361084	0.4055500268344395	0.413087600927787
50	0.7325775623321533	0.4055500268344395	0.413087600927787

Таблиця 3.7 – Метрики на найкращій ітерації кожної моделі для задачі бінарної класифікації табличних даних

	MLP with gradient descent	MLP with single-point mutation	$(1 + \lambda)$-EA with GP encodings
Accuracy	0.8750	0.8618	0.8882
Precision	0.8810	0.8000	0.8148
Recall	0.7255	0.7843	0.8627
F1-score	0.7957	0.7921	0.8381

вже згадували у розділі 2.2 для цього ми використовували датасет Chest X-Ray Images (Pneumonia) [14]. Оптимальні гіперпараметри для усіх трьох моделей можна знайти у попередньому розділі. Зазначимо, що моделі тренувалися протягом 5, 5000 та 350 епох відповідно. Після тренування ми отримали результати для моделі MLP with gradient descent, які можна подивитися в таблиці 3.8, для моделі MLP with single-point mutation – у таблиці 3.9, для моделі $(1 + \lambda)$-EA with GP encodings – у таблиці 3.10. Результуючі метрики на найкращій ітерації для усіх трьох моделей знаходяться в таблиці 3.11. Графіки зміни функцій втрат для кожної моделі можна знайти на рисунку 3.2.

З цих результатів можна бачити, що усі три моделі досягають приблизно однакових метрик. Найкращі результати моделі MLP with gradient descent, MLP with single-point mutation та $(1 + \lambda)$-EA with GP encodings мали після 4, 4067 та 314 епох відповідно, але якщо проаналізувати часову складову результатів, то видно, що модель MLP with gradient descent має доволі гарні результати, а саме найкращий

Таблиця 3.8 – Результати моделі MLP with gradient descent для задачі бінарної класифікації картинок

Номер епохи	Час тренування, секунди	Функція втрат для тренувальної вибірки	Функція втрат для тестувальної вибірки
1	0.07669401168823242	0.497593407800549	0.49597089084771523
2	0.13824105262756348	0.3568302337710611	0.4084733561498103
3	0.1832282543182373	0.27670439918304984	0.3740271194609116
4	0.24207353591918945	0.22753888110518627	0.36484990066487627

Таблиця 3.9 – Результати моделі MLP with single-point mutation для задачі бінарної класифікації картинок

Номер епохи	Час тренування, секунди	Функція втрат для тренувальної вибірки	Функція втрат для тестувальної вибірки
1000	9.006126642227173	0.5257421296896034	0.6655046677021189
2000	17.133771657943726	0.43521634220254685	0.5836374769195368
3000	25.34369707107544	0.3100229401909172	0.4715569650508267
4000	33.565962076187134	0.20390489213330115	0.4422651786252664
4067	34.12131118774414	0.19762805502900416	0.41155911737659917
5000	41.51892828941345	0.1392642392080235	0.4697005443408103

показник функції втрат на тестовій вибірці до якого модель зійшлась за 0.24207353591918945 секунди, в той час, як моделі MLP with single-point mutation та $(1 + \lambda)$-EA with GP зійшлись до приблизно таких же показників за значно більший час, а саме 34.12131118774414 та 324.0474681854248 секунд відповідно. Варто зазначити, що модель MLP with gradient descent отримала такі гарні показники за доволі малу кількість ітерацій – 4, це говорить про те, що початкова ініціалізація, яку ми використовуємо добре підходить для цієї задачі, або що функція втрат гладка, що також є великим плюсом для алгоритмів з оптимізаційним алгоритмом в основі якого градієнтний спуск (детальніше про це буде далі в цьому розділі).

Тепер розглянемо задачу багатокласової класифікації табличних даних. Як вже згадувалося у розділі 2.2 для цього ми використовували датасет Human Activity Recognition with Smartphones [3]. Оптимальні гіперпараметри для усіх трьох моделей можна знайти у попередньому розділі. Зазначимо, що моделі тренувалися протягом 100, 40000 та 15000 епох відповідно. Після тренування ми отримали результати для моделі

Таблиця 3.10 – Результати моделі $(1+\lambda)$-EA with GP encodings для задачі бінарної класифікації картинок

Номер епохи	Час тренування, секунди	Функція втрат для тренувальної вибірки	Функція втрат для тестувальної вибірки
50	44.43767476081848	0.5537949254678176	0.6333528246887804
100	88.432137966156	0.4843073680614884	0.5863055797417965
150	137.01325964927673	0.46765369242436094	0.5756337873519055
200	222.5579433441162	0.4620113572761242	0.5659198633287362
250	269.51554799079895	0.454406885688853	0.5489630835809405
300	312.19648265838623	0.4472672712306635	0.5455341469156758
314	324.0474681854248	0.3729512316574252	0.5281766391124211
350	355.8637042045593	0.3133054339713204	0.5942058280567918

Таблиця 3.11 – Метрики на найкращій ітерації кожної моделі для задачі бінарної класифікації картинок

	MLP with gradient descent	MLP with single-point mutation	$(1+\lambda)$-EA with GP encodings
Accuracy	0.8702	0.8766	0.8686
Precision	0.8705	0.8903	0.8649
Recall	0.9308	0.9154	0.9359
F1-score	0.8996	0.9027	0.8990

MLP with gradient descent, які можна знайти у таблиці 3.12, для моделі MLP with single-point mutation – у таблиці 3.13, для моделі $(1+\lambda)$-EA with GP encodings – у таблиці 3.14. Результуючі метрики на найкращій ітерації для усіх трьох моделей знаходяться в таблиці 3.15. Графіки зміни функцій втрат для кожної моделі можна знайти на рисунку 3.3.

Таблиця 3.12 – Результати моделі MLP with gradient descent для задачі багатокласової класифікації табличних даних

Номер епохи	Час тренування, секунди	Функція втрат для тренувальної вибірки	Функція втрат для тестувальної вибірки
20	1.1138453483581543	1.0118081236438938	1.0082035330355594
40	2.184493064880371	0.4627890664318615	0.5151359254304455
60	3.2554190158843994	0.09009024931076086	0.19775012361327168
80	4.3574652671813965	0.047309657612472356	0.1823308544604816
100	5.6243696212768555	0.03386235980354522	0.1748647621659461

З цих результатів можна бачити, що моделі MLP with gradient

Таблиця 3.13 – Результати моделі MLP with single-point mutation для задачі багатокласової класифікації табличних даних

Номер епохи	Час тренування, секунди	Функція втрат для тренувальної вибірки	Функція втрат для тестувальної вибірки
10000	100.6342236995697	0.15087430617007383	0.2156044823936043
20000	197.79181599617004	0.0743021215597616	0.14429604026200504
30000	297.6863377094269	0.05071144863336119	0.13711871334278997
30355	301.4574043750763	0.05017607099733541	0.13309840177561694
40000	392.26202487945557	0.04274506317254896	0.13881867833488568

Таблиця 3.14 – Результати моделі $(1+\lambda)$-EA with GP encodings для задачі багатокласової класифікації табличних даних

Номер епохи	Час тренування, секунди	Функція втрат для тренувальної вибірки	Функція втрат для тестувальної вибірки
5000	36851.879930496216	0.19436272051067008	0.47455611625452243
10000	74160.39372968674	0.10905480004203517	0.4321945893354135
13151	98058.28979992867,	0.09055499503260159	0.3412774664794485
15000	112014.1368405819	0.0801461615938035	0.4050035438429106

descent та MLP with single-point mutation показали трохи кращі результати за $(1 + \lambda)$-EA with GP encodings, але метод $(1 + \lambda)$-EA with GP encodings виділився легкістю контролювання, оскільки для покращення його метрик нам потрібно просто збільшити експресивність кодувань, а саме збільшити значення гіперпараметрів tree_depth, або λ, в той час, щоб покращити значення метрик методів MLP with gradient descent та MLP with single-point mutation нам потрібно робити оптимізацію гіперпараметрів, у першому випадку нам потрібно буде підбирати параметри, які наведені у таблиці 3.1, а для другого випадку потрібно буде підбирати значення гіперпараметрів з таблиці 3.2 (більш детальніше про це буде далі в цьому розділі). Найкращі результати моделі MLP with gradient descent, MLP with single-point mutation та $(1 + \lambda)$-EA with GP encodings мали після 100, 30355 та 13151 епох відповідно, але якщо проаналізувати часову складову результатів, то видно, що модель MLP with gradient descent має значно кращі результати, ніж два інші методи, а саме найкращий показник функції втрат на тестовій вибірці до якого модель зійшлась за 5.6243696212768555 секунд, в

Таблиця 3.15 – Метрики на найкращій ітерації кожної моделі для задачі багатокласової класифікації табличних даних

	MLP with gradient descent	MLP with single-point mutation	$(1 + \lambda)$-EA with GP encodings
Accuracy	0.9389	0.9532	0.8992
Precision	0.9409	0.9540	0.9024
Recall	0.9389	0.9532	0.8992
F1-score	0.9386	0.9530	0.8994

той час, як моделі MLP with single-point mutation та $(1 + \lambda)$-EA with GP зійшлись до приблизно таких же показників за значно більший час, а саме 301.4574043750763 та 98058.28979992867 секунд відповідно. Тобто бачимо, що час, який знадобився моделі $(1 + \lambda)$-EA with GP значно більший за час двох інших моделей.

Перейдемо до розгляду останньої задачі для якої ми робили порівняння моделей, а саме багатокласової класифікації картинок. Як ми вже згадували у розділі 2.2 для цього ми використовували датасет Chest X-Ray Images (Pneumonia) [14]. Оптимальні гіперпараметри для усіх трьох моделей можна знайти у попередньому розділі. Зазначимо, що моделі тренувалися протягом 4, 4000 та 3000 епох відповідно. Після тренування ми отримали результати для моделі MLP with gradient descent, які можна знайти у таблиці 3.16, для моделі MLP with single-point mutation – у таблиці 3.17, для моделі $(1 + \lambda)$-EA with GP encodings – у таблиці 3.18. Результуючі метрики на найкращій ітерації для усіх трьох моделей знаходяться в таблиці 3.19. Графіки зміни функцій втрат для кожної моделі можна знайти на рисунку 3.4.

Таблиця 3.16 – Результати моделі MLP with gradient descent для задачі багатокласової класифікації картинок

Номер епохи	Час тренування, секунди	Функція втрат для тренувальної вибірки	Функція втрат для тестувальної вибірки
1	0.07427215576171875	0.6596709569877551	0.7961464350044815
2	0.11884450912475586	0.5825224777602218	0.7283898040355544
3	0.15764594078063965	0.5474291612309434	0.7117366958849365
4	0.19653797149658203	0.5275167286856616	0.7098814448968397

Таблиця 3.17 – Результати моделі MLP with single-point mutation для задачі багатокласової класифікації картинок

Номер епохи	Час тренування, секунди	Функція втрат для тренувальної вибірки	Функція втрат для тестувальної вибірки
1000	5.942003488540649	0.945515138222308	1.0223175800878757
2000	11.862092018127441	0.7553421211520547	0.8680202926289025
3000	17.861849069595337	0.6135833466250156	0.8390999846599778
3198	19.033700942993164,	0.5913123295800808	0.825761457470355
4000	23.806163787841797	0.5163107311385272	0.8979753230452852

Таблиця 3.18 – Результати моделі $(1+\lambda)$-EA with GP encodings для задачі багатокласової класифікації картинок

Номер епохи	Час тренування, секунди	Функція втрат для тренувальної вибірки	Функція втрат для тестувальної вибірки
1000	18508.034254550934	0.7505749322899249	0.8955381290152556
2000	36454.2142970562	0.650466259949188	0.8406074944163802
2382	43179.45660710335,	0.6230484310558458	0.8230165307547841
3000	53760.212540864944	0.5791197740770248	0.8660695333168389

Таблиця 3.19 – Метрики на найкращій ітерації кожної моделі для задачі багатокласової класифікації картинок

	MLP with gradient descent	MLP with single-point mutation	$(1+\lambda)$-EA with GP encodings
Accuracy	0.7420	0.7356	0.6955
Precision	0.7705	0.7677	0.7195
Recall	0.7420	0.7356	0.6955
F1-score	0.7356	0.7288	0.6903

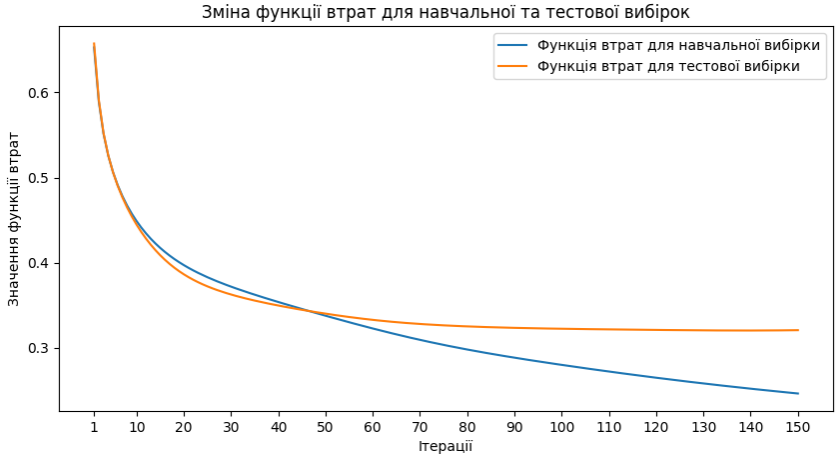
З цих результатів можна бачити, що усі три моделі досягають приблизно однакових метрик. Найкращі результати моделі MLP with gradient descent, MLP with single-point mutation та $(1 + \lambda)$-EA with GP encodings мали після 4, 3198 та 2382 епох відповідно. Проаналізувати часову складову результатів видно, що модель MLP with gradient descent має доволі гарні результати, а саме найкращий показник функції втрат на тестовій вибірці до якого модель зійшлась за 0.19653797149658203 секунд,

в той час, як моделі MLP with single-point mutation та $(1 + \lambda)$-EA with GP зійшлись до приблизно таких же показників за значно більший час, а саме 19.033700942993164 та 43179.45660710335 секунд відповідно.

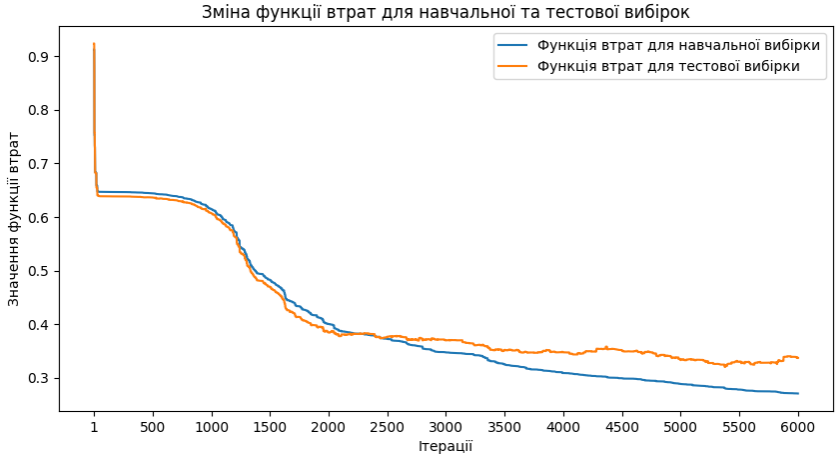
Провівши експерименти для чотирьох різних задач можна зробити висновок, що усі три методи можуть досягти однаково високих показників, або порівнюваних, але в кожного з методів є свої сильні та слабкі сторони.

Висновки до розділу 3

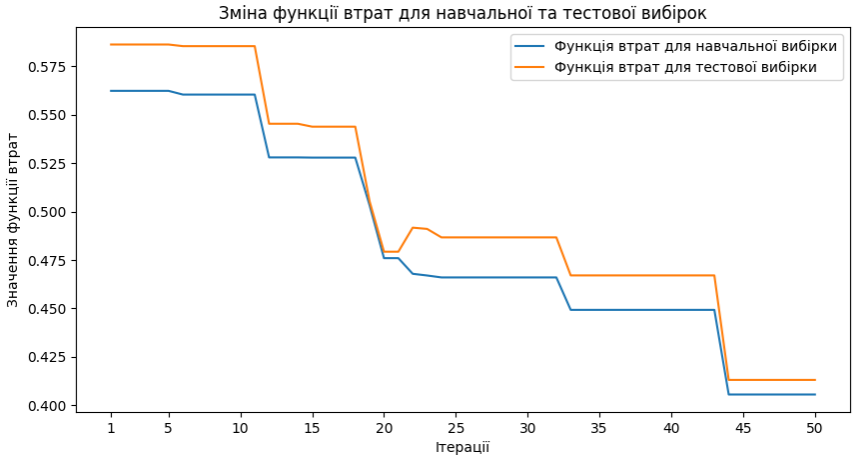
Висновки до останнього розділу є, фактично, підсумковими під усім дослідженням; однак вони повинні стостуватись саме того, що розглядалось у розділі.



(а)

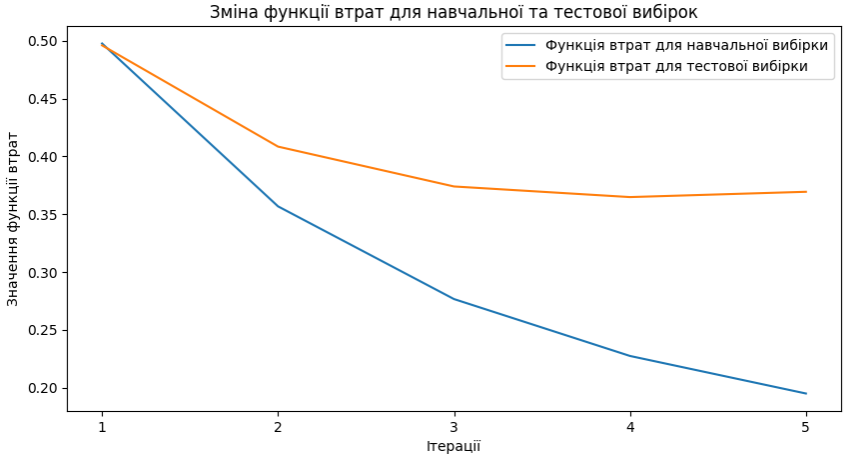


(б)

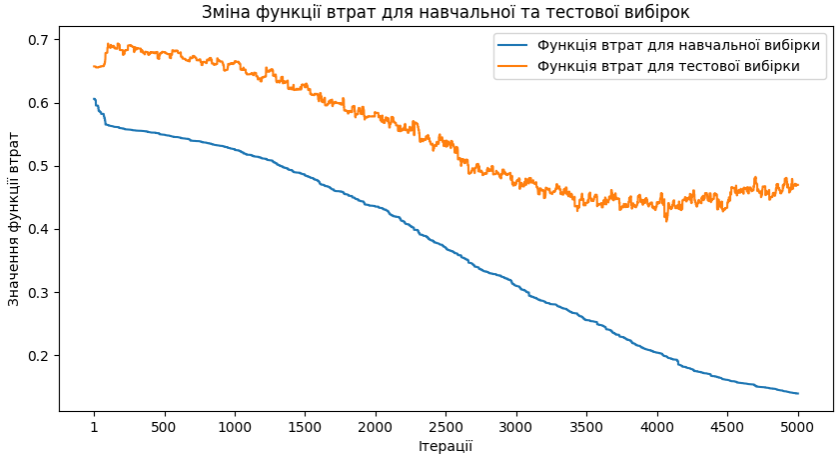


(в)

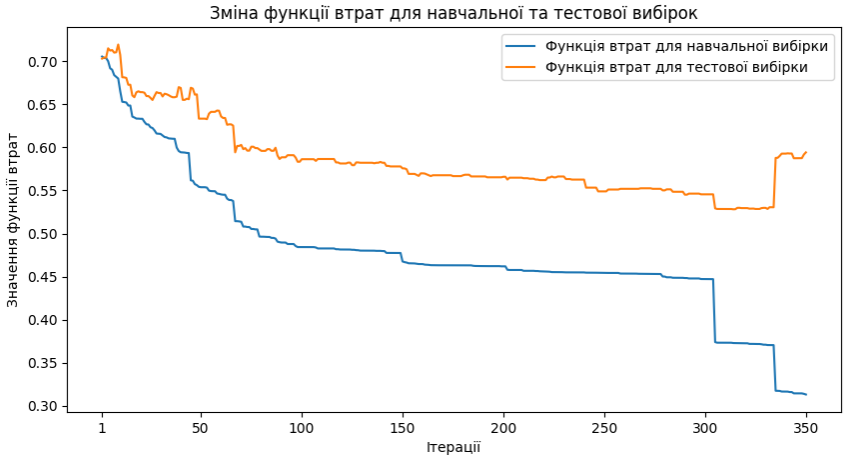
Рисунок 3.1 – Графіки залежності функцій втрат від кількості ітерацій для методів: (а) MLP with gradient descent, (б) MLP with single-point mutation, (в) $(1 + \lambda)$-EA with GP encodings, для задачі бінарної класифікації табличних даних



(а)

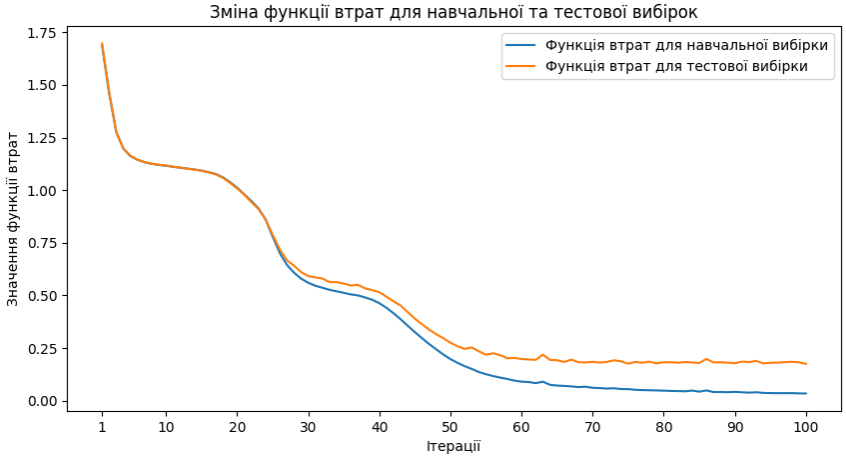


(б)

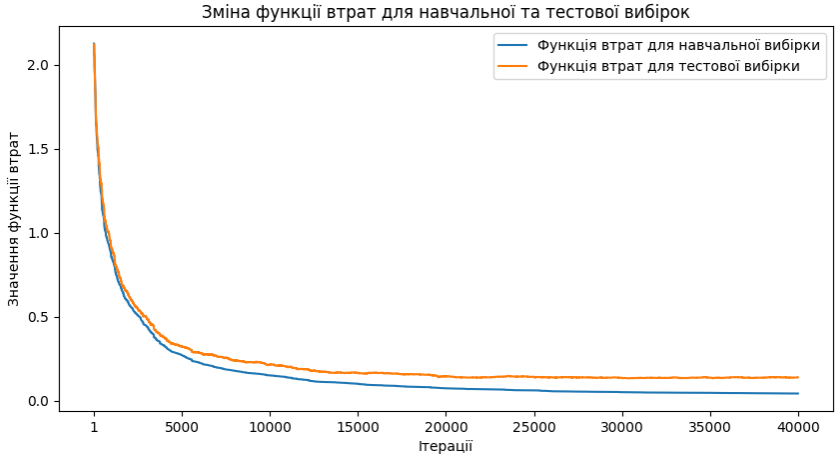


(в)

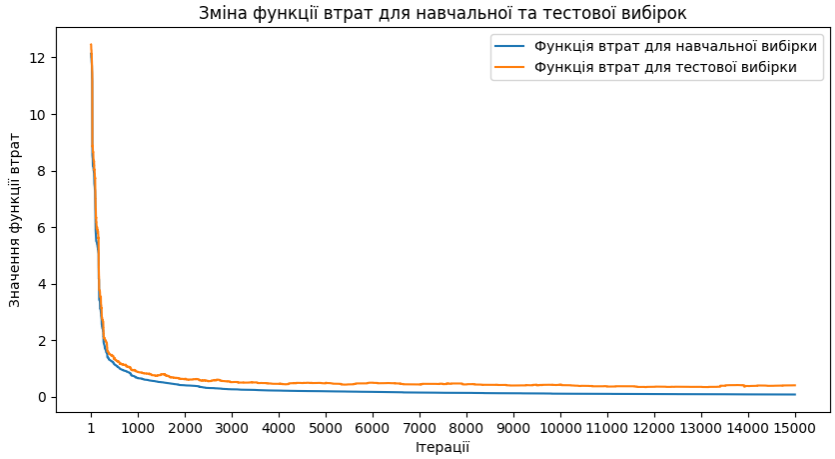
Рисунок 3.2 – Графіки залежності функцій втрат від кількості ітерацій для методів: (а) MLP with gradient descent, (б) MLP with single-point mutation, (в) $(1 + \lambda)$-EA with GP encodings, для задачі бінарної класифікації картинок



(а)

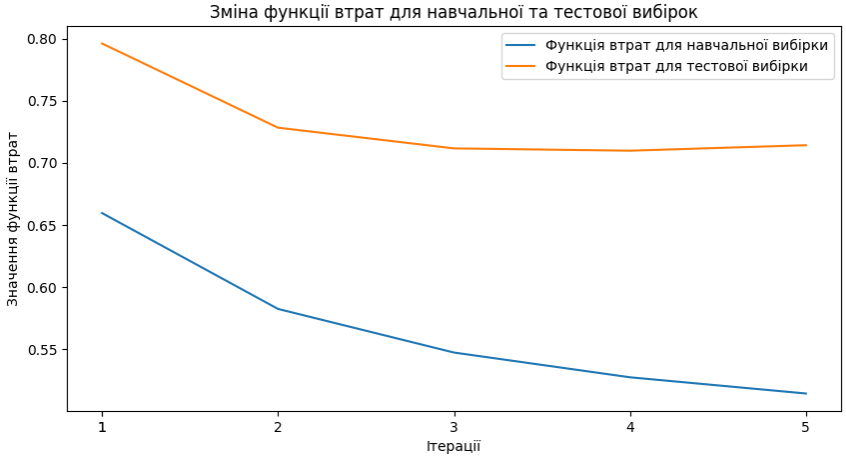


(б)

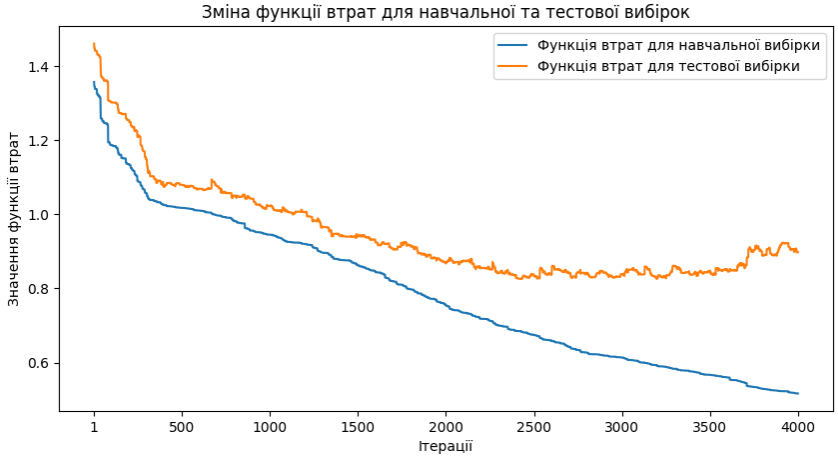


(в)

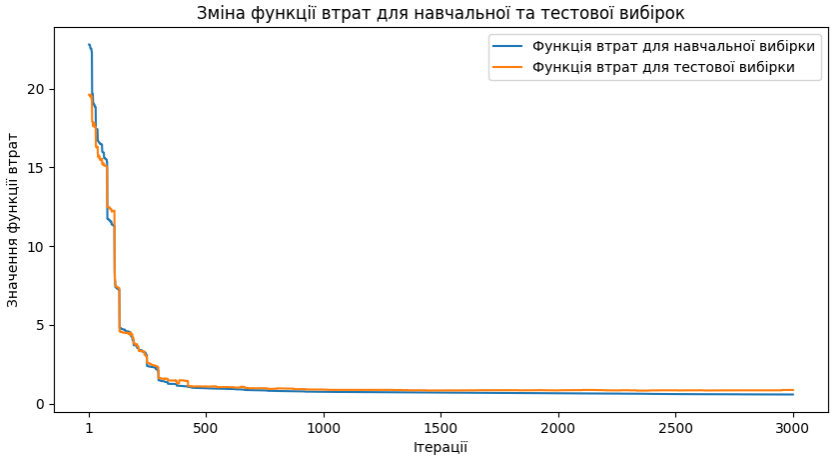
Рисунок 3.3 – Графіки залежності функцій втрат від кількості ітерацій для методів: (а) MLP with gradient descent, (б) MLP with single-point mutation, (в) $(1 + \lambda)$-EA with GP encodings, для задачі багатокласової класифікації табличних даних



(а)



(б)



(в)

Рисунок 3.4 – Графіки залежності функцій втрат від кількості ітерацій для методів: (а) MLP with gradient descent, (б) MLP with single-point mutation, (в) $(1 + \lambda)$-EA with GP encodings, для задачі багатокласової класифікації картинок

ВИСНОВКИ

Загальні висновки до роботи повинні підсумовувати усі ваші досягнення у даному напрямку досліджень.

За кожним пунктом завдань, поставлених у вступі, у висновках повинен міститись звіт про виконання: виконано, не виконано, виконано частково (І чому саме так). Наприклад, якщо першим поставленим завданням у вас іде «огляд літератури за тематикою досліджень», то на початку висновків ви повинні зазначити, що «у ході даної роботи був проведений аналіз опублікованих джерел за тематикою (...), який показав, що (...)». Окрім простої констатації про виконання ви повинні навести, які саме результати ви одержали та проінтерпретувати їх з точки зору поставленої задачі, мети та загальної проблематики.

В ідеалі загальні висновки повинні збиратись з висновків до кожного розділу, але ідеал недосяжний. :) Однак висновки не повинні містити формул, таблиць та рисунків. Дозволяється (та навіть вітається) використовувати числа (на кшталт «розроблена методика дозволяє підвищити ефективність пустопорожньої балаканини на 2.71%»).

Наприкінці висновків необхідно зазначити напрямки подальших досліджень: куди саме, як вам вважається, необхідно прямувати наступним дослідникам у даній тематиці.

ЛІТЕРАТУРА

[1] Takuya Akiba та ін. “Optuna: A Next-generation Hyperparameter Optimization Framework”. В: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2019.

[2] N.Sarav anaN та V.Gaya thri. “Performance and Classification Evaluation of J48 Algorithm and Kendall’s Based J48 Algorithm (KNJ48)”. В: *International Journal of Computer Trends and Technology* 59 (2018), с. 73—80. URL: https://api.semanticscholar.org/CorpusID:69700602.

[3] Davide Anguita та ін. “A Public Domain Dataset for Human Activity Recognition Using Smartphones”. В: *21st European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2013*. Bruges, Belgium, квіт. 2013. URL: https://www.kaggle.com/datasets/uciml/human-activity-recognition-with-smartphones/data.

[4] Markus Breunig та ін. “LOF: Identifying Density-Based Local Outliers.” В: т. 29. Черв. 2000, с. 93—104. DOI: 10.1145/342009.335388.

[5] John Duchi, Elad Hazan та Yoram Singer. “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization”. В: *Journal of Machine Learning Research* 12.61 (2011), с. 2121—2159. URL: http://jmlr.org/papers/v12/duchi11a.html.

[6] Juan Durillo та ін. “On the Effect of the Steady-State Selection Scheme in Multi-Objective Genetic Algorithms”. В: т. 5467. Квіт. 2009, с. 183—197. ISBN: 978-3-642-01019-4. DOI: 10.1007/978-3-642-01020-0_18.

[7] Yongsheng Fang та Jun li. “A Review of Tournament Selection in Genetic Programming”. В: жовт. 2010, с. 181—192. ISBN: 978-3-642-16492-7. DOI: 10.1007/978-3-642-16493-4_19.

[8] Félix-Antoine Fortin та ін. “DEAP: Evolutionary Algorithms Made Easy”. В: *Journal of Machine Learning Research* 13 (лип. 2012), с. 2171—2175.

[9] Gongde Guo та ін. “KNN Model-Based Approach in Classification”. В: *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*. За ред. Robert Meersman, Zahir Tari та Douglas C. Schmidt. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, с. 986—996.

[10] Gongde Guo та ін. “KNN Model-Based Approach in Classification”. В: (серп. 2004).

[11] Vinutha H P, B. Poornima та B. Sagar. “Detection of Outliers Using Interquartile Range Technique from Intrusion Dataset”. В: січ. 2018, с. 511—518. ISBN: 978-981-10-7562-9. DOI: 10.1007/978-981-10-7563-6_53.

[12] Kaiming He та ін. “Deep residual learning for image recognition”. В: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, с. 770—778.

[13] D.W. Hosmer та S. Lemeshow. *Applied Logistic Regression*. Applied Logistic Regression. Wiley, 2004. ISBN: 9780471654025. URL: https://books.google.com.ua/books?id=Po0RLQ7USIMC.

[14] Daniel Kermany, Kang Zhang та Michael Goldbaum. *Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification*. Вер. 2. Січ. 2018. DOI: 10.17632/rsxcbjbr9sj.2. URL: http://dx.doi.org/10.17632/rsxcbjbr9sj.2.

[15] Diederik P Kingma та Jimmy Ba. "Adam: A method for stochastic optimization". В: *arXiv preprint arXiv:1412.6980* (2014).

[16] Adam Lipowski та Dorota Lipowska. "Roulette-wheel selection via stochastic acceptance". В: *Physica A: Statistical Mechanics and its Applications* 391 (вер. 2011). DOI: 10.1016/j.physa.2011.12.004.

[17] Andrzej Maćkiewicz та Waldemar Ratajczak. "Principal components analysis (PCA)". В: *Computers and Geosciences* 19.3 (1993), с. 303—342. ISSN: 0098-3004. DOI: https://doi.org/10.1016/0098-3004(93)90090-R. URL: https://www.sciencedirect.com/science/article/pii/009830049390090R.

[18] TorchVision maintainers та contributors. *TorchVision: PyTorch's Computer Vision library*. https://github.com/pytorch/vision. 2016.

[19] Anqi Mao, Mehryar Mohri та Yutao Zhong. "Cross-entropy loss functions: Theoretical analysis and applications". В: *International Conference on Machine Learning*. PMLR. 2023, с. 23803—23828.

[20] Antonio Menditto, Marina Patriarca та Bertil Magnusson. "Understanding the meaning of accuracy, trueness and precision". В: *Accreditation and Quality Assurance* 12 (жовт. 2007), с. 45—47. DOI: 10.1007/s00769-006-0191-z.

[21] Elliot Meyerson, Xin Qiu та Risto Miikkulainen. "Simple genetic operators are universal approximators of probability distributions (and other advantages of expressive encodings)". В: *Proceedings of the Genetic and Evolutionary Computation Conference*. 2022, с. 739—748.

[22] Adam Paszke та ін. "Automatic differentiation in PyTorch". В: (2017).

[23] F. Pedregosa та ін. "Scikit-learn: Machine Learning in Python". В: *Journal of Machine Learning Research* 12 (2011), с. 2825—2830.

[24] Joanne Peng, Kuk Lee та Gary Ingersoll. "An Introduction to Logistic Regression Analysis and Reporting". В: *Journal of Educational Research - J EDUC RES* 96 (вер. 2002), с. 3—14. DOI: 10.1080/00220670209598786.

[25] *Pima Indians Diabetes Database*. URL: https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database/data.

[26] Riccardo Poli та W. B. Langdon. "Genetic Programming with One-Point Crossover". В: *Soft Computing in Engineering Design and Manufacturing*. За ред. P. K. Chawdhry, R. Roy та R. K. Pant. London: Springer London, 1998, с. 180—189.

[27] Marius-Constantin Popescu та ін. “Multilayer perceptron and neural networks”. В: *WSEAS Transactions on Circuits and Systems* 8 (лип. 2009).

[28] David Powers та Ailab. “Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation”. В: *J. Mach. Learn. Technol* 2 (січ. 2011), с. 2229—3981. DOI: 10.9735/2229-3981.

[29] Raul Robu та Holban Stefan. “A genetic algorithm for classification”. В: трав. 2011, с. 52—56.

[30] Usha Ruby та Vamsidhar Yendapalli. “Binary cross entropy with deep learning technique for Image classification”. В: *International Journal of Advanced Trends in Computer Science and Engineering* 9 (жовт. 2020). DOI: 10.30534/ijatcse/2020/175942020.

[31] Jasper Snoek, Hugo Larochelle та Ryan P Adams. “Practical bayesian optimization of machine learning algorithms”. В: *Advances in neural information processing systems* 25 (2012).

[32] Marina Sokolova, Nathalie Japkowicz та Stan Szpakowicz. “Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation”. В: т. Vol. 4304. Січ. 2006, с. 1015—1021. ISBN: 978-3-540-49787-5. DOI: 10.1007/11941439_114.

[33] Tamer Soliman та Ayman Abd-elaziem. “A Multi-Layer Perceptron (MLP) Neural Networks for Stellar Classification: A Review of Methods and Results”. В: *International Journal of Advances in Applied Computational Intelligence* 3 (серп. 2023). DOI: 10.54216/IJAACI.030203.

[34] William M. Spears та Kenneth A. De Jong. “An Analysis of Multi-Point Crossover”. В: за ред. GREGORY J.E. RAWLINS. Т. 1. Foundations of Genetic Algorithms. Elsevier, 1991, с. 301—315. DOI: https://doi.org/10.1016/B978-0-08-050684-5.50022-7. URL: https://www.sciencedirect.com/science/article/pii/B9780080506845500227.

[35] The pandas development team. *pandas-dev/pandas: Pandas*. Вер. latest. Лют. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[36] Guido Van Rossum та Fred L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.

[37] Vikramkumar, B Vijaykumar та Trilochan. “Bayes and Naive Bayes Classifier”. В: 2014. URL: https://api.semanticscholar.org/CorpusID:10272111.