НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Навчально-науковий інститут прикладного системного аналізу
Кафедра штучного інтелекту

До захисту допущено:
Завідувач кафедри
_____ Чумаченко О. Іл.
«__» _____20__ р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системи та методи штучного інтелекту»

спеціальності 122 «Комп'ютерні науки»

на тему: «Напівкероване навчання нейронних мереж на основі графів»

Виконала: студентка IV курсу,
групи КІ-91
Мосійчук Юлія Олександрівна _____

Керівник:
д.н.т. проф. Синєглазов Віктор Михайлович _____

Консультант з економічного розділу:
Доцент, к.е.н. Рощина Надія Василівна _____

Консультант з нормоконтролю:
фахівець першої категорії кафедри ШІ
Гончарук Максим Миколайович _____

Рецензент:
проф. каф. АКІК НАУ
к. т. н. проф. Філяшкін Микола Кирилович _____

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

Київ – 2023 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Інститут прикладного системного аналізу
Кафедра штучного інтелекту

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки»

Освітньо-професійна програма «Системи та методи штучного інтелекту»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ О. І. Чумаченко

«__» _____20__ р.

ЗАВДАННЯ

на дипломну роботу студенту

Мосійчук Юлії Олександрівни

1. Тема роботи: «Напівкероване навчання нейронних мереж на основі графів», керівник роботи Синєглазов Віктор Михайлович, затверджений наказом по університету від «30» травня 2023 року №2065-с.
2. Термін подання студентом роботи 8.06.2023
3. Вихідні дані до роботи: двовимірні датасети точок різного виду: концентричні кільця, "місяці" далеко, близько розташовані та класичні, у формі бананів.
4. Зміст роботи: Аналіз предметної області, обробка літературних джерел, аналіз алгоритму розв'язку задачі напівкерованого навчання нейронних мереж на основі графів, розробка програмного продукту для розв'язку задачі бінарної класифікації точок на площині, тестування та аналіз результатів.
5. Перелік ілюстративного матеріалу: опис наборів даних, архітектура нейронних мереж, опис алгоритмів напівкерованого навчання, результати навчання графовим методом та нейронною мережею.

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Економічний	Рощина Надія Василівна, канд. економ. наук, доцент		

7. Дата видачі завдання: 21 лютого 2023 року

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Огляд літератури за темою	12.03.2023	
2.	Підготовка першого розділу	16.04.2023	
3.	Підготовка другого розділу	7.05.2023	
4.	Розробка програмного продукту	18.05.2023	
5.	Підготовка третього розділу	21.05.2023	
6.	Підготовка економічної частини	24.05.2023	
7.	Оформлення розділів відповідно до нормоконтролю	26.05.2023	
8.	Оформлення дипломної роботи	28.05.2023	
9.	Підготовка презентації доповіді	28.05.2023	

Студент Юлія МОСІЙЧУК

Керівник Віктор СИНЄГЛАЗОВ

РЕФЕРАТ

Дипломна робота складається з: 89 с., 25 табл., 30 рис., 1 додатку, 20 джерел.

КЛАСИФІКАЦІЯ, МАШИННЕ НАВЧАННЯ, НАПІВКЕРОВАНЕ НАВЧАННЯ, ГРАФОВИЙ МЕТОД, НЕЙРОННІ МЕРЕЖІ

У роботі розглянуто різні алгоритми машинного навчання, зокрема напівкероване навчання нейронних мереж на основі графів та його різні підходи і реалізації, а саме методи спарсифікації, вибору функції оцінки ваг, способу навчання. Обрані підходи та методи були застосовані для розв'язання практичної задачі, а саме бінарної класифікації точок датасетів різної природи на площині.

Об'єкти дослідження: задача бінарної класифікації датасетів у вигляді точок на площині різного характеру, які належать двом класів.

Предмети дослідження: методи навчання штучного інтелекту, напівкероване навчання, підходи та методи використання графового методу, кероване навчання нейронних мереж, вплив природи даних на ефективність графового методу та нейронних мереж.

ABSTRACT

The thesis consists of: 89 p., 25 tables, 30 fig., 1 appendix, 20 sources.

CLASSIFICATION, MACHINE LEARNING, SEMI-SUPERVISED LEARNING, GRAPH METHOD, NEURAL NETWORKS

The work examines various machine learning algorithms, in particular, semi-supervised learning of neural networks based on graphs and its various approaches and implementations, namely, sparsification methods, selection of weight estimation function, learning methods. The selected approaches and methods were applied to solve a practical problem, namely binary classification of dataset points of different nature on the plane.

Research objects: binary classification task of datasets in the form of points on a plane of different nature, which belong to two classes.

Research subjects: artificial intelligence learning methods, semi-supervised learning, approaches and methods of using the graph method, supervised learning of neural networks, the influence of the nature of data on the effectiveness of the graph method and neural networks.

ЗМІСТ

ВСТУП

Машинне навчання стає все більш популярною галуззю інформатики та штучного інтелекту, яка зосереджується на розробці алгоритмів, які дозволяють комп'ютерам навчатися на основі даних і робити прогнози чи рішення без явного програмування. Така стрімка популярність та актуальність обумовлена тим, що саме за допомогою ШІ можна автоматизувати велику кількість задач, такі як діагностика, розпізнавання зображень і т. д., а також покращити та полегшити собі життя. Люди використовують ШІ у повсякденні навіть про це не здогадуючись, наприклад, це стосується рекомендаційних систем Google, YouTube, штучних перекладачів, асистентів і т. п.

Однією з ключових переваг машинного навчання є здатність розкривати закономірності та взаємозв'язки в складних і великих наборах даних, які можуть бути неочевидними для людини. Це дозволило досягти значних успіхів у таких задачах, як обробка природної мови, розпізнавання зображень і мови, а також прогнозне моделювання.

Відповідно, сфера застосування машинного навчання для рішення задач продовжує розвиватися та охоплювати найрізноманітніші галузі. Машинне навчання можна умовно розділити на три основні типи: кероване навчання, некероване навчання та навчання з підкріпленням. Кероване навчання передбачає навчання моделі на позначених даних для прогнозування нових результатів, тоді як некероване – пошук закономірностей у непозначених даних. Але у реальному світі досить часто виникає проблема нестачі маркованих даних, тому все більшої популярності набуває напівкероване машинне навчання, яке навчається на інформації з невеликої кількості маркованих та великої кількості немаркованих даних.

Одним відносно новим методом напівкерованого навчання є навчання на основі графів, яке використовується в різних сферах, таких як комп'ютерне бачення, обробка природної мови, аналіз соціальних мереж і систем рекомендацій.

Недоліком такого методу є його трансдуктивність, тобто неможливість використовувати побудовану модель для класифікації елементів, які не входять у навчальну вибірку. Метою роботи є поєднання трансдуктивного графового методу напівкерованого навчання з нейронною мережею для оцінки нових екземплярів.

У першому розділі розглянуто перспективи та необхідність використання ШІ, нейронні мережі та їх класифікацію, методи та підходи машинного навчання.

У другому розділі наведено особливості застосування графового методу напівкерованого навчання та постановку задачі напівкерованого навчання.

У третьому розділі наведено опис датасетів, що розглядаються та оцінка і аналіз результатів виконання програмного продукту.

У четвертому розділі проведено функціонально-вартісний аналіз програмного продукту.

Предметом дослідження є задача бінарної класифікації множини точок датасетів різної складності методом напівкерованого навчання нейронних мереж на основі графів.

РОЗДІЛ 1 НЕОБХІДНІСТЬ ВИКОРИСТАННЯ НАПІВКЕРОВАНОГО НАВЧАННЯ ДЛЯ РОЗРОБКИ СИСТЕМ ШТУЧНОГО ІНТЕЛЕКТУ

1.1 Перспективи використання штучного інтелекту в Україні

У той час, коли у інших країнах активно використовувалися технології штучного інтелекту, залучалися інвестиції у розвиток ШІ, зокрема у США вже активно застосовуються роботи для діагностики та проведення операцій[1][2], наприклад, в окремих напрямках широко використовується робот Da Vinci, який забезпечує точне та якісне проведення операцій, в Україні лише зароджується застосування штучного інтелекту, в тому числі роботизованих систем, в медицині.

Впровадження та використання інформаційних технологій, які включають в себе технології штучного інтелекту, є надзвичайно важливим компонентом розвитку держави. У статті О. Краковецького[3] розглянуто національні стратегії країн щодо розвитку ШІ на 2018-й рік. Зокрема Комісія ЄС прийняла Communication on Artificial Intelligence документ, у якому висвітлено три підходи розвитку ШІ – це збільшити технологічний та промисловий потенціал ЄС та рівень використання ШІ у державному і приватному секторах, підготувати європейців до змін, спричинених впровадженням технологій ШІ та забезпечити належний правовий та етичний рівень.

Хоча Україна посідає далеко не найперше місце у розвитку технологій штучного інтелекту, останні роки можна відслідкувати стрімку діджиталізацію країни, зокрема в Концепції Єдиної судової інформаційно-телекомунікаційної системи прийнятої у 2018 році[4] окреслено необхідність впровадження автоматизованих систем у судову систему, що передбачає покращення застосовуваних засобів, в тому числі використання новітніх технологій таких як штучний інтелект та машинне навчання. Вже у 2020-ому році в Україні використовувався додаток Дія “Держава у смартфоні”, який

продовжує вдосконалюватися та забезпечує надання державних, економічних послуг дистанційно та без зайвих папірців; у 2021-ому році була затверджена стратегія кібербезпеки України[5]; розвивається сфера електронного банкінгу, страхування, медицини і т. п., що дозволяє України виділятися з-поміж країн Європи.

Оскільки Україна постійно розвивається, можливості та сфери використання технологій штучного інтелекту розширюються. Актуальним питанням, особливо під час війни, є забезпечення інформаційною безпеки суспільства та країни. Нейронні мережі дозволяють відловлювати, попереджати зломи та інші кібератаки. Зокрема ШІ активно застосовується у визначенні неправомірної поведінки користувачів або вірусного програмного забезпечення, фільтрації злоякісних повідомлень, що запобігає витоку конфіденційних та секретних даних, зараженню комп'ютерних систем злоякісним ПО тощо. Також нейронні мережі можуть знайти своє застосування у фільтрації інформації, оскільки внаслідок інформаційно-психологічної війни зростає кількість пропаганди та деструктивної інформації, спрямованої на українців.

Крім того, актуальним питанням є застосування ШІ не лише в інформаційній діяльності, а також у:

- медицині: використання роботизованих систем для точних операцій, використання систем діагностики, розпізнавання злоякісних утворень, виявлення ураження, застосування нанотехнологій тощо;
- промисловості: використання маніпулятор у виготовленні продукції, для виявлення ушкоджень або браку, для випробовування обладнання тощо;
- оборонній промисловості: створення оборонних комплексів, дронів, іншого обладнання та зброї із застосуванням новітніх технологій ШІ;
- економіці: аналіз якості продукції, створення досконалої рекомендаційної системи для забезпечення інтересів споживачів, прогнозування ризиків і т. д.

1.2 Штучні нейронні мережі та їх класифікація

Нейронна мережа – це математична обчислювальна модель, будова якої ґрунтується на біологічних нейронних мережах. Особливістю нейронних мереж є здатність навчатися. Навчання нейронної мережі – це її налаштування з метою покращити кінцевий результат шляхом коригування ваг та мінімізації похибки.

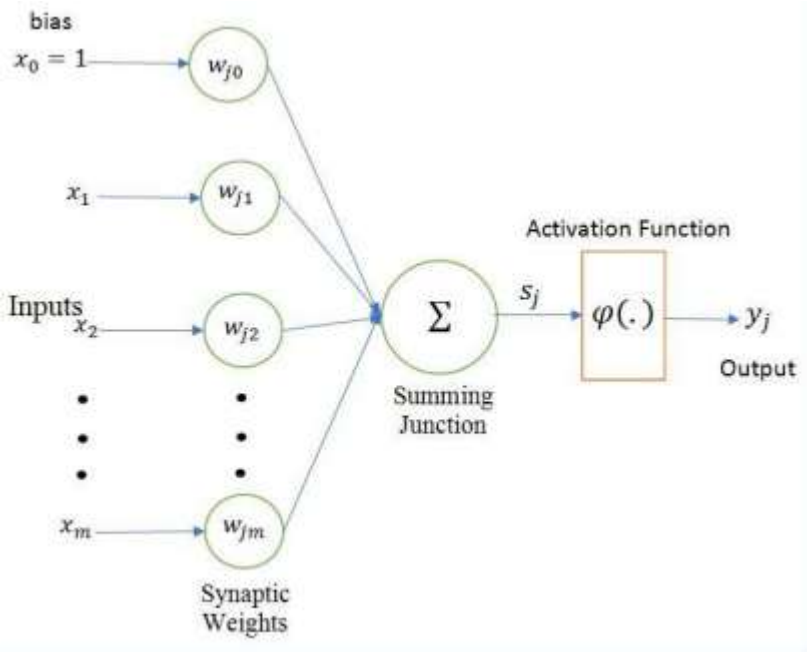


Рис. 1.1 - Нелінійна модель нейрона штучної НМ[6].

На рис. 1.1 можна побачити будову нейрона такої штучної нейронної мережі. Біологічний нейрон має дендрити, на які поступають сигнали, та аксон для виходу сигналу з нейрону і передачі його далі по мережі. Відповідно, у якості вхідних сигналів моделі, що розглядається, виступають значення x_i, які множаться на відповідний ваговий коефіцієнт w_{ji}, де j – номер нейрону, та сумуються разом зі зміщенням $x_0 w_{j0}$. Отримане значення називається зваженою сумою. Далі зважена сума подається на вхід нелінійної функції активації, це може бути сигмоїдна функція, гіперболічний тангенс, ReLU, Leaky ReLU, softmax тощо. Якщо нейрон є лінійною моделлю, то застосування функції активації опускається. Вибір функції активації залежить від будови нейронної мережі, її прикладного значення, предметної області застосування і

т. п. Вихід функції активації є виходом нейрона, який подається на вхід наступних нейронів.

Відповідно, нейрони у штучних нейронних мережах можуть об’єднуватися у шари, а також передавати інформацію як між шарами, так і в межах одного шару. Перший шар нейрону, на який поступають вхідні сигнали називається вхідним шаром, останній шар, який відповідає за формування остаточного виходу мережі – вихідний шар. Між ними можуть розміщуватися один або декілька прихованих шарів. Багатошаровість дозволяє вирішувати більш комплексні задачі машинного навчання.

Коригування ваг в штучних нейронних мережах відбувається із використанням методу зворотного поширення похибки. Він включає в себе обчислення похідної функції вартості (функції втрат) відносно ваг та подальше оновлення ваг за допомогою стохастичного градієнтного спуску, або інших методів. Загалом поняття зворотне поширення похибки включає в себе розуміння того, що похибка поширюється від вихідного шару до вхідного, тоді як нейронна мережа поширює дані у зворотному напрямку – від входу до виходу.

Ґрунтуючись на цьому можна навести деякі класифікації нейронних мереж.

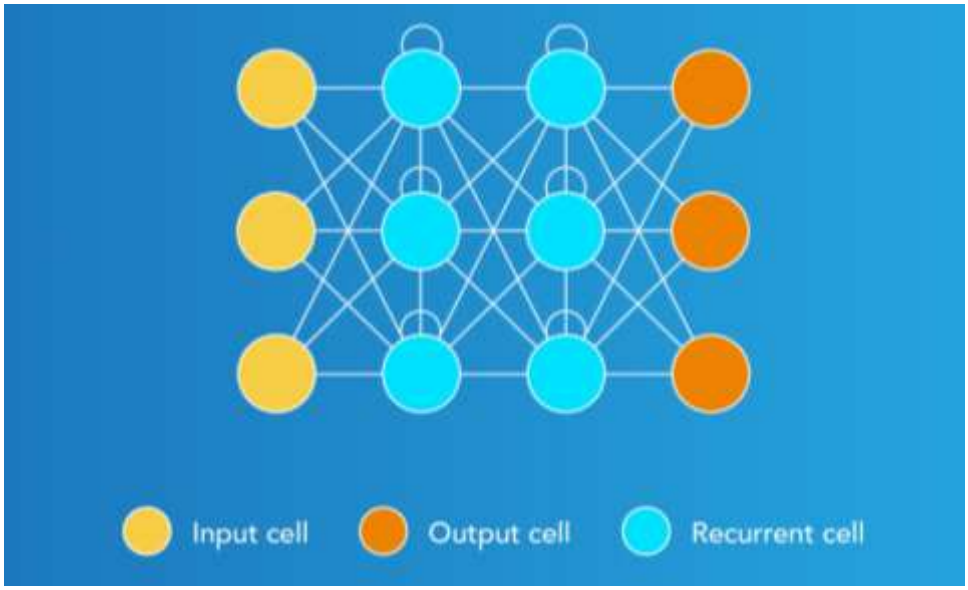


Рис. 1.2 - Рекурентна нейронна мережа[7].

За напрямком поширення можна виділити:

1. Мережі прямого поширення – мережі, у яких нейрони поточного шару впливають лише на нейрони наступного шару, тобто всі виходи нейронів спрямовані на входи нейронів наступного шару.

Перцептрони належать саме до цього класу нейронних мереж.

2. Рекурентні мережі – мережі, у яких нейрони можуть впливати одне на одного в межах одного шару та навіть на самих себе (рис. 1.2). Це дозволяє мережам приймати на вхід послідовності даних, що є необхідним для вирішення задач розпізнавання мовлення, аналізу та класифікації текстів, доповнення текстів, перевірки граматики, перекладу текстів тощо.

За принципом будови можна виділити[7]:

1. Одношаровий перцептрон.

Перцептрон – це простий штучний нейрон, який приймає один або кілька вхідних даних, комбінує їх за допомогою вагових коефіцієнтів і створює єдиний вихід. Вихід визначається тим, чи перевищує зважена сума вхідних даних певний поріг чи ні. Перцептрон є алгоритмом керованого машинного навчання для рішення задачі бінарної класифікації, який дозволяє розділити вхідні дані на дві категорії. Навчання відбувається шляхом корегування ваг в залежності від похибки між фактичними виходами та цільовими значеннями. Вперше представлений Френком Розенблаттом у 1957-ому році. Даний алгоритм є простим у реалізації, але показує невисоку ефективність на комплексних задачах, що зумовило створення його багатошарової альтернативи.

2. Багатошаровий перцептрон.

Кожен окремий вузол такої нейронної мережі (рис. 1.3) з’єднаний з усіма нейронами наступного шару, що робить її повнозв’язною мережею. Мережа складається з вхідного та вихідного шари з одним або кількома прихованими шарами. Зважена сума нейрону подається на вхід кожного нейрону у наступному шарі, можливо із застосуванням нелінійної функції активації для кожного шару.

Багатошарові перцептрони можуть використовуватися для розпізнавання мовлення, класифікації, тощо.

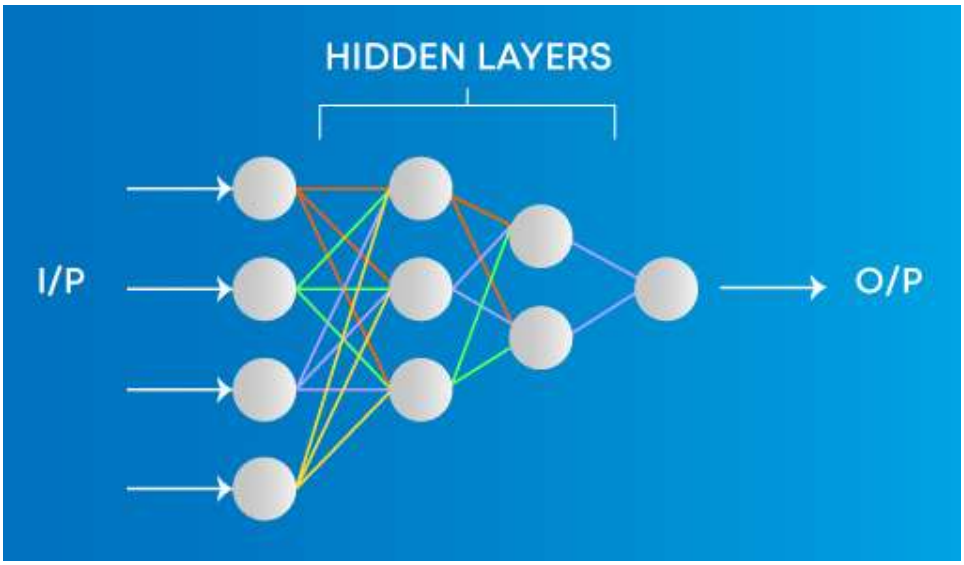


Рис. 1.3 - Модель багатошарового перцептрона[7].

3. Згорткові нейронні мережі.

Архітектура згорткових нейронних мереж ґрунтується на морфології нейронів кори головного мозку, яка відповідає за зорове сприйняття, а саме у 1958 і 1959 рр. Девід Х. Хьюбел і Торстен Візель зробили висновки, що багато нейронів в зоровій корі мають невелике локальне рецепторне поле, деякі нейрони реагують тільки на зображення ліній у певному напрямку (наприклад горизонтальні), частина нейронів сприймає складні образи, які є об’єднанням низькорівневих образів, кожний нейрон пов’язаний тільки з декількома нейронами попереднього рівня[8].

Тобто однією з відмінностей згорткових нейронних мереж є те, що вони не є повнозв’язними, на вхід кожного нейрону наступного шару подаються виходи лише частини нейронів з попереднього шару.

Для згорткових нейронних мереж характерна наявність таких шарів[9]:

- шару згортки (перший шар зазвичай є шаром згортки)
- шару пулінгу
- повнозв’язного шару (є вихідним шаром)

Шар згортки представляє собою проходження матриці-фільтра по даним (наприклад по матриці, що описує зображення) та дозволяє виділити ключові властивості зображення, а шари пулінгу акумулюють значення декількох клітинок (це може бути максимальне, або мінімальне значення, тощо), що дозволяє акцентувати увагу на важливішій/виразнішій інформації та зменшувати розмірність даних.

В якості функцій активації прихованих шарів часто використовується ReLU, для задачі класифікації вихідна функція – softmax. Згорткові мережі часто використовується для розпізнавання і обробки зображень, розпізнавання мовлення тощо.

4. Мережі радіально базисних функцій.

Мережа радіально базисних функцій – це двошарова нейронна мережа, головні відмінності якої полягають у тому, що у першому шарі, замість підрахунку зваженої суми вхідних даних та коефіцієнтів ваг, обчислюється відстань між вхідним вектором та вектором ваг та множиться на значення зміщення, тоді як у перцептроні зміщення додається[10]. Тобто кожен рядок матриці ваг виступає точкою, у якій вхідне значення буде нульовим. Зміщення виконує операцію масштабування базисної функції.

Мережа радіально базисних функцій може використовуватися для апроксимації функцій та розпізнавання патернів у вхідних даних.

5. Довга короткочасна пам’ять (Long short-term memory).

Недоліком рекурентних нейронних мереж є ефект зникнення градієнту. В таких випадках значення градієнту сягає нуля та спричиняє зупинку навчання (наприклад, якщо нейронна мережа сягнула локального мінімуму функції втрат).

На допомогу приходять LSTM моделі, вузли якої використовують три вентилі: вхідний, вихідний та вентиль забуття. Вхідні вентилі контролюють, які дані слід зберігати в пам’яті. Вихідні – які дані передаватимуться на наступний рівень, а вентиль забуття відповідає за видалення вже непотрібних

даних. Така будова дозволяє LSTM мережам обробляти дані, між якими є часові затримки.

LSTM мережа вирішує проблему зникаючого градієнту, не використовуючи функції активації в рекурентних компонентах і не змінюючи збережені значення.

LSTM мережі вирішують задачі розпізнавання рукописного тексту, мовлення тощо.

1.3 Огляд методів машинного навчання

Усі методи машинного навчання поділяються на чотири типи – це кероване, некероване, напівкероване навчання та навчання з підкріпленням. Розглянемо окремо кожний з них.

Некероване навчання[11] передбачає організацію елементів, представлених характеристиками $\{x_1, \dots, x_n\}$. Особливістю є те, що елементи немарковані, тобто кожному елементу з вхідного датасету не ставиться у відповідність жодної позначки. Зазвичай некероване навчання вирішує задачі кластеризації, виявлення викидів, тобто елементів датасету, які суттєво вирізняються з-поміж інших, зменшення розмірності з виділенням певних властивостей. Прикладом найелементарнішої задачі може бути кластеризація точок на площині.

Недоліком такого підходу є обмеженість використання та неточність, оскільки значень для перевірки немає.

Протилежним до некерованого навчання є кероване навчання, де кожному елементу з датасету ставиться у відповідність цільове значення. Тобто навчальний набір має вигляд $\{(x_1, y_1), \dots., (x_n, y_n)\}$, де y_i – це цільове значення елементу $i,\ i = \overline{1,n}$ Наприклад, для задачі розпізнавання зображень характеристиками одного елементу, зображення, слугуватимуть значення кольору відповідного пікселю, а цільовим значенням буде назва зображеного

об’єкту. Для задачі розпізнавання цифр дані маркуються значеннями від 0 до 9 і т. д.

Задача керованого навчання називається регресією, коли $y \in R$ та класифікацією, коли y може приймати кілька дискретних значень (наприклад, розпізнавання зображень).

Навчання з підкріпленням є зовсім відмінною концепцією навчання, яка передбачає, що система, яка навчається, на кожному кроці аналізує середовище та обирає дію, а після отримує винагороду. Записуючи дії та використовуючи метод спроб і помилок у встановленому середовищі, система може максимізувати кумулятивну винагороду.

Перевагою такого навчання є те, що система орієнтується лише на дані, зібрані та досліджені під час роботи моделі, не спираючись на історичні дані, як цього потребують інші підходи навчання. Такі системи показують гарні результати в іграх, наприклад, шахах або Go – відомі системи Deep Blue та Alpha Go, які здатні перемагати навіть професійних гравців.

Четвертим типом методів машинного навчання є напівкероване навчання. Відповідно назві, напівкероване навчання є містком між керованим та некерованим типами та дозволяє навчити модель за допомогою відносно малого набору позначених елементів, та поширити знання, а також отримати нову інформацію з непозначених даних, вибірка яких є зазвичай у рази більша. У наступному пункті розглянемо чому необхідне напівкероване навчання та в чому його основні переваги.

1.4 Необхідність використання напівкерованого навчання

Керовані методи машинного навчання є точними, оскільки кожний елемент навчального датасету є промаркованим, але у реальному світі досить часто виникає проблема в недоступності такої великої кількості позначених даних. Це може бути пов’язано з технічними складнощами, закритістю ресурсів або їх високою вартістю, вартістю обробки даних тощо. Проте

зазвичай непозначені дані є широко доступними у великій кількості, наприклад, якщо перед нами стоїть задача класифікації зображень або текстів, в мережі Інтернет з легкістю можна знайти сотні тисяч екземплярів датасету у сирому вигляді. Аналогічно із аудіозаписами, які можна отримати з бази радіозаписів або інших джерел. Проте класифікувати деяку скінченну множину екземплярів ймовірно доведеться власноруч. Це вимагає велику кількість зусиль і часу, що не є релевантно, особливо коли йдеться про мануальну класифікацію повного датасету, що включає величезну кількість екземплярів.

Незважаючи на те, що камери спостереження працюють цілодобово, у великій кількості місць, є можливість придбати камеру для власного дослідження – виділення і позначення кожного об’єкту, або об’єктів, які досліджуються є ресурсозатратним.

Певний клас спостережуваного датасету може включати переважну кількість екземплярів, тоді як деякий інший не містити жодного позначеного прикладу. Наприклад, за розглянутий період часу до лікарні, або системи, у якій дослідження має місце не приходило жодного пацієнта з хворобою X, тоді як більшість хворіє хворобою Y, що спотворює результати класифікації.

Для виділення та класифікації певних характеристик може бути необхідною експертна думка. Наприклад, для вирішення задач обробки, розпізнавання мовлення позначати дані варто із залученням експертів – лінгвістів, носіїв мови, тощо. Для ідентифікації тривимірної структури білка можуть знадобитися місяці дорогої лабораторної роботи експертів-кристалографів[11].

Іншою проблемою є неточність вже доступних позначень екземплярів датасету. Гарним прикладом можуть бути теги під відео, фотографіями, статтями, тощо. Користувачі мереж, журналісти і інші досить часто додають неправдиві, кричущі теги для зацікавлення та приваблювання аудиторії, що заважає використовувати їх для адекватної моделі класифікації. Крім того,

теги можуть назначатися автоматично неякісними моделями, що також унеможливлює їх використання для розв’язання задачі класифікації.

Враховуючи наведені недоліки: недоступність позначених даних, недостовірність або недоцільність існуючих позначень, велику вартість мануальної класифікації даних, та переважаючу кількість та доступність необроблених, непозначених даних, можна зробити висновок, що при побудові моделі позначені дані варто поєднувати з непозначеними. Це дозволяє не лише зменшити витрати на дослідження та створення моделі, а також збільшити спектр розв’язуваних задач з використанням немаркованих даних.

Це і зумовило поширеність напівкерованого методу навчання, що включає в себе навчання на великому датасеті при наявності невеликої скінченної множини початково класифікованих даних, що покращує точність моделі. Використання немаркованих даних ґрунтується на пошуку патернів, схожостей у датасеті, модифікації або зміни пріоритетів гіпотез, отриманих лише з позначених даних[12]. Об’єднання керованого та некерованого методів машинного навчання подарувало абсолютно новий спосіб навчання та застосування моделей штучного інтелекту.

1.5 Класифікація методів напівкерованого навчання

Інші назви: “навчання на основі позначених і немаркованих даних” або “навчання на основі частково позначених/класифікованих даних”. Напівкероване навчання може бути трансдуктивним або індуктивним[12]. Трансдуктивна модель працює лише на даних, які використовувалися в навчанні, маркованих і ні, тоді як індуктивні моделі здатні класифікувати нові дані, які ще не розглядалися моделлю.

Серед найбільш поширених методів машинного навчання можна виділити EM (expectation–maximization метод) з генеративними сумішевими моделями, самонавчання, спільне навчання (co-training), трансдуктивні опорні

векторні машини та методи на основі графів[12]. Розглянемо окремо кожен з них.

1.5.1 Expectation–maximization з генеративними сумішевими моделями

Такі моделі припускають, що $p(x,y) = p(y)p(x|y)$, де $p(x|y)$ – розподіл суміші, який можна ідентифікувати, наприклад, для цього можна використовувати розподіл Гауса, тоді матимемо сумішеву модель Гауса. З великою кількістю немаркованих даних компоненти суміші можна ідентифікувати за одним маркованим екземпляром для кожного компонента (в ідеалі, але насправді для деякого класу можуть бути відсутні позначені екземпляри), щоб повністю визначити розподіл суміші, використовуючи expectation–maximization алгоритм – алгоритм максимізації математичного сподівання[12]. Алгоритм EM використовується для отримання оцінок максимальної правдоподібності параметрів, коли деякі дані відсутні[13]. Що дозволяє застосовувати його у напівкерованому навчанні.

Генеративні сумішеві моделі мають свої недоліки, наприклад, модель має бути повністю ідентифікованою (можливо знайти всі необхідні параметри розподілу суміші), адекватною і відповідною досліджуваному датасету – якщо модель вибрано неправильно, непозначені дані можуть зіпсувати її точність[12]. Обирати модель також варто із врахуванням структури екземплярів датасету. Наприклад розв’язуючи задачу категоризації тексту, де кожний екземпляр можна розділити на кілька підтем, доцільно використовувати моделі, що описуються декількома поліномами, а не одним. Для вибору моделі також слід користуватися початково відомими знаннями про розподіл даних.

Ще одним недоліком розглянутого методу є ймовірність збіжності EM-алгоритму до локального мінімуму або максимуму. Якщо локальний екстремум знаходиться далеко від глобального, непозначені дані можуть погіршити точність навчання моделі. Тому завжди доцільно пробувати різні початкові умови, або обирати їх, застосовуючи методи оптимізації.

1.5.2 Самонавчання (self-training)

Самонавчання можна поділити на два етапи: спершу класифікатор навчається на позначених даних, а потім застосовується на непозначених. Найбільш правдоподібні екземпляри з непозначеної множини додаються як позначені елементи до множини маркованих даних. Далі процес навчання повторюється, з кожною ітерацією позначаючи все більше екземплярів датасету.

Очевидно, що головною проблемою такого підходу є те, що класифікатор навчається на самостійно класифікованих даних, що може призвести до значної деградації моделі у разі неправильної початкової класифікації. Є алгоритми, які допомагають уникати такої поведінки, “забуваючи” класифіковані непозначені дані, які додалися до позначеної множини, якщо їхня правдоподібність менша певного порогу.

В літературі також можна зустріти назви “self-teaching”, “bootstrapping”[11].

Самонавчання може застосовуватися для вирішення різних задач, таких як: розпізнавання зображень, класифікація текстів на емоційні та нейтральні, розпізнавання поняття слова, що є частиною розкриття невизначеностей природної мови (стосується слів-омонімів), задач машинного перекладу та ін[12].

Застосування генеративних сумішевих моделей з EM-алгоритмом та метод самонавчання все ще є актуальними у наш час через їхню простоту реалізації та зрозумілість.

1.5.3 Спільне навчання (co-training)

Під час такого навчання припускається, що всі характеристики конкретного екземпляру датасету можна представити у двох виглядах, тобто розділити на дві підмножини, які незалежні одна від одної в межах класу. Кожна така множина вважається достатньою для навчання окремого

класифікатору. Класифікатори навчаються на позначених даних з відповідних підмножин, а потім використовують кілька найбільш точно визначених екземплярів для перенавчання одне одного. Далі процес повторюється, для перенавчання класифікаторів до позначених елементів підмножин додаються нові екземпляри. Два класифікатори (або гіпотези) мають бути узгоджені як на позначених даних, так і непозначених, яких може бути у кілька сотень, тисяч разів більше[12].

Спільне навчання можна використовувати для класифікації текстів у мережі Інтернет, адже текст на веб-сторінках можна представити як текст, так і за допомогою посилання на цей текст у мережі. До того ж, веб-сторінки містять тексти, які не є позначеними, якщо не враховувати теги, недоліки яких розглянуті у попередньому підрозділі, що вказує на доцільність використання напівкерованого навчання.

Існують також варіації методу самонавчання, у якому окремі класифікатори навчаються не на окремих представленнях множини характеристик елементу датасету, а на повному наборі, після чого аналогічно вже описаному, вони навчають одне одного на позначених даних та найбільш правдоподібно класифікованих первинно непозначених даних. Крім того, класифікаторів може бути не обов’язково два, а й більше.

1.5.4 Трансдуктивні опорні векторні машини

Transductive support vector machines в англомовній літературі. Мета SVM — знайти гіперплощину в N-вимірному просторі (N — кількість ознак або характеристик), яка чітко класифікує точки даних.

Рисунок 1.4 показує на прикладі двовимірного простору (коли конкретний екземпляр датасету описується двома характеристиками, кожній з них відповідає вісь на рисунку), що гіперплощин, які розділяють датасет може бути декілька, але задача полягає у пошуку такої гіперплощини, для якої відступ між крайніми точками класів найбільший.

Розмірність такої гіперплощини, очевидно, залежить від розмірності множини характеристик, яка описує один екземпляр датасету. Для двох характеристик маємо пряму, для трьох – площину у тривимірному просторі.

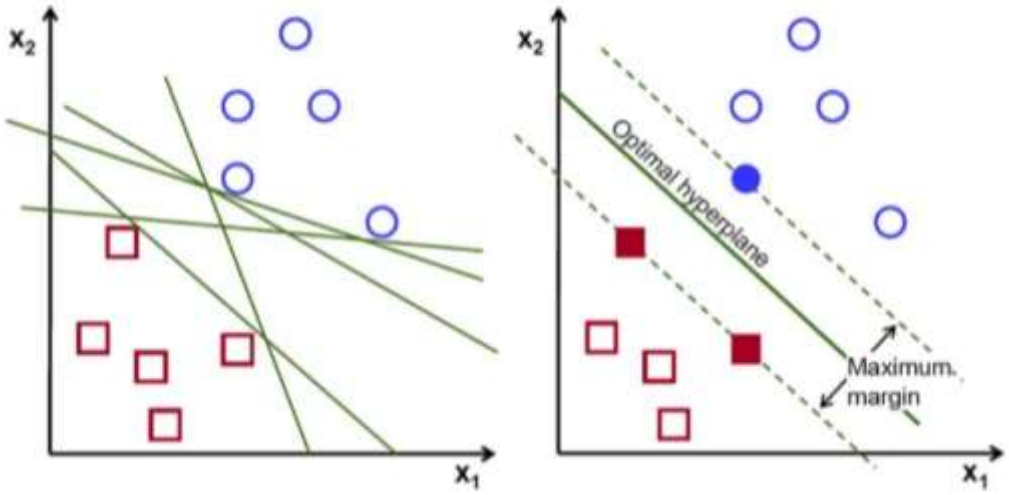


Рис. 1.4 - Пошук гіперплощини у двовимірному просторі[14].

На рис. 1.5 зображені опорні вектори (support vectors) – це точки датасету, які розташовані найближче до гіперплощини. Якщо видалити ці точки, гіперплощина зміститься до наступних таких точок. Саме опорні вектори використовуються для пошуку потрібної гіперплощини.

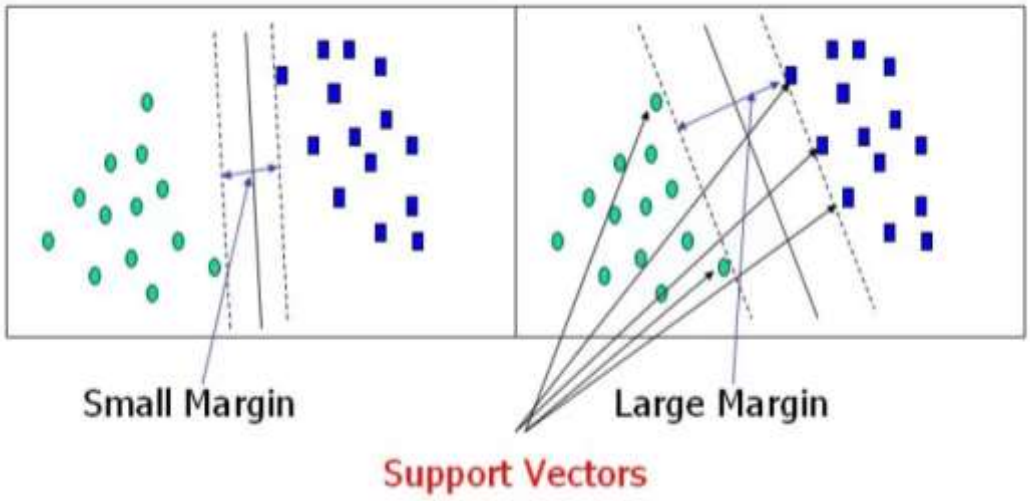


Рис. 1.5 - Опорні вектори[14].

Трансдуктивні SVM мають на меті знайти оптимальну гіперплощину у просторі з позначеними, та непозначеними (але позначеними у процесі

навчання) даними. Відповідно, оцінювання моделі відбувається не з використанням тестових даних, а даних з навчального датасету.

1.5.5 Методи на основі графів

В останній час графові методи машинного навчання стали набирати популярності. Їх суть полягає у представленні датасету та зв’язків між екземплярами у вигляді графу, в якому точка – один приклад з датасету (це може бути позначений чи непозначений екземпляр), а ребра відповідають за схожість двох екземплярів, у найпростішому випадку для цього можна використовувати, наприклад, евклідову відстань. Припускаємо, що з’єднані точки, з найменшою відстанню (з найбільшою схожістю) матимуть однакові позначення. Вершини, які відповідають позначеним даним, маркуються, а потім поширюють своє маркування далі по ребрам графу, до непозначених вершин. Марковані ж з самого початку вершини зберігають свої позначення. Таким чином, в кінці навчання отримуємо повністю позначений граф. Завдяки використанню структури графу графові методи показують точні результати.

Однією з задач графового підходу є задача вибору структури графу, тобто спосіб з’єднання вершин. Це включає в себе вибір правильної функції оцінки ваг ребер та кількості і методу з’єднання вузлів, наприклад, для цього можна використовувати певний поріг відстані, або регулювати кількість ребер з однієї вершини тощо. Від правильно обраної структури графу залежить те, наскільки він добре представлятиме досліджувані дані, тобто наскільки адекватну та точну модель ми отримаємо.

За замовчуванням графові методи трансдуктивні, тобто працюють лише на даних, які використовуються у навчанні, тоді як не підходять для класифікації екземплярів, які модель ще “не бачила”. Але є методи оптимізації графового підходу, що надають йому індуктивності.

Основними перевагами графових методів в порівнянні з іншими методами напівкерованого навчання є[15]:

- універсальність: у вигляді графу можна представити та вже представлено велику кількість даних, таких як: Всесвітня мережа, або World Wide Web, де наприклад прямі посилання на сайти можна представити вебграфом, вершини якого – це вебсторінки, які з’єднуються ґрунтуючись на наявності посилань на інші сторінки на сторінці відповідного вузла; мережі цитування – де дані можуть бути представлені графом з вершинами-документами та ребрами-цитуваннями певного документа в іншому; соціальні мережі, які демонструють соціальні взаємовідносини між суб’єктами, які можуть бути як індивідами, так і цілими організаціями;
- опуклість: оскільки для представлення даних в основному використовуються неорієнтовані графи (без направлених ребер) це дає змогу звести задачу навчання в задачу опуклої оптимізації, що охоплює багато різних методів;
- масштабованість: більшість методів навчання на основі графів розроблено так, щоб їхня складність лінійно залежала від кількості елементів датасету. Як наслідок їх часто розпаралелюють для обробки великих датасетів.

Дана робота присвячена огляду саме графового методу напівкерованого навчання.

1.6 Висновки до першого розділу

У даному розділі було розглянуто актуальність використання штучного інтелекту в Україні, поняття нейронних мереж та їх класифікацію, методи машинного навчання: кероване, некероване, напівкероване, а також навчання з підкріпленням, а особливо методи напівкерованого навчання, яке швидко набуває популярності, внаслідок частої недоступності позначених даних і навпаки – великої кількості немаркованих даних.

РОЗДІЛ 2 ОСОБЛИВОСТІ ВИКОРИСТАННЯ ГРАФОВОГО МЕТОДУ НАПІВКЕРОВАНОГО НАВЧАННЯ

2.1 Постановка завдання напівкерованого навчання

Оскільки напівкероване навчання є середнім між керованим та некерованим навчанням, в яких використовуються лише позначені, або лише непозначені датасети відповідно, задачею напівкерованого навчання стає класифікація непозначених даних за допомогою інформації з позначеної множини набору. Крім цього, напівкероване навчання залучає зв’язки, або схожість між конкретними екземплярами датасету, знаходить шаблони як між позначеними, так і непозначеними даними, що покращує точність моделі.

Як вже зазначалося, напівкеровані методи навчання широко застосовуються, оскільки у реальному світі більшість даних представлені у сирому вигляді без позначень.

Нижче наведені характеристики задачі, які дозволяють вирішити її методами напівкерованого навчання ефективніше:

- розмір множини непозначених даних у наборі набагато більший за розмір множини позначених;
- належність елементів до певного класу залежить від їхньої схожості – припускаємо, що елементи, які більш схожі один на одного відповідають одному класу, тобто, якщо точки належать одному кластеру, їхні результати – схожі;
- простота позначення та невелика розмірність.

У випадку напівкерованого навчання на основі графів, очевидно, що вимогою є можливість представлення датасету у вигляді графу. Головною метою є поширення позначень від маркованих даних до немаркованих, оптимізуючи значення деякої функції вартості, або функції витрат, яка описує точність моделі на досліджуваному датасеті.

2.2 Етапи використання графового методу

Графовий метод включає в себе наступні етапи[16]:

1. Вибір функції подібності, або схожості.

Необхідно обрати функцію, якою оцінюватиметься схожість двох екземплярів датасету, або довжина ребра між відповідними вершинами (similarity function). Досить часто в якості такої функції використовують функцію Гауса. Як альтернатива для даних, представлених у вигляді гістограми, наприклад, документів, описаних кількістю слів, або TF-IDF – показником, який характеризує важливість слів у документі, порівнюючи частоту вживання конкретного слова в документах, можна використовувати косинус чи критерій Пірсона (функція хі-квадрат). Також, як вже зазначалося вище, функцією схожості може слугувати евклідова відстань. Обирати функцію варто, спираючись на досліджуваний датасет та поставлену задачу.

Функція схожості слугує функцією оцінки ваг ребер графу між двома вершинами – елементами датасету.

2. Вибір способу спарсифікації (розрідження) графу.

Найпростіший спосіб з’єднати вершини – це побудувати повнозв’язний граф, тобто граф, в якому будь-які дві вершини з’єднані ребром. Але це не буде оптимальним рішенням, оскільки такий граф робить процес навчання ресурсозатратним та не є виправданим – немає потреби з’єднувати ребром вершини, сильно віддалені одна від одної, тобто з невеликою схожістю, оскільки з самого початку ми припускаємо, що вершини в межах одного кластеру матимуть один клас, тоді точки з меншою схожістю належатимуть до різних класів.

Тому існує декілька способів розрідження графу, або спарсифікації (graph sparsification), що включає в себе в першу чергу видалення ребер, залишаючи з’єднаними лише найближчих (або керуючись іншим алгоритмом) сусідів, а також можливим перевизначенням ваг ребер у графу. Спарсифікація

підвищує ефективність навчання, а також стійкість алгоритмів до шуму, і є невід'ємною складовою напівкерованого навчання на основі графів.

Варто виділити три найвідоміших алгоритми розрідження графів – це узгодження, або b-matching, епсилон-окіл, або ε-neighborhood та k найближчих сусідів, або kNN (k nearest neighbor).

a) ε-neighborhood.

Алгоритм, в якому обирається певне значення епсилон, якщо відстань між двома вершинами не перевищує дане значення, то вони з'єднуються ребром. Оскільки це фактично означає з'єднання вершин в одному околі, метод отримав таку назву.

b) k найближчих сусідів.

Як вже зрозуміло з назви, згідно методу кожна вершина з'єднується з k найближчими вершинами. Фактично метод теж об'єднує вершини в певному околі, хоча і визначає його іншим способом.

kNN є популярнішим алгоритмом[16], оскільки він більш адаптивний до кількості та щільності даних, у той час як неправильне порогове значення в графі епсилон-околі може призвести до роз'єднаних компонентів або підграфів у наборі даних або навіть до ізольованих одиночних вершин. Недоліком методу є те, що в процесі він продукує неорієнтований підграф, який не є kNN-графом, тобто існують вершини, з більше ніж k сусідами.

Задачу оптимізації методу kNN можна представити у вигляді[16]:

$$\min_{\hat{P}\in B} \sum\nolimits_{ij} \hat{P}_{ij} D_{ij}$$

$$\sum\nolimits_{j} \hat{P}_{ij} D_{ij} = k, \hat{P}_{ii} = 0, \forall i, j \ \in \overline{1,n}$$

де $\hat{P} \in B$ – матриця, яка характеризує зв'язки між вершинами, $\hat{P}_{ij} = 1$, якщо i і j з'єднуються ребром і 0, якщо ні. Відповідно елементи $\hat{P}_{ii}$ дорівнюють 0, оскільки жодна вершина не з'єднана сама з собою. D_{ij} – відстань між двома вершинами.

c) b-matching.

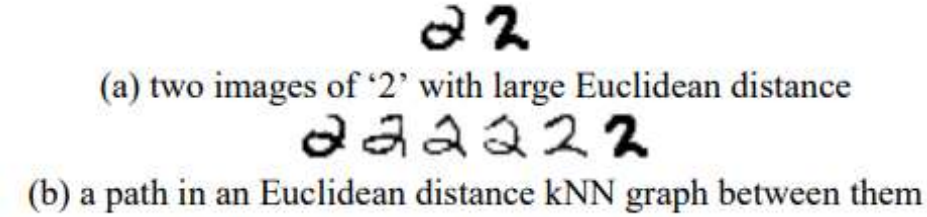
Порівняно з розглянутими вище способами граф b-узгодження є збалансованим або регулярним. Тобто з кожної вершини графу виходить рівно b ребер. Це відіграє ключову роль при навчанні на нерівномірно розподілених даних. Крім того, завдяки швидшій реалізації алгоритму поширення переконань (belief propagation) метод можна використовувати на великомасштабних даних без критичного зростання часу виконання навчання.

Задачу оптимізації методу можна представити у наступному вигляді:

$$\min_{P \in B} \sum_{ij} P_{ij} D_{ij}$$

$$\sum_{j} P_{ij} D_{ij} = k, P_{ii} = 0, P_{ij} = P_{ji} \;\; \forall i, j \;\; \in \overline{1, n}$$

Однією з переваг графових методів є можливість з'єднання двох несхожих між собою (якщо порівнювати значення функції, що оцінює подібність вершин), але насправді однакових з точки зору людини, екземплярів через ланцюг з інших елементів датасету. Наприклад, на рис. 2.1(a), бачимо, два зображення, які є екземплярами одного датасету та позначають число "два", але мають велику евклідову відстань, що обчислюється як корінь від суми квадратів різниць між координатами точок (усім відома формула зі школи). Але не зважаючи на це, вони належать до одного класу "двійки". Це відбувається завдяки тому, що між цими вершинами, хоч і відсутній пряме з'єднання у вигляді ребра, зате є шлях (рис. 2.1(b)) через інші сусідні вершини, що сприяє поширенню класу до обох екземплярів.



(a) two images of '2' with large Euclidean distance

(b) a path in an Euclidean distance kNN graph between them

Рис. 2.1 - Шлях у графу між двома несхожими на перший погляд елементами[11].

3. Перевизначення ваг після спарсифікації.

Після побудови розрідженого графу замість повнозв'язного необхідно переоцінити ваги ребер. Найпростіший спосіб – це двійковий підхід, коли вага між двома будь-якими з'єднаними ребром вершинами дорівнює 1, і 0 між не з'єднаними вершинами, відповідно. Але такий спосіб може бути неефективним, якщо спарсифікація графу не до кінця правильна.

Іншим підходом є використання функції Гауса у вигляді[16]:

$$W_{ij} = P_{ij} \exp\left(-\frac{d(x_i, x_j)}{2\sigma^2}\right)$$

Де $d(x_i, x_j)$ – функція, яка визначає подібність двох вершин графу, σ – параметр пропускної здатності.

Насправді, для оцінки ваг ребер можна використовувати різні функції, в тому числі норми, різницю по модулю, тангенс і т. д.

4. Поширення позначень.

Головною ціллю напівкерованого навчання на основі графів є поширення позначень від первинно позначених даних до непозначених. Створення стійкого алгоритму поширення все ще залишається важливою і актуальною проблемою у наш час.

Матрицю ймовірності переходу позначення від вершини i до вершини j можна описати наступним чином[11]:

$$P_{ij} = P(i \to j) = \frac{w_{ij}}{\sum_k w_{ik}}$$

Згідно формули, позначення передається до найближчого сусіда.

Якщо f відповідає позначенням непозначених даних, то алгоритм поширення можна представити:

1) поширити позначення $f \leftarrow Pf$;

2) закріпити початкові позначення $f_L = Y_L$;

3) повторювати, поки f не зійдеться.

Поширення позначень досягається шляхом мінімізації функції вартості[17], яка передбачає компроміс між двома функціями, одна з них

характеризує плавність переходу позначень між вузлами, тобто узгодженість двох близьких з'єднаних точок, а друга – точність передбачення класів точок в порівнянні із заздалегідь промаркованими даними.

Для цього використовуються різні методи, в тому числі метод гаусових полів і гармонійних функцій (GFHF – Gaussian fields and harmonic functions), метод локальної та глобальної узгодженості (LGC - local and global consistency) та метод трансдукції графу за допомогою змінної мінімізації (GTAM – Graph Transduction via Alternating Minimization).

Методи GFHF та LGC, як зазначалося вище, визначають функцію вартості, яка залежить від узгодженості позначень у зв'язаних вузлах Q_{smooth} та точності передбачених класів Q_{fit}[17]. Оптимальне рішення $F - F^*$ досягається шляхом мінімізації функції Q:

$$F^* = \arg\min_{F \in \mathbb{R}^{n*c}} Q(F) = \arg\min_{F \in \mathbb{R}^{n*c}} \{Q_{smooth}(F) + Q_{fit}(F)\}$$

Для методу локальної та глобальної узгодженості це функція Q набуває наступного вигляду:

$$Q^{LGC}(F) = \frac{1}{2}\left(tr(F^T L F) + \mu(\|F - Y\|^2)\right)$$

де tr – trace, функція яка обчислюється як сума елементів діагоналі, L – Лапласіан графу, Y – матриця початково позначених вершин. Перший доданок характеризує узгодженість, другий – точність.

Метод GFHF ж обмежується лише першою частиною функції:

$$Q^{\mathrm{GFHF}}(F) = \frac{1}{2} tr(F^T L F)$$

Обидва вищезазначені методи поширення міток залежать від одновимірної функції та трактують класифікаційну функцію F як єдину змінну, що розглядається. Ці методи надзвичайно чутливі вибору початково позначених даних та шуму. Щоб зменшити залежність від початкових міток, було запропоновано оптимізувати не лише класифікаційну функцію, а й передбачені значення класів. Тоді Q набуває наступного вигляду:

$$Q^{GTAM}(F) = \min_{\substack{F\in\mathbb{R}^{|V|*c}\\ Y\in\mathbb{R}^{|V|*c}}} tr\left(F^T LF + \mu(F-VF)^T(F-YF)\right)$$

де V — матриця регуляризації міток, яка характеризує важливість позначень і співвідношення класів. Метод трансдукції графу за допомогою змінної мінімізації.

2.3 Постановка задачі напівкерованого навчання нейронних мереж на основі графів

Як вже зазначалося задачею напівкерованого навчання є класифікація, або позначення даних з вибірки за допомогою невеликої кількості вже позначених даних. Перші називаються soft labeled даними, другі – hard labeled. Досягається це шляхом поширення позначень від маркованих екземплярів до немаркованих та пошуком зв'язків, або патернів між даними в цілому. Розглядаючи графовий метод, у разі правильної класифікації або кластеризації, елементи одного класу мають знаходитися в одному й тому ж кластері вершин графу.

Недоліком графового підходу є те, що це першочергово трансдуктивний метод, тобто для класифікації нового екземпляра, взятого не з навчального датасету, необхідно перебудовувати граф заново, що не є оптимально. Задачею даної роботи було розглянути використання нейронної мережі як кінцевого індуктивного рішення. Тобто повністю класифікований графовим методом датасет можна подати на вхід глибокої нейронної мережі та застосувати до неї методи керованого навчання, що дасть змогу використовувати її для класифікації екземплярів, які були відсутні у навчальній вибірці.

Загалом задачу напівкерованого навчання нейронних мереж на основі графів можна описати наступним чином:

1) Нехай $\{(x_1, y_1), \dots, (x_l, y_l)\}$ – позначені дані, де x_i – відповідає вектору характеристик конкретного екземпляру i, це може бути, наприклад, координати точок, пікселі зображення, слова тексту, тощо, а y_i – відповідно

клас до якого належить екземпляр i, $y_i \in \{1, \dots, C\}$, де C – кількість класів; а $\{(x_{l+1}, y_{l+1}), \dots, (x_{l+u}, y_{l+u})\}$ – множина непозначених даних. При чому $l \ll u$ $(l - labeled, u - unlabeled)$.

2) Нехай для оцінки подібності (ваг ребер між елементами) двох елементів використовуватимемо функцію Гауса:

$$w_{ij} = \exp\left(-\gamma \left\| x_i - x_j \right\|^2\right), \qquad \text{де } \gamma > 0$$

За допомогою функції подібності необхідно оцінити відстань між кожними двома елементами датасету, а потім на основі цього побудувати оптимальний розріджений граф, використовуючи підходящий з вище наведених методів: $\varepsilon NN,\ kNN,\ b - matching$.

3) Використати алгоритм поширення позначень для позначення немаркованих даних з датасету $\{(x_{l+1}, y_{l+1}), \dots, (x_{l+u}, y_{l+u})\}$.

4) Оцінити результати застосування графового методу для бінарної класифікації даних. Порівняти різні параметри, алгоритми та обрати оптимальні.

5) Побудувати глибоку (більше ніж з одним прихованим шаром) нейронну мережу прямого поширення та навчити її методом зворотного поширення похибки:

$$\theta^{t+1} = \theta^t - \frac{\alpha \partial E(X, \theta^t)}{\partial \theta}$$

де $E(X, \theta^t)$ – функція помилки, яка визначає різницю між бажаним і реальним виходом нейронної мережі, та в найпростішому випадку є середньоквадратичною похибкою датасету; θ^t – конкретне значення параметру в момент t градієнтного спуску, наприклад значення одного з коефіцієнтів ваг, X – множина пар вхід-вихід (x_i, y_i).

6) Оцінити результати роботи мережі на валідаційних даних, налаштувати і обрати кращу модель нейронної мережі, визначити оптимальний коефіцієнт навчання мережі. Порівняти ефективність використання графового методу на даних різного виду.

2.4 Метрики оцінювання роботи алгоритму

Спочатку розглянемо поняття матриці помилок (невідповідностей), або з англомовної літератури Confusion matrix. Вона має вигляд представлений у таблиці 2.1.

Табл. 2.1. Матриця помилок[18].

	Справжні значення		
Передбачені значення		Positive (1)	Negative(0)
	Positive (1)	TP	FP
	Negative(0)	FN	TN

TP – означає True Positive і вказує на те, що передбачене значення як позитивне справді є позитивним.

TN – означає True Negative, передбачене значення як негативне справді є негативним.

FP і FN – False Positive та False Negative, описують значення які були передбачені хибно, наприклад, справді негативне значення передбачене як позитивне і навпаки.

В даній роботі матриця помилок використовуватиметься для оцінки кількості правильно передбачених значень, елементи матриці відповідають кількості передбачень певного виду: True Positive, True Negative, False Positive, False Negative.

Крім того, для оцінки адекватності моделей в роботі використовуватимуться наступні величини:

а) Точність (accuracy)

Характеризує загальну кількість успішно класифікованих елементів. Визначається за формулою:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{\text{К} - \text{сть правильних прогнозів}}{\text{Загальна к} - \text{сть прогнозів}}$$

Під прогнозами маються на увазі прогнозовані, або, іншими словами, класифіковані моделлю елементи.

б) Влучність (precision)

$$Precision = \frac{TP}{TP + FP}$$

Описує скільки з позитивно передбачених класів елементів, насправді позитивні.

в) Повнота (recall)

$$Precision = \frac{TP}{TP + FN}$$

Характеризує скільки елементів позитивного класу було передбачено правильно.

Влучність і повнота обчислюється аналогічно і для елементів негативного класу.

г) F1-score

$$F1\text{-}score = \frac{2 * Recall * Precision}{Recall + Precision}$$

F1-score дозволяє одночасно оцінити $Recall$ і $Precision$ та порівнювати моделі для яких один з цих параметрів високий, а інший низький.

Для оцінки нейронної мережі під час навчання використовується точність та функція втрат, яка має наближатися до нуля. В даній роботі в якості такої функції виступає функція втрат перехресної ентропії, яка обчислюється за наступною формулою[19]:

$$Loss = -\sum_{i=1}^{\substack{output \\ size}} y_i * log\widehat{y_i}$$

де y_i – справжній ймовірнісний розподіл, $\widehat{y_i}$ – передбачений моделлю, відповідно.

2.5 Висновки до другого розділу

В цьому розділі було розглянуто постановку задачі напівкерованого навчання та етапи застосування графового методу, який складається з вибору функції подібності, спарсифікації, або розрідження створеного повнозв’язного графу, оскільки алгоритм зацікавлений у збереженні зв’язків – ребер лише між подібними вершинами, для цього можна використовувати різні методи: метод узгодження, епсилон-околу або метод k найближчих сусідів, який є одним з поширених методів. Опісля, ваги ребер вже нового графу необхідно переоцінити, для цього застосовуються різноманітні функції, такі як тригонометричні, експоненційні, різні норми і т. п. Останнім етапом є поширення міток вздовж ребер від початково визначених вершин. Популярними підходами, які також були розглянуті у розділі, є метод гаусових полів і гармонійних функцій (GFHF – Gaussian fields and harmonic functions), метод локальної та глобальної узгодженості (LGC - local and global consistency) та метод трансдукції графу за допомогою змінної мінімізації (GTAM – Graph Transduction via Alternating Minimization).

Як показує практика, немає однозначно кращого підходу чи методу, кожен з них може бути застосований до вирішення притаманної йому задачі.

РОЗДІЛ 3 ПОРІВНЯННЯ ТА ОЦІНКА РЕЗУЛЬТАТІВ НАПІВКЕРОВАНОГО НАВЧАННЯ НЕЙРОННИХ МЕРЕЖ НА ОСНОВІ ГРАФІВ

3.1 Обґрунтування вибору засобів програмного забезпечення

3.1.1 Мова програмування Python

Python – популярна інтерпретована об’єктно-орієнтована мова програмування, яка включає в себе безліч корисних та зручних бібліотек для обчислень, машинного навчання, штучного інтелекту, візуалізації даних тощо. Деякі з них використовуються в даній роботі, а саме Pandas, TensorFlow, Numpy, Matplotlib. Крім того Python має простіший синтаксис в порівнянні, наприклад, з Java та велику спільноту розробників і обширну документацію. Може застосовуватися в найрізноманітніших сферах: машинному навчанні, кібербезпеці, розробці вебзастосунків, ігор, статистиці, тощо.

3.1.2 TensorFlow

TensorFlow – одна з відкритих Python бібліотек, що використовується для задач машинного навчання, в тому числі, написання нейронних мереж. Містить безліч реалізованих функцій, що дозволяє будувати нейронні мережі різних типів, з різними функціями активації, шарами, нормалізаціями, оптимізаторами та налаштуваннями, при цьому надає можливість візуалізації навчання нейронної мережі. TensorFlow підходить для обробки даних великого обсягу, та інтегрується з іншими Python бібліотеками, наприклад, Numpy.

Ще однією з переваг TensorFlow є наявність документації та різних інструкцій щодо реалізацій конкретних задач.

3.1.3 Numpy

Бібліотека Numpy особлива ефективним використанням багаторозмірних векторів та матриць, а також математичними функціями і операціями, які до них застосовуються. Вони включають лінійні оператори, тригонометричні, експоненційні, статистичні та інші функції. Загалом, Numpy представляє різні способи маніпуляції масивами.

3.1.4 Matplotlib

Matplotlib – Python бібліотека для візуалізації даних. З її допомогою можна будувати графіки, гістограми, діаграми та ін. Має зручний для використання інтерфейс та багато можливостей налаштування графіків. Також, Matplotlib інтегрується з іншими бібліотеками, наприклад, Pandas, та дозволяє представляти .csv та табличні дані у вигляді графіків.

3.1.5 Scikit-learn

Scikit-learn – відкрита Python бібліотека призначена для аналізу даних та вирішування задач машинного навчання, що включають в себе: прогнозування, прийняття рішень, категоризацію, класифікацію та інші. Зокрема її перевагою є наявність документації та велика спільнота розробників, як і для більшості бібліотек Python. Інтегрується з іншими бібліотеками: Numpy, TensorFlow та ін.

3.2 Підготовка датасету

Дані, які використовуються в роботі є штучно згенерованими множинами точок у вигляді бананів, місяців та кілець (рис. 3.1 – 3.3). Різні датасети мають різну складність. Датасети для яких множини двох класів (точки різних кольорів) не перетинаються є найпростіші для класифікації, в той час як найбільш поєднанні множини – найскладніші (наприклад банани направлені в одну сторону і майже накладені один на одного). Відповідно, ми

використовуватимемо їх для більш комплексного порівняння ефективності графового методу.

Дані з датасетів представлені у вигляді двовимірних точок з координатами (x_1, x_2) та значенням цільової функції y, яка може приймати значення 0 та 1. Значенню 1 відповідає помаранчевий колір, 0 – синій.

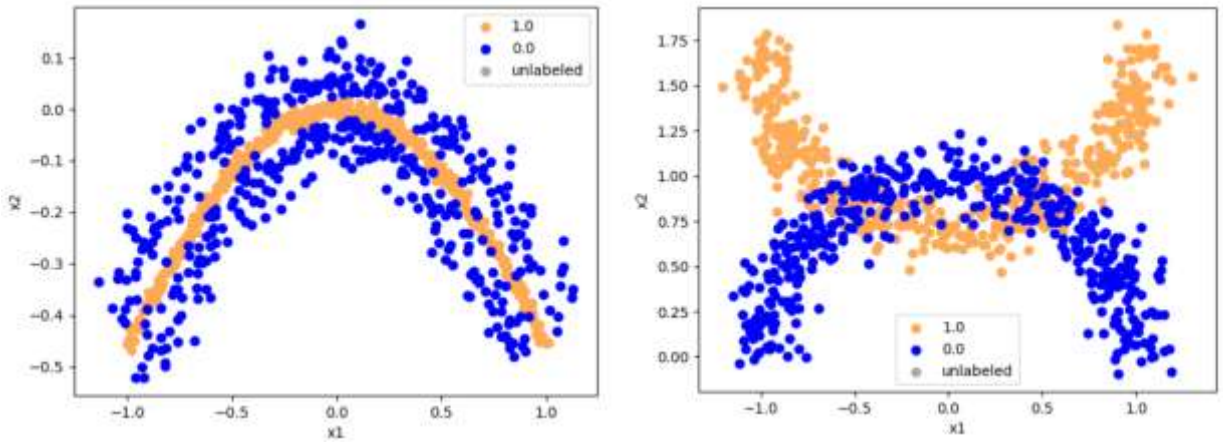


Рис. 3.1 - Множини у вигляді бананів. В першому випадку вони сходяться, в другому – направлені в різні боки.

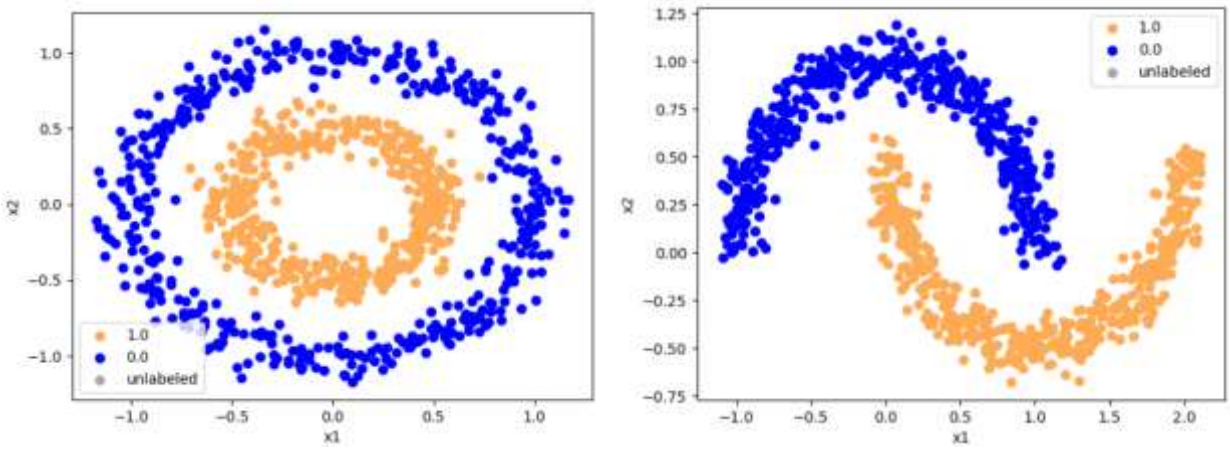


Рис. 3.2 - Множини у вигляді кілець (зліва) та середньовіддалених місяців (справа).

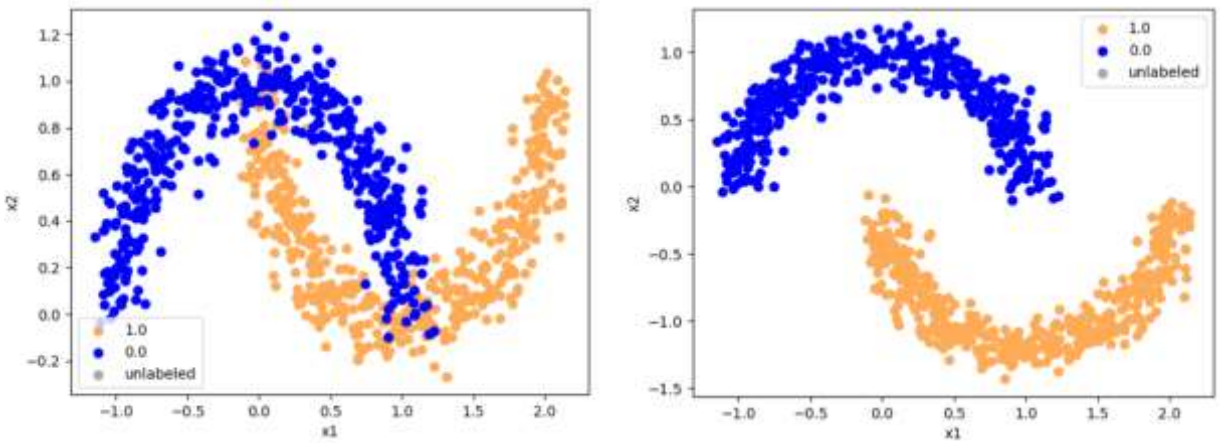


Рис. 3.3 - Множини у вигляді двох місяців, які розташовані близько одне до одного (зліва), та далеко (справа).

Кожний набір даних складається з 900 позначених точок.

Перед початком алгоритму позначення частини елементів прирівнюються до -1, з них формується множина непозначених даних. Кількість таких екземплярів залежить від обраної ймовірності. Для перевірки навчання нейронної мережі у кінці використовується ще додатково 100 точок, які описують ті ж класи, але не використовувалися під час застосування графового методу та навчання нейронної мережі.

3.3 Результати роботи обраних алгоритмів

Оскільки різні набори даних становлять різну складність для графового методу, спершу будуть оцінені результати роботи алгоритму на простіших екземплярах.

3.3.1 Два місяці віддалені

Спершу з ймовірністю 90% у частини випадково перемішаних елементів датасету видаляється клас, таким чином, близько 90 елементів з 900 будуть непозначеними. На рис. 3.4. вони відмічені сірим кольором.

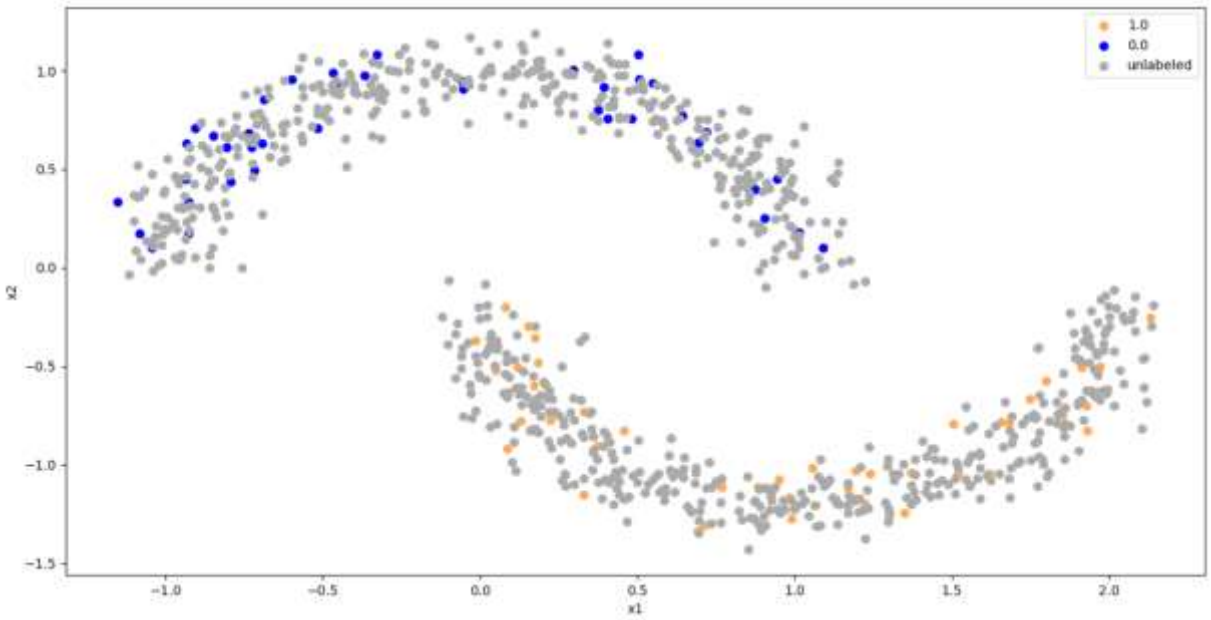


Рис. 3.4 - Візуалізація даних підготовлених до напівкерованого навчання

На рис. 3.5 показано графік межі прийняття рішень графового методу, можна переконатися, що елементи з точністю 100% класифіковані правильно. В якості функції подібності використовувалася евклідова відстань, розріджений граф створений методом k найближчих сусідів, де $k = 7$.

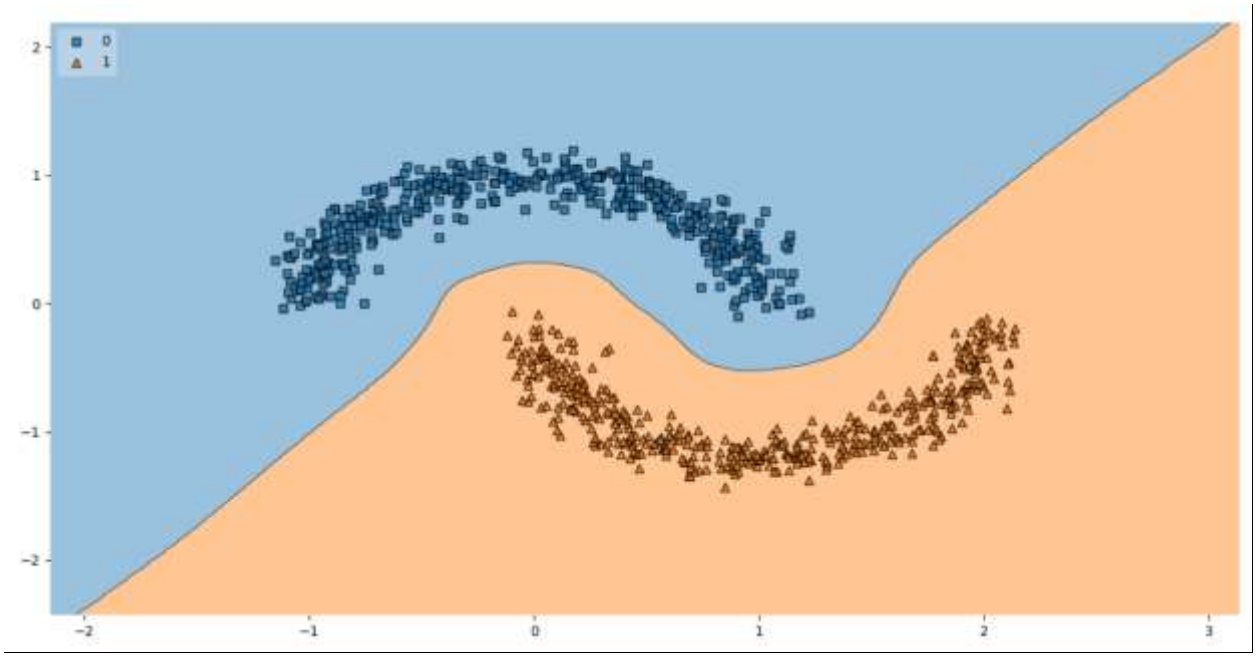


Рис. 3.5 - Графік межі прийняття рішень (графовий метод).

Нижче у таблиці 3.1 наведено деякі метрики щодо оцінки напівкерованого навчання графовим методом.

Табл. 3.1 - Оцінка результатів графового методу.

Клас	Precision	Recall	F1-score	Support
0	1.00	1.00	1.00	402
1	1.00	1.00	1.00	405
Accuracy		1.00		
Всього непозначених даних		807		
Всього позначених даних		92		

Матриця невідповідностей (помилок) має наступний вигляд:

$$\begin{pmatrix} 402 & 0 \\ 0 & 405 \end{pmatrix}$$

що вказує на те, що модель не здійснила жодних хибних прогнозів, усі точки класу 0 були класифіковані правильно, як і класу 1.

Перед початком навчання нейронної мережі дані діляться на навчальну, тестову вибірки у відношенні 80 до 20. Для валідації передбачень нейронної мережі використовуються характеристики з тестової вибірки і їхні відповідні оригінальні класи. Далі необхідно обрати найкращу модель нейронної мережі.

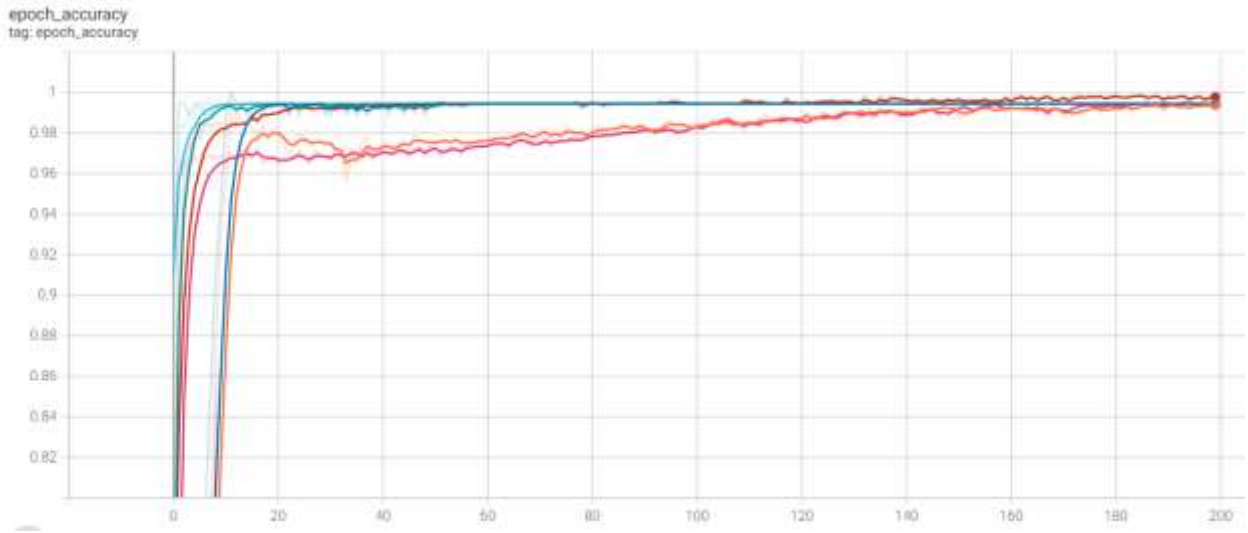


Рис. 3.6 - Графіки точності НМ: ELU, ReLU, Leaky ReLU функції.

На рис. 3.6 зображені графіки точності нейронних мереж в залежності від епох під час навчання, а на рис. 3.7 залежність функції втрат відповідно.

Всі моделі показують гарні результати, оскільки точність наближається до 1, а функція втрат до 0 на тестовому та навчальному наборах.

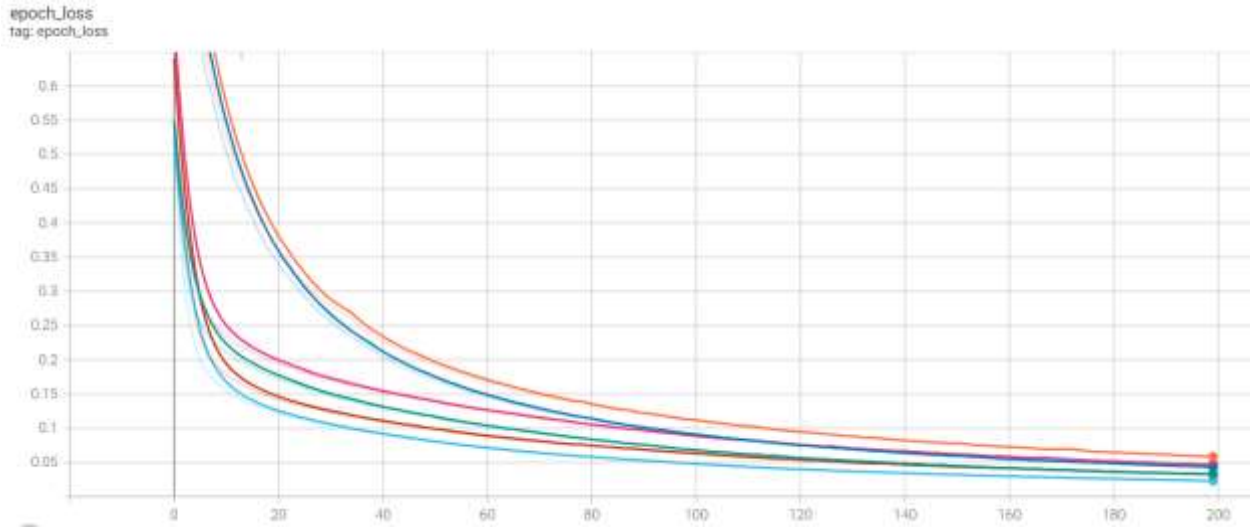


Рис. 3.7 - Графіки функції втрат НМ: ELU, ReLU, Leaky ReLU функції.

В табл. 3.2 наведені деякі характеристики зображених моделей.

Табл. 3.2 - Оцінка навчання нейронної мережі.

Оптимі-затор	Функція активації прихованого шару	Швидкість навчання	Точність		Функція втрат	
			Навч.	Тест.	Навч.	Тест.
Adam	ELU	$2 * 10^{-4}$	0.9936	0.9944	0.0588	0.0435
Adam	Leaky ReLU	$2 * 10^{-4}$	0.9939	0.9944	0.0471	0.0332
Adam	ReLU	$2 * 10^{-4}$	0.9979	0.9944	0.0328	0.0233

З табл. 3.2 оберемо модель з функцією активації ReLU за найкращу. Швидкість навчання моделей була попередньо підібрана шляхом побудови графіку залежності функції втрат від значення швидкості навчання (рис. 3.8). Згідно графіку обрано значення 0.001. Модель збігається за трохи більше 100 епох.

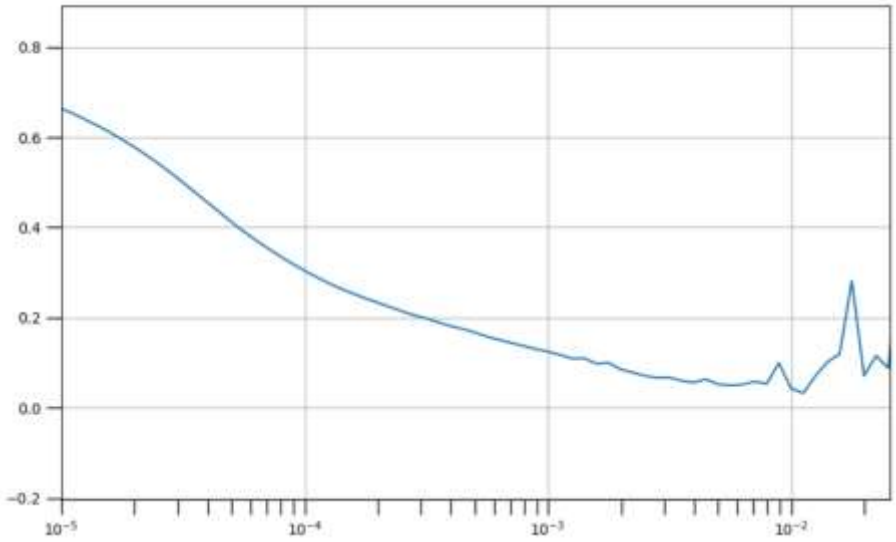


Рис. 3.8 - Графік залежності функції втрат від швидкості навчання.

Перевіримо наскільки класифіковані нейронною мережею дані співпадають з оригінальними позначеннями. В таблиці 3.3 наведені результати оцінки прогнозованих моделлю значень.

Табл. 3.3 - Оцінка результатів класифікації нейронною мережею.

Клас	Precision	Recall	F1-score	Support
0	1.00	0.99	0.99	94
1	0.99	1.00	0.99	85
Accuracy		0.99		
Всього валідаційних даних		179		
Тривалість прогнозування		0.08 с		

Матриця невідповідностей (помилок) має наступний вигляд:

$$\begin{pmatrix} 93 & 1 \\ 0 & 85 \end{pmatrix}$$

Вдалося досягти точності 99% - 100%, модель добре описує даний датасет. Спробуємо застосувати модель для класифікації елементів, які не використовувалися при навчанні, отримаємо точність 100% (табл. 3.4) на вибірці зі 100 елементів.

Табл. 3.4 - Оцінка класифікації нейронною мережею “нових” точок.

Клас	Precision	Recall	F1-score	Support
0	1.00	1.00	1.00	47
1	1.00	1.00	1.00	52
Accuracy		1.00		
Всього валідаційних даних		99		
Тривалість прогнозування		0.03 с		

Можна пересвідчитися, що на чітко розмежованих даних графовому методу достатньо хоча б кілька класифікованих точок в кожному з класів, щоб передбачити класи всіх інших точок. Для перевірки, ймовірність стирання класу була збільшена (рис. 3.9). Внаслідок експерименту отримано наступний графік межі прийняття рішень (рис. 3.10), з якого видно, що 100% елементів класифіковані правильно.

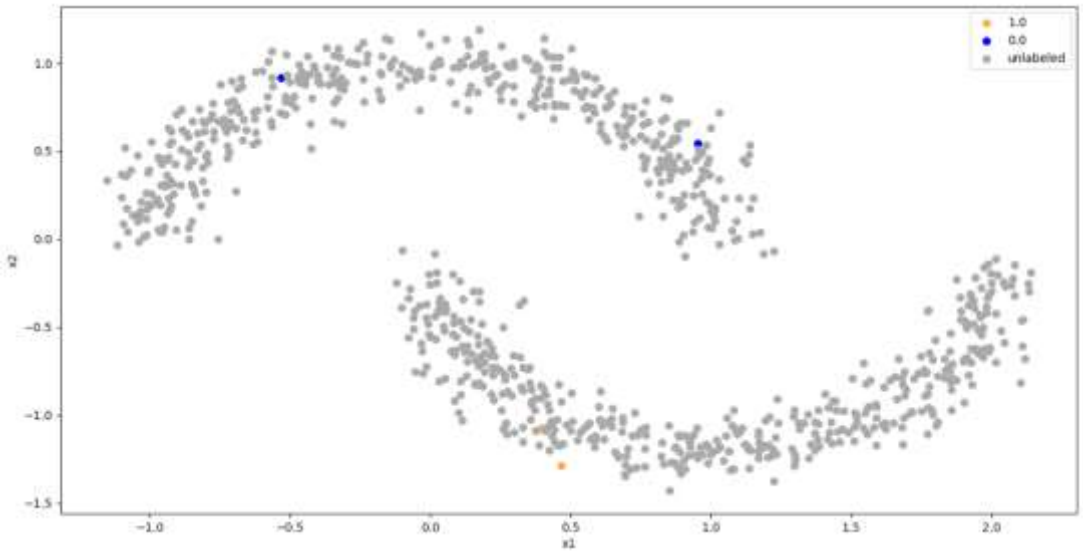


Рис. 3.9 - Вигляд даних з ймовірністю стирання позначення 99%

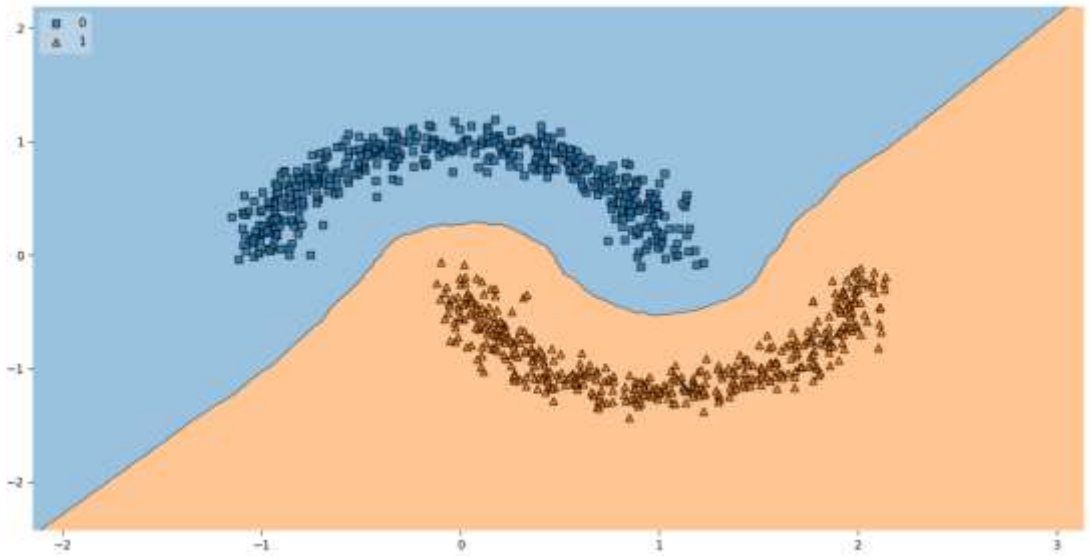


Рис. 3.10 - Графік межі прийняття рішень.

Отже, графовий метод на kNN ідеально підходить для використання, коли класи досліджуваних елементів є чітко розмежовані, адже поширення позначень не може вийти за межі класу.

3.3.2 Два місяці середньо віддалені

Спробуємо застосувати графовий метод та обрану у попередньому пункті нейронну мережу на датасеті з місяцями, що розташовані ближче, ніж у попередньому (рис. 3.11).

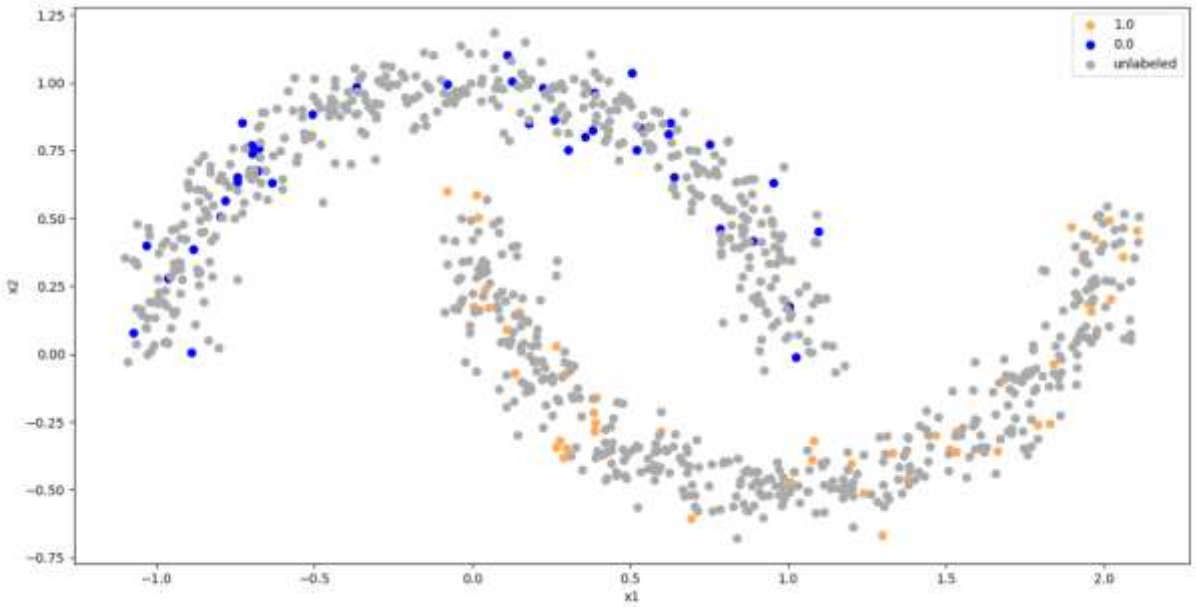


Рис. 3.11 - Візуалізація даних підготовлених до напівкерованого навчання

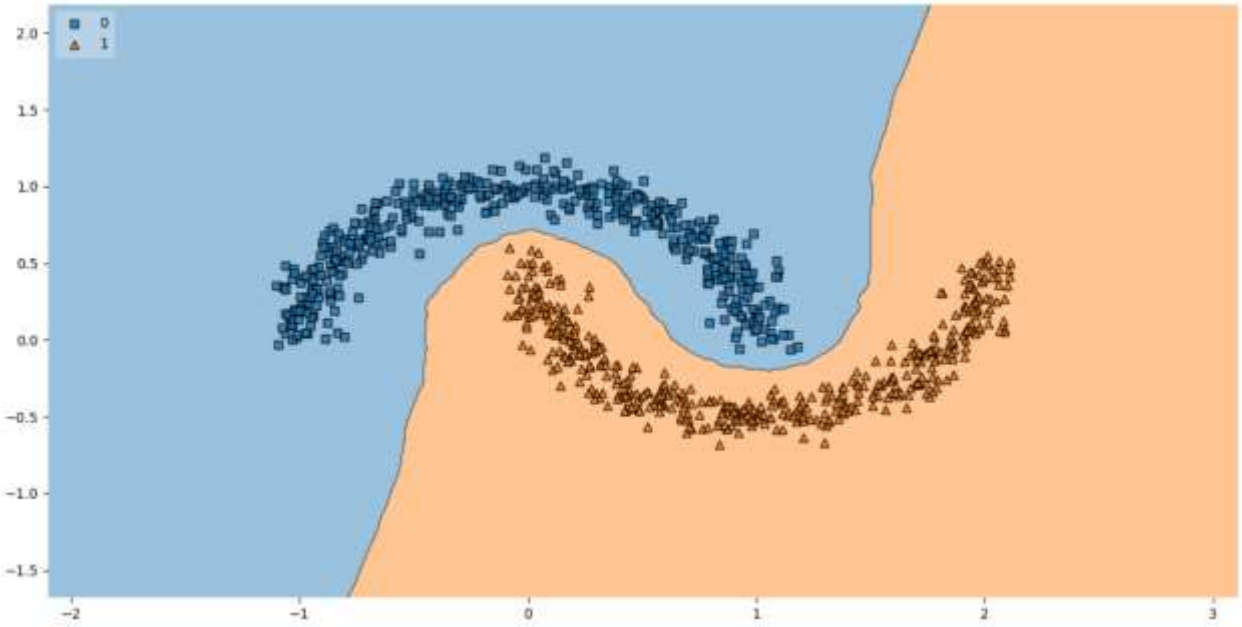


Рис. 3.12 - Графік межі прийняття рішень (графовий метод).

З рис. 3.12 та таблиці 3.5 видно, що графовий метод успішно справився з класифікацією даних, класи яких розташовані ближче одне до одного з точністю 100%.

Табл. 3.5 - Оцінка результатів графового методу

Клас	Precision	Recall	F1-score	Support
0	1.00	1.00	1.00	411
1	1.00	1.00	1.00	396
Accuracy		1.00		
Всього непозначених даних		807		
Всього позначених даних		92		

Матриця невідповідностей (помилок) має наступний вигляд:

$$\begin{pmatrix} 411 & 0 \\ 0 & 396 \end{pmatrix}$$

З рис. 3.13-3.14 можна побачити, що модель нейронної мережі з попереднього пункту погано справляється з даним датасетом (червоний та синій графіки), тому до неї було додано ще один прихований шар і проведено

деяке дослідження залежності точності моделі від активаційних функцій прихованих шарів.

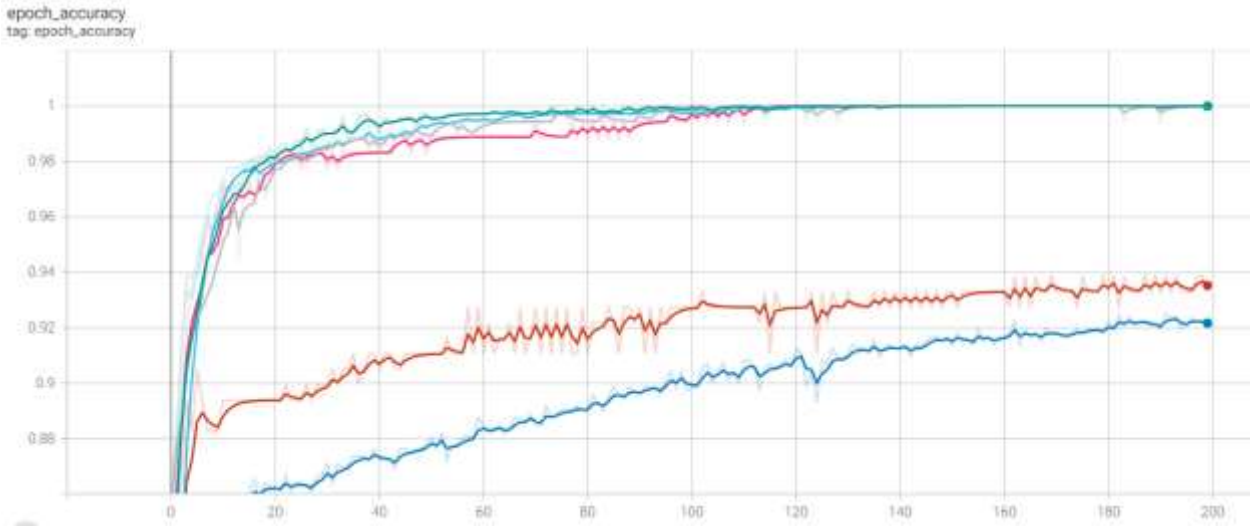


Рис. 3.13 - Графіки точності НМ: ReLU, Leaky ReLU функції та попередньої НМ.

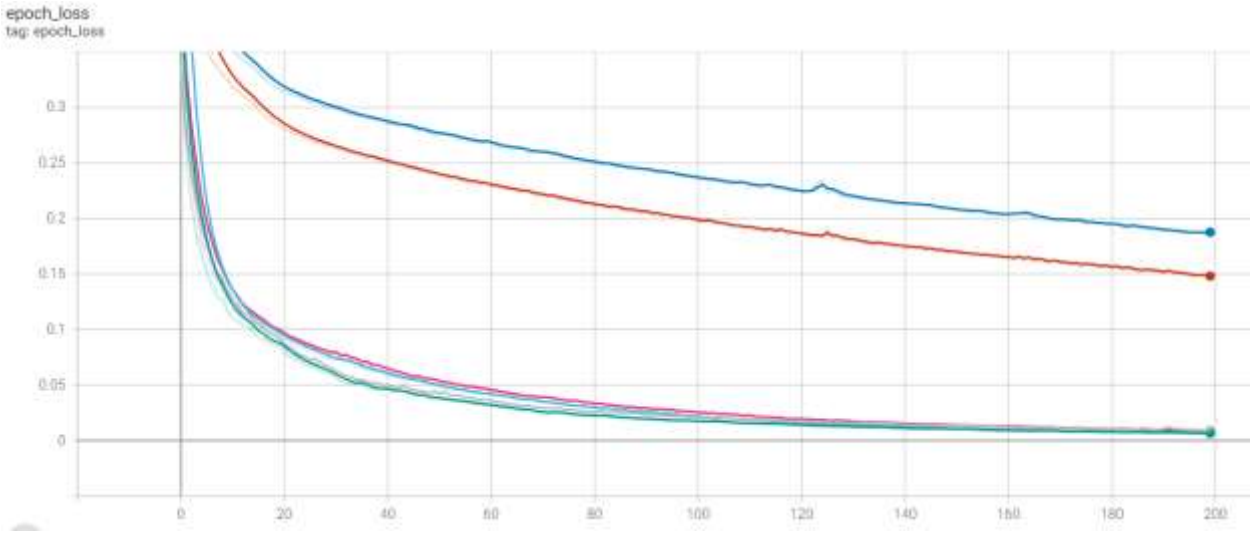


Рис. 3.14 - Графіки функції втрат НМ: ReLU, Leaky ReLU функції та попередньої НМ.

В табл. 3.6 наведені деякі характеристики зображених моделей. В таблиці наведена точність і функція витрат останньої епохи.

Табл. 3.6 - Оцінка навчання нейронної мережі.

К-сть прихова-них шарів	Функція активації прихованого шару	Швидкість навчання	Точність		Функція втрат	
			Навч.	Тест.	Навч.	Тест.
2	ReLU	$2 * 10^{-4}$	0.9217	0.9353	0.1876	0.1481
3	Leaky ReLU	$2 * 10^{-4}$	1	1	≈0	≈0
3	ReLU	$2 * 10^{-4}$	1	1	≈0	≈0

Оберемо модель з трьома прихованими шарами та функцією активації ReLU за кращу. У таблиці 3.7 наведено результати класифікації моделлю тестової вибірки.

Табл. 3.7 - Оцінка результатів класифікації нейронною мережею.

Клас	Precision	Recall	F1-score	Support
0	1.00	1.00	1.00	90
1	1.00	1.00	1.00	89
Accuracy		1.00		
Всього валідаційних даних		179		
Тривалість прогнозування		0.08 с		

Матриця невідповідностей (помилок) має наступний вигляд:

$$\begin{pmatrix} 90 & 0 \\ 0 & 89 \end{pmatrix}$$

У табл. 3.8 наведені результати класифікації елементів, які не використовувалися у навчанні. Знову отримали точність 100%.

Табл. 3.8 - Оцінка класифікації нейронною мережею “нових” точок.

Клас	Precision	Recall	F1-score	Support
0	1.00	1.00	1.00	46
1	1.00	1.00	1.00	53
Accuracy		1.00		
Всього валідаційних даних		99		
Тривалість прогнозування		0.03 с		

3.3.3 Кільця

Застосуємо модель з попереднього пункту на датасеті з кільцями. На рис. 3.15 зображено графік межі прийняття рішень графового методу, а табл. 3.9 метрики оцінки правильності моделі.

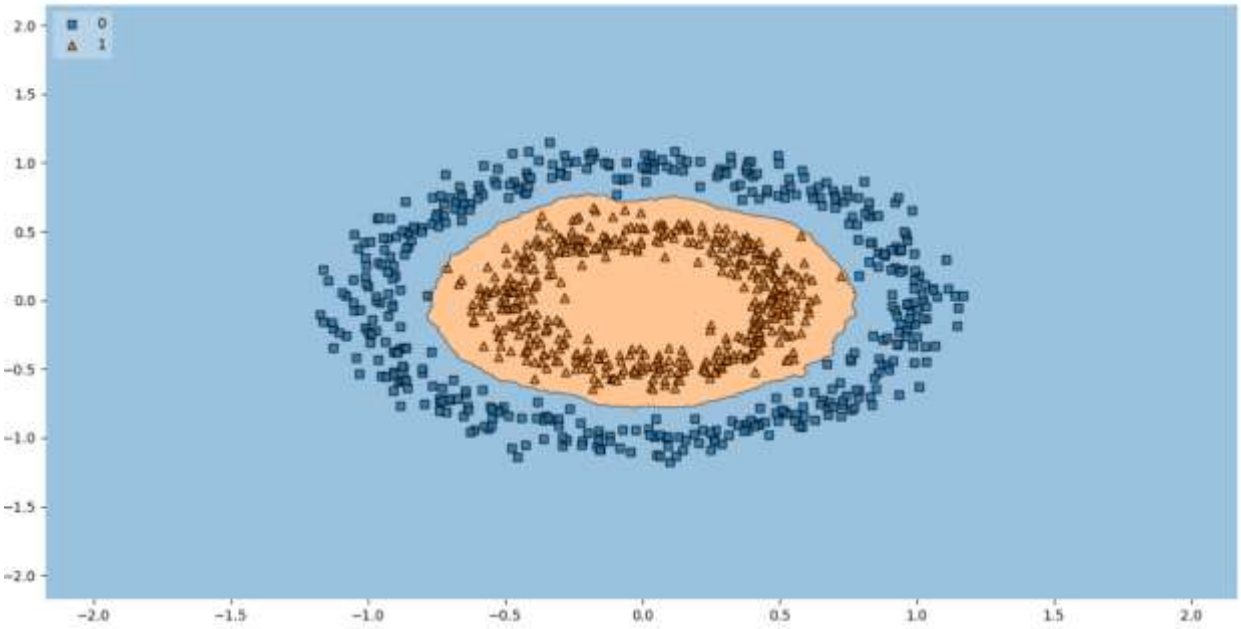


Рис. 3.15 - Графік межі прийняття рішень (графовий метод).

Табл. 3.9 - Оцінка результатів графового методу

Клас	Precision	Recall	F1-score	Support
0	1.00	1.00	1.00	413
1	1.00	1.00	1.00	394
Accuracy		1.00		
Всього непозначених даних		807		
Всього позначених даних		92		

Матриця невідповідностей має наступний вигляд:

$$\begin{pmatrix} 413 & 0 \\ 0 & 394 \end{pmatrix}$$

З рис. 3.16 видно, що функція втрат збігається до нуля, а отже модель нейронної мережі підходить для класифікації датасету. Точність на навчальній вибірці, як і на тестовій на останній ітерації (епосі) сягає близько 1, функція втрат на навчальній сягає 0.015, тестовій – 0.013.

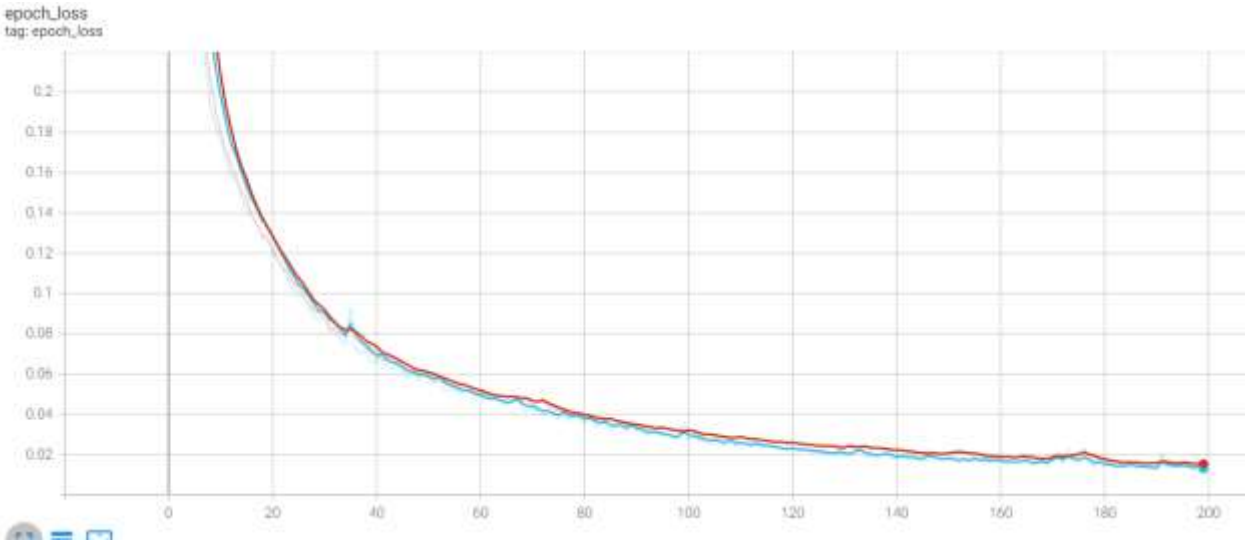


Рис. 3.16 - Графіки функції втрат НМ: навчальна та тестова вибірки.

У таблиці 3.10 наведено результати класифікації моделлю тестової вибірки.

Табл. 3.10 - Оцінка результатів класифікації нейронною мережею.

Клас	Precision	Recall	F1-score	Support
0	1.00	1.00	1.00	80
1	1.00	1.00	1.00	99
Accuracy		1.00		
Всього валідаційних даних		179		
Тривалість прогнозування		0.085 с		

Матриця помилок має наступний вигляд:

$$\begin{pmatrix} 80 & 0 \\ 0 & 99 \end{pmatrix}$$

Табл. 3.11 - Оцінка класифікації нейронною мережею “нових” точок.

Клас	Precision	Recall	F1-score	Support
0	1.00	1.00	1.00	44
1	1.00	1.00	1.00	55
Accuracy		1.00		
Всього валідаційних даних		99		
Тривалість прогнозування		0.03 с		

3.3.4 Два місяці наближені

З рис. 3.17 можна помітити, що позначення на межах стику двох класів поширюються зі складнощами, оскільки точки в таких місцях не мають чіткої структури і точки різних класів виявляються змішаними (рис. 3.3). На рис. 3.18 зображені класифіковані графовим методом точки разом з первинно позначеними. Класи майже не перетинаються.

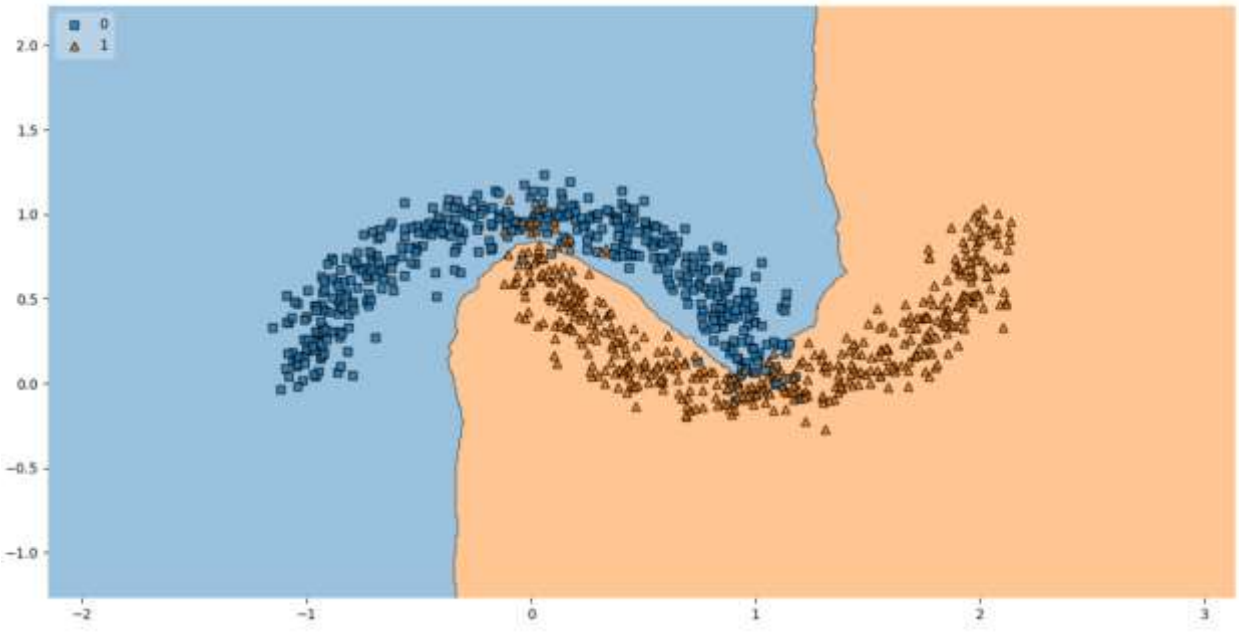


Рис. 3.17 - Графік межі прийняття рішень (графовий метод).

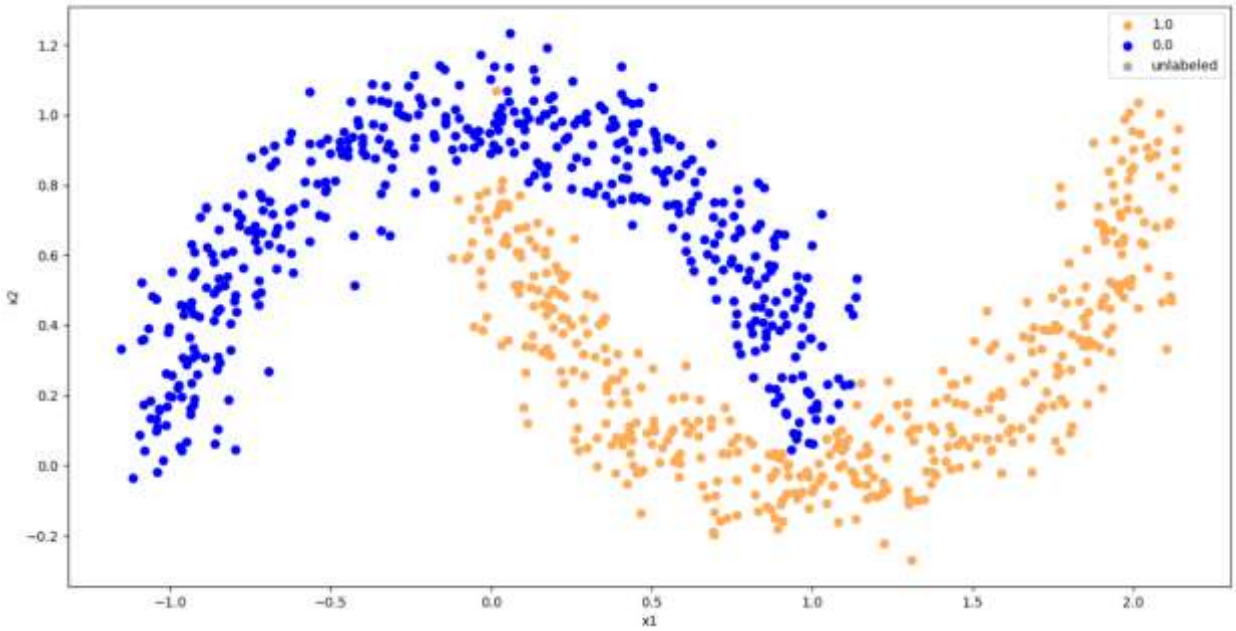


Рис. 3.18 - Класифіковані графовим методом дані.

У таблиці 3.12 наведено результати класифікації непозначених даних графовим методом. В залежності від непозначеної і позначеної вибірок, точність може як досягати 96%, так і бути 94%, що свідчить про важливість вибору початкових позначених даних. В таблиці, як і на графіках, наведено конкретний результат виконання.

Табл. 3.12 - Оцінка результатів графового методу

Клас	Precision	Recall	F1-score	Support
0	0.95	0.96	0.96	414
1	0.95	0.95	0.95	393
Accuracy		0.95		
Всього непозначених даних		807		
Всього позначених даних		92		

Матриця помилок має наступний вигляд:

$$\begin{pmatrix} 396 & 18 \\ 19 & 374 \end{pmatrix}$$

Навіть із збільшенням кількості позначених даних до 30% точність не сягає більше 96%.

Результати навчання нейронної мережі наведені на рис. 3.19 та в таблиці 3.13.

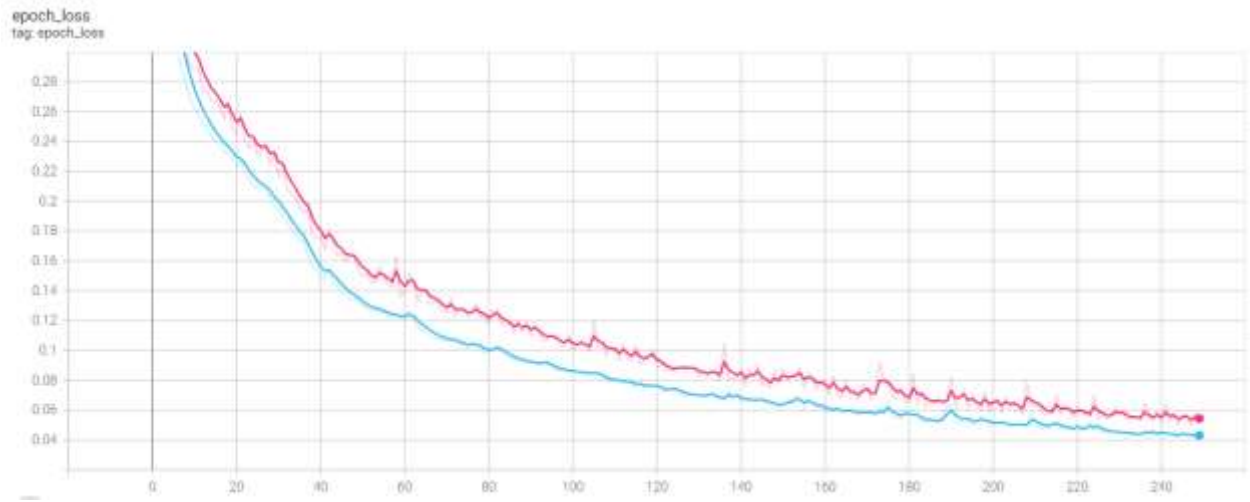


Рис. 3.19 - Графіки функції втрат НМ: навчальна (блакитний) та тестова (рожевий) вибірки.

Не зважаючи на те, що на останніх ітераціях точність нейронної мережі більша 99%, а функція втрат наближається до нуля, точність класифікації такою мережею на валідаційних даних лише 95%, що все ще є гарним результатом, враховуючи, що неправильне початкове позначення

немаркованих даних графовим методом могло призвести до неправильного навчання нейронної мережі.

Табл. 3.13 - Оцінка результатів класифікації нейронною мережею.

Клас	Precision	Recall	F1-score	Support
0	0.96	0.95	0.95	95
1	0.94	0.95	0.95	84
Accuracy		0.95		
Всього валідаційних даних		179		
Тривалість прогнозування		0.086 с		

Матриця помилок має наступний вигляд:

$$\begin{pmatrix} 90 & 5 \\ 4 & 80 \end{pmatrix}$$

Вона вказує на те, що з 95-и елементів 0-го класу п’ять були класифіковані як елементи 1-го класу, тоді як 4 елементи 1-го класу нейронна мережа визначила як 0-ий клас.

Табл. 3.14 - Оцінка класифікації нейронною мережею “нових” точок.

Клас	Precision	Recall	F1-score	Support
0	0.92	0.96	0.94	47
1	0.96	0.92	0.94	52
Accuracy		0.94		
Всього валідаційних даних		99		
Тривалість прогнозування		0.03 с		

Матриця помилок: $\begin{pmatrix} 45 & 2 \\ 4 & 48 \end{pmatrix}$.

3.3.5 “Банани” різнобічно направлені

На рис. 3.20 зображено графік межі прийняття рішень для датасету. Як бачимо, в місцях стику двох класів знову виникають складнощі. Точки не можуть обрати “правильних” k сусідів, в результаті змішуються.

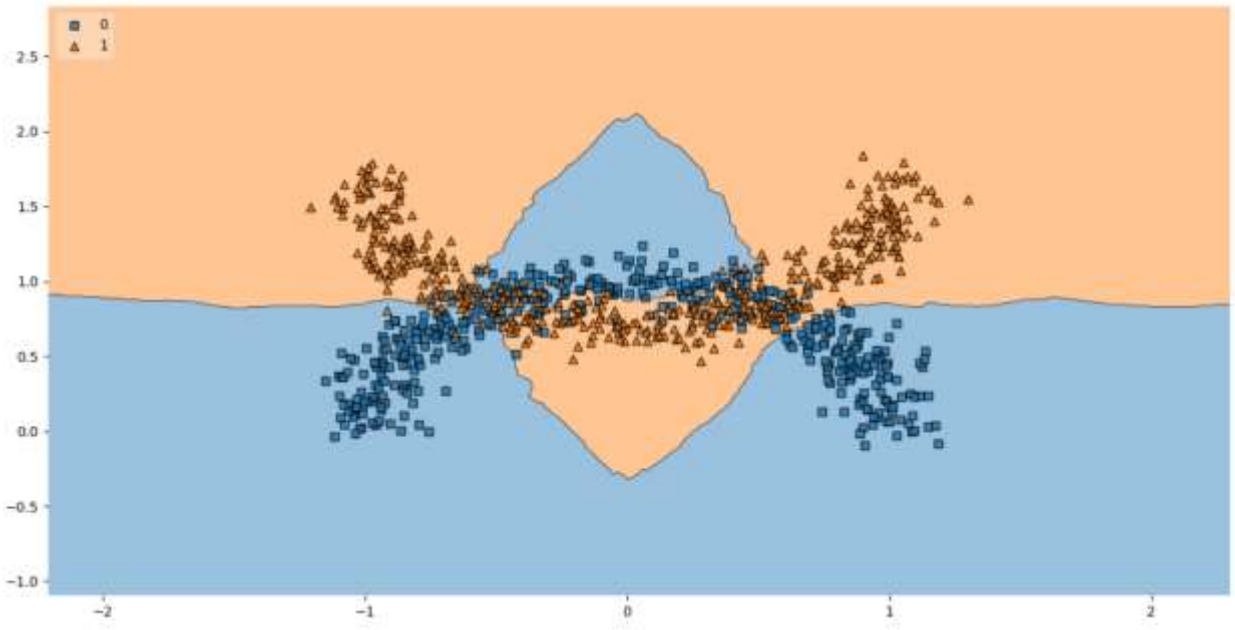


Рис. 3.20 - Графік межі прийняття рішень (графовий метод).

В таблиці 3.15 наведено результати напівкерованого навчання графовим методом.

Табл. 3.15 - Оцінка результатів графового методу

Клас	Precision	Recall	F1-score	Support
0	0.88	0.79	0.83	404
1	0.81	0.89	0.85	403
Accuracy		0.84		
Всього непозначених даних		807		
Всього позначених даних		92		

Матриця помилок має наступний вигляд:

$$\begin{pmatrix} 321 & 83 \\ 44 & 359 \end{pmatrix}$$

Було досліджено вплив кількості найближчих сусідів на ефективність графового методу, але точність не перевищує 85% для будь-якого значення. Також кращих результатів не показує застосування повнозв'язного графу з функцією Гауса.

На рис. 3.21 зображено графік залежності функції втрат від епохи навчання нейронної мережі. Як бачимо, нейронній мережі складніше описати дані даного характеру.

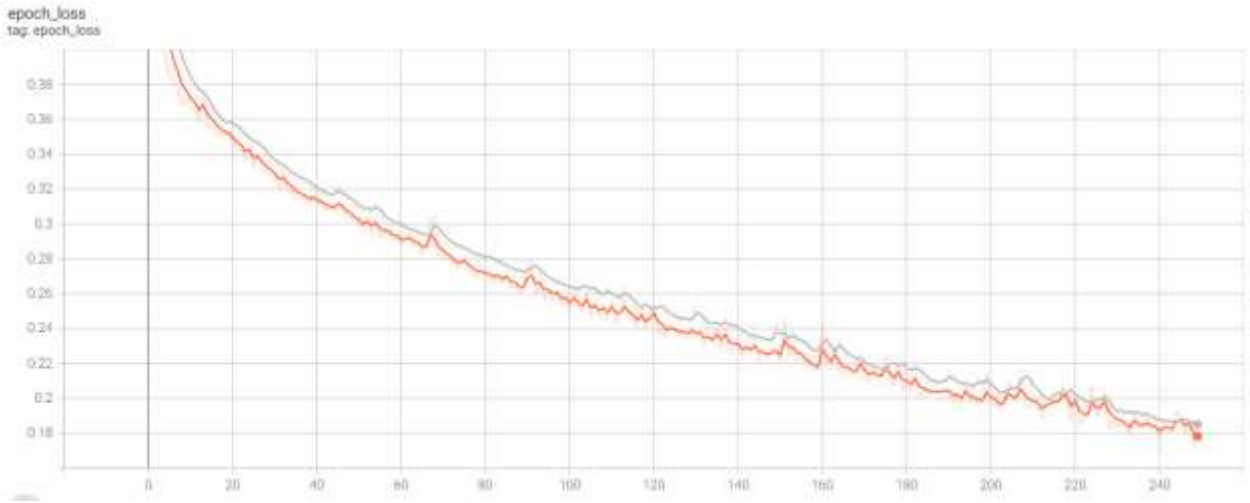


Рис. 3.21 - Графіки функції втрат НМ: навчальна (помаранчевий) та тестова (сірий) вибірки.

Табл. 3.16 - Оцінка результатів класифікації нейронною мережею.

Клас	Precision	Recall	F1-score	Support
0	0.78	0.95	0.86	91
1	0.93	0.73	0.82	88
Accuracy		0.84		
Всього валідаційних даних		179		
Тривалість прогнозування		0.086 с		

Матриця помилок має наступний вигляд:

$$\begin{pmatrix} 86 & 5 \\ 24 & 64 \end{pmatrix}$$

Табл. 3.17 - Оцінка класифікації нейронною мережею “нових” точок.

Клас	Precision	Recall	F1-score	Support
0	0.85	0.85	0.85	53
1	0.83	0.83	0.83	46
Accuracy		0.84		
Всього валідаційних даних		99		
Тривалість прогнозування		0.03 с		

Матриця помилок: $\begin{pmatrix} 45 & 8 \\ 8 & 38 \end{pmatrix}$ вказує на те, що 16 точок з 99 були класифіковані неправильно, що, на мою думку, свідчить про недоцільність об’єднання графового методу з навчанням нейронної мережі для індуктивності алгоритму.

3.3.6 Набір точок “банан”

Використання kNN, де $k = 10$, як для більшості розглянутих наборів призводить до розмежування, яке зображене на рис. 3.22.

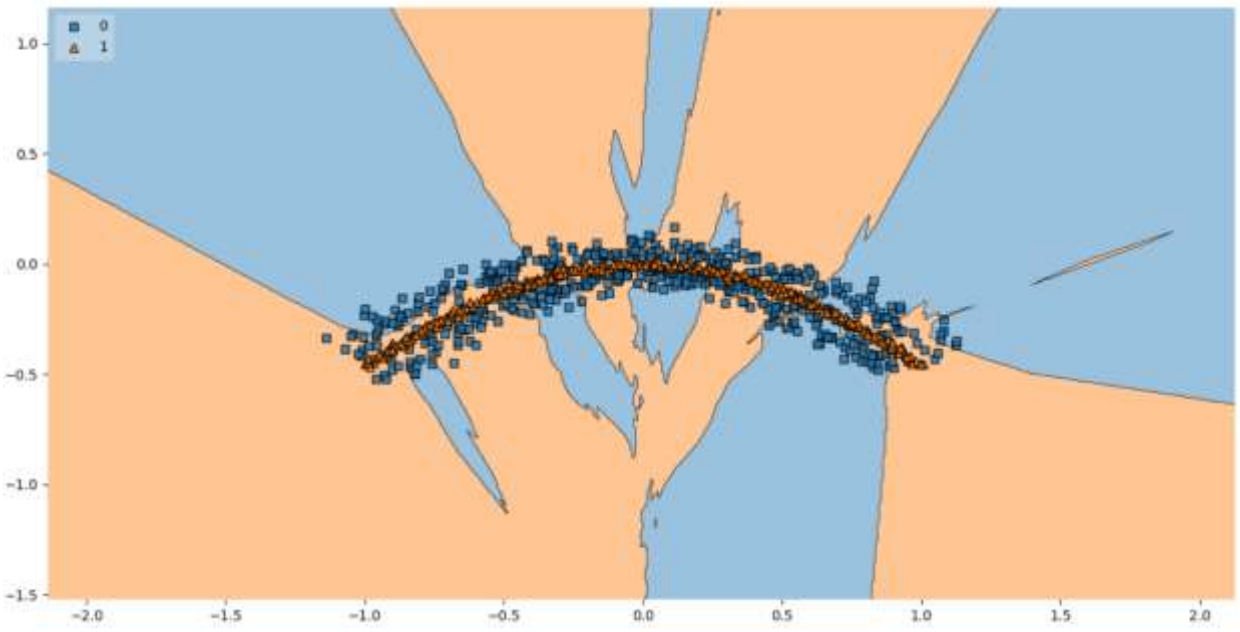


Рис. 3.22 - Графік межі прийняття рішень $k = 10$.

Як результат, отримуємо точність 0.63 з матрицею помилок $\begin{pmatrix} 132 & 273 \\ 22 & 380 \end{pmatrix}$, що свідчить про велику кількість хибно класифікованих елементів 0-го класу (синій), оскільки позначення 1-го класу вийшли за потрібні межі, спробуємо зменшити значення k, що не дозволятиме класу 1 інтенсивно поширюватися. При значенні $k = 4$ точність підвищується до 74% (рис. 3.23), і все ще є недостатньою для навчання нейронної мережі. До того ж значення точності сильно залежить від того, яке саме дані залишаються позначеними перед початком застосування графового методу. Наприклад, один з класів може переважати інший по кількості, тоді він поширюватиметься краще. Або в деяких ключових місцях може бути недостатньо позначень.

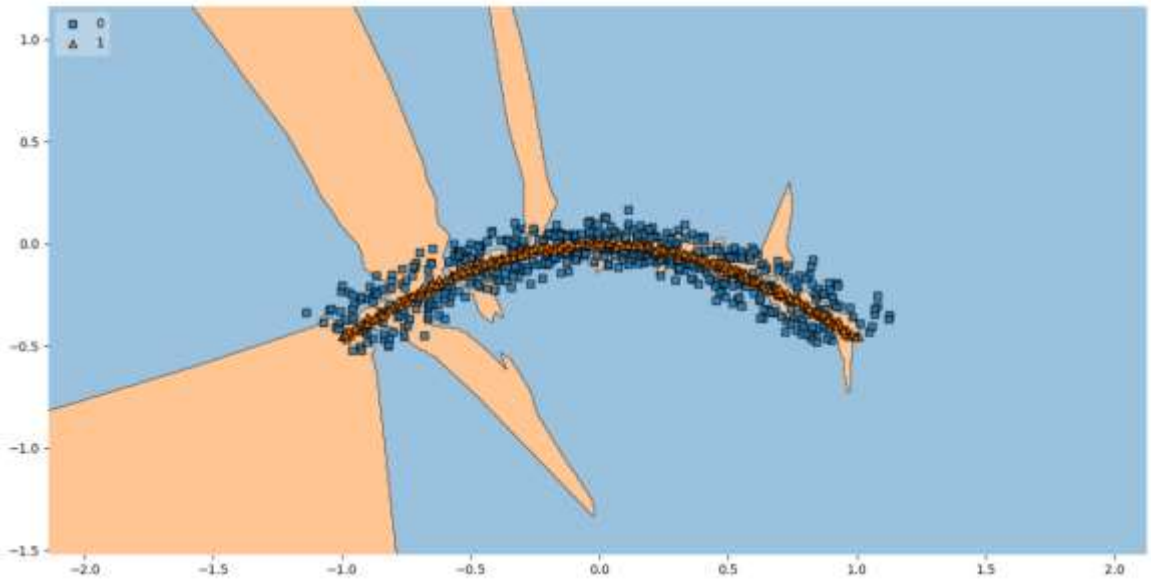


Рис. 3.23 - Графік межі прийняття рішень $k = 4$.

Табл. 3.18 - Оцінка результатів графового методу

Клас	Precision	Recall	F1-score	Support
0	0.78	0.67	0.72	404
1	0.71	0.81	0.76	405
Accuracy		0.74		
Всього непозначених даних		807		
Всього позначених даних		92		

Матриця помилок $\begin{pmatrix} 271 & 133 \\ 75 & 330 \end{pmatrix}$ вказує те, що неправильно класифікованих елементів 208 з 807.

При цьому найкращі показники валідації навчання нейронної мережі, які вдалося досягти показані в табл. 3.19.

Табл. 3.19 - Оцінка результатів класифікації нейронною мережею

Клас	Precision	Recall	F1-score	Support
0	0.95	0.66	0.78	83
1	0.77	0.97	0.86	96
Accuracy		0.83		
Всього валідаційних даних		179		
Тривалість прогнозування		0.086 с		

Матриця помилок: $\begin{pmatrix} 271 & 133 \\ 75 & 330 \end{pmatrix}$.

3.4 Висновки до третього розділу

В даному розділі було проаналізовано роботу графового методу, а також нейронної мережі, на наборах даних різного виду. Ефективність використання графового методу залежить від того, наскільки чітко розмежовані класи, на межах стику точок різних класів виникають складнощі з поширенням маркувань до правильних точок. Це можна корегувати кількістю найближчих сусідів до яких поширюються позначення.

Якщо класи не мають значного розмежування, доцільно використовувати інші методи напівкерованого навчання. Оскільки, погані результати навчання графовим методом впливають на подальші результати навчання нейронної мережі та погіршують їх.

В розділі також досліджено процес навчання нейронної мережі, залежність точності моделі від її параметрів, функцій активації, будови та ін. та побудовано відповідні графіки і таблиці.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

В заданому розділі буде проведено оцінювання основних характеристик для майбутнього програмного продукту, що спеціалізується на дослідженні демографічного стану.

Дана реалізація буде сприяти проведенню усіх необхідних досліджень, що дасть змогу якісно дослідити питання не лише в Україні, проте у всьому світі.

Також в даному дослідженні показано різні варіанти реалізації для забезпечення найбільш коректної та оптимальної стратегії вибору, що має вплив на економічні фактори та сумісність з майбутнім програмним продуктом. Для цього застосовувався апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) передбачає собою технологію, що дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. ФВА проводиться з метою виявлення резервів зниження витрат за рахунок ефективніших варіантів виробництва, кращого співвідношення між споживчою вартістю виробу та витратами на його виготовлення. Для проведення аналізу використовується економічна, технічна та конструкторська інформація.

Алгоритм функціонально-вартісного аналізу включає в себе визначення послідовності етапів розробки продукту, визначення повних витрат (річних) та кількості робочих часів, визначення джерел витрат та кінцевий розрахунок вартості програмного продукту.

4.1 Постановка задачі проектування

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки системи прогнозу стійкості фінансових

показників. Оскільки рішення стосовно проектування та реалізації компонентів, що розробляється, впливають на всю систему, кожна окрема підсистема має її задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу даних по компанії.

Технічні вимоги до програмного продукту є наступні:

- функціонування на персональних комп'ютерах із стандартним набором компонентів;
- зручність та зрозумілість для користувача;
- швидкість обробки даних та доступ до інформації в реальному часі;
- можливість зручного масштабування та обслуговування;
- мінімальні витрати на впровадження програмного продукту.

4.2 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка можливого програмного продукту, яка дозволяє аналізувати різні характеристики, що безпосередньо впливають на стійкість підприємства. Беручи за основу цю функцію, можна виділити наступні:

- F_1 – вибір мови програмування.
- F_2 – реалізація методів.
- F_3 – оцінка ефективності алгоритму.

Кожна з цих функцій має декілька варіантів реалізації:

Функція F_1:

а) Python.

б) Java.

в) C++.

Функція F_2:

а) Застосування бібліотечних функцій.

б) Часткове застосування бібліотечних функцій.

в) Реалізація власних методів.

Функція F_3:

а) Використання бібліотечних функцій.

б) Створення своїх.

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1).

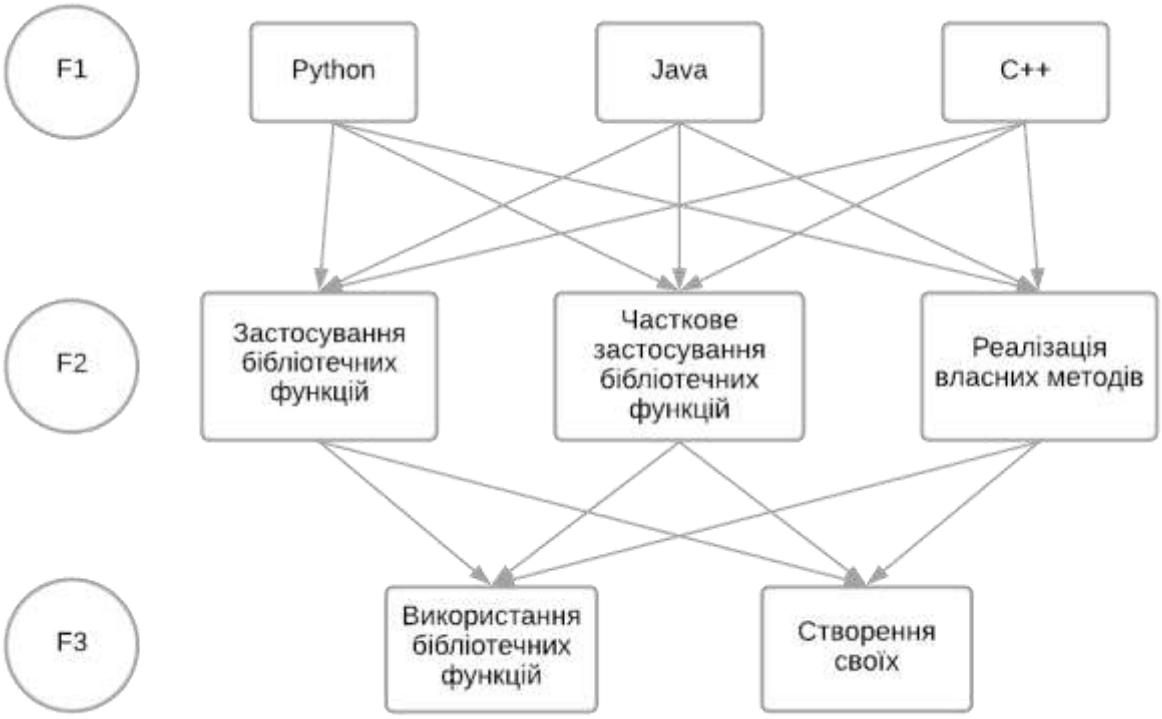


Рис. 4.1 - Морфологічна карта.

Морфологічна карта відображає множину всіх можливих варіантів основних функцій. Позитивно-негативна матриця показана в таблиці 4.1.

Табл. 4.1 - Позитивно-негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F_1	A	Включає різні бібліотеки для машинного навчання та ШІ, операцій над	Синтаксис значно відмінний від інших мов програмування, нижча

		векторами і матрицями, візуалізації; дозволяє ефективно оброблювати велику кількість даних; кросплатформна	швидкість, якщо використовувати вбудовані масиви
	Б	Проста у використанні, кросплатформна, стійка, багатопоточна	Не створена для ШІ та комплексних математичних обчислень; використання пам’яті більшого об’єму, що робить її дорожчою
	В	Швидка, ефективно використовує пам’ять, кросплатформна	Важка у використанні і розумінні; вимагає мануального керування пам’яттю; більше часу на розробку
F_2	*А*	Просте налаштування алгоритмів чи НМ; швидке написання коду; велика різноманітність готових функцій	Обмеженість лише вже готовими функціями
	Б	Швидкість написання коду, простота налаштування, можливість реалізувати функції, яких не вистачає	Потребує час на розробку та інтеграцію з бібліотечними функціями
	В	Послідовна реалізація кожного методу	Великі витрати часу на розробку, не ефективно

			для перевірки різних налаштувань
F_3	*А*	Простота у використанні, інтеграція з іншими бібліотеками	Непрозорість обчислення метрик
	Б	Прозорість обчислення метрик, можливість реалізувати власні	Потребує більше часу на реалізацію і обробку даних

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція F_1:

Перевагу надаємо наявності необхідних бібліотек для машинного навчання та ШІ та можливості ефективної обробки даних великих обсягів. Для реалізації продукту підходить варіант А.

Функція F_2:

Мова програмування допускає обрання усіх трьох варіантів, але надаємо перевагу простоті, швидкості розробки та підтримці різноманітних функцій. Можливо використати варіанти А чи Б.

Функція F_3:

Реалізація першого варіанту є сприйнятливою для обраної мови програмування, оскільки бібліотеки також мають вбудовані метрики для оцінки адекватності та ефективності моделі і алгоритмів. Це варіант А.

Таким чином, будемо розглядати такий варіанти реалізації ПП:

$$F_1\text{а} - F_2\text{а} - F_3\text{а}$$

$$F_1\text{а} - F_2\text{б} - F_3\text{а}$$

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.3 Обґрунтування системи параметрів програмного продукту

На основі даних, розглянутих вище, визначаються основні параметри вибору, які будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

- *X1* – швидкодія мови програмування;
- *X2* – об’єм пам’яті для обчислень та збереження даних;
- *X3* – час навчання даних;
- *X4* – потенційний об’єм програмного коду.

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію програмного продукту, як показано у таблиці 4.2.

Табл. 4.2 - Основні параметри програмного продукту

Назва параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія мови програмування	*X1*	оп/мкс	10	50	100
Об’єм пам’яті	*X2*	Мб	60	50	30
Час попередньої обробки даних	*X3*	мс	80	50	20
Потенційний об’єм програмного коду	*X4*	кількість рядків коду	2100	1500	600

За даними таблиці 4.3 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.5.

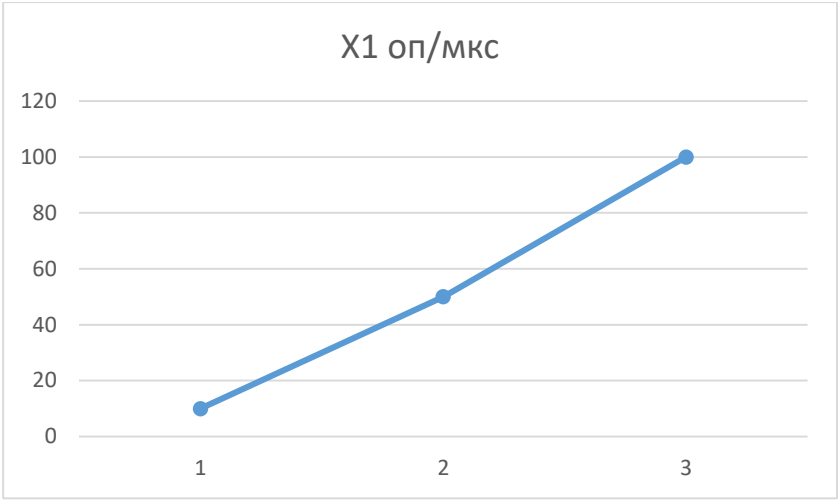


Рис. 4.2 - *X1*, швидкодія мови програмування.

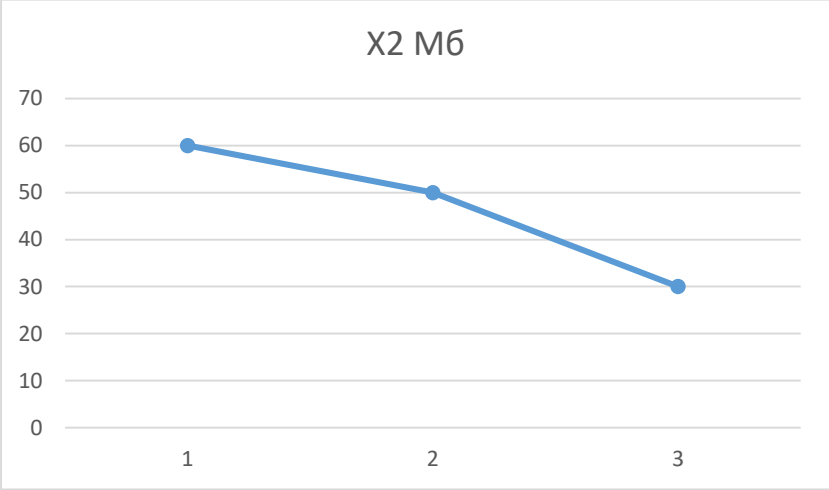


Рис. 4.3 - *X2*, об’єм пам’яті.

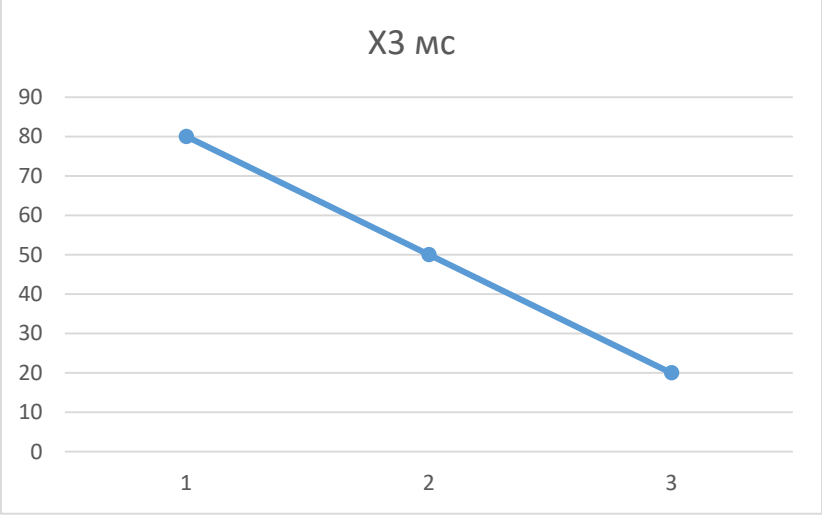


Рис. 4.4 - *X3*, час попередньої обробки даних.

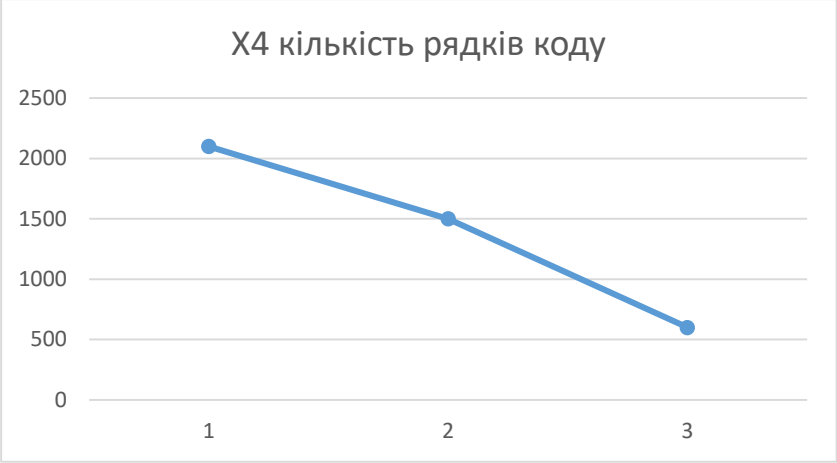


Рис. 4.5 - *X4*, потенційний об’єм програмного коду.

4.4 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробки програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей класифікації.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

- визначення рівня значимості параметра шляхом присвоєння різних рангів;
- перевірку придатності експертних оцінок для подальшого використання;
- визначення оцінки попарного пріоритету параметрів;
- обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Табл. 4.3 - Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхи-лення Δ_i	$\Delta_i{}^2$
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмува-ння	оп/мс	4	3	4	3	3	3	4	11	-6.5	42.25
X2	Об’єм пам’яті	Мб	2	2	2	2	1	2	1	23	5.5	30.25
X3	Час попередньої обробки даних	мс	3	4	3	4	4	4	3	10	-7.5	56.25
X4	Потенційний об’єм програмного коду	Кількість рядків коду	1	1	1	1	2	1	2	26	8.5	72.25
	Разом		10	10	10	10	10	10	10	70	0	201

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij}, \qquad R_{ij} = \frac{Nn(n+1)}{2} = 70, \tag{4.1}$$

де N – число експертів, n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17{,}5 \tag{4.2}$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T. \tag{4.3}$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г) загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 201. \quad (4.4)$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 201}{7^2(4^3 - 4)} = 0{,}82 > W_k = 0{,}67. \quad (4.5)$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4.

Табл. 4.4 - Попарне порівняння параметрів.

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	>	>	>	>	>	>	>	>	1,5
X1 і X3	>	<	>	<	<	<	>	<	0,5
X1 і X4	>	>	>	>	>	>	>	>	1,5
X2 і X3	<	<	<	<	<	<	<	<	0,5
X2 і X4	>	>	>	>	<	>	<	>	1,5
X3 і X4	>	>	>	>	>	>	>	>	1,5

Числове значення, що визначає ступінь переваги i–го параметра над j–тим, a_{ij} визначається по формулі:

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}. \quad (4.6)$$

З отриманих числових оцінок переваги складемо матрицю $A = \| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{\text{ві}}$ за наступними формулами:

$$K_{\text{ві}} = \frac{b_i}{\sum_{i=1}^{n} b_i} \quad (4.7)$$

$$b_i = \sum_{i=1}^{N} a_{ij} \tag{4.8}$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{ві}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \tag{4.9}$$

$$b_i' = \sum_{i=1}^{N} a_{ij} b_j \tag{4.10}$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Табл. 4.5 - Розрахунок вагомості параметрів.

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	*X2*	*X3*	*X4*	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$
X1	1	1,5	0,5	1,5	4,5	0,28	16,25	0,27	59,125	0,27
X2	0,5	1	0,5	1,5	3,5	0,22	12,25	0,21	44,875	0,21
X3	1,5	1,5	1	1,5	5,5	0,34	21,25	0,36	77,875	0,36
X4	0,5	0,5	0,5	1	2,5	0,16	9,25	0,16	34,125	0,16
Всього:					16	1	59	1	216	1

4.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2* (Об’єм пам’яті), *X3* (час попередньої обробки даних) та *X4* (потенційний об’єм програмного коду) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X1* (швидкість роботи мови програмування) обрано не найгіршим.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j}\,, \qquad (4.11)$$

де n – кількість параметрів;

$K_{вi}$ – коефіцієнт вагомості i–го параметра;

B_i – оцінка i–го параметра в балах.

Табл. 4.6 - Розрахунок показників рівня якості варіантів реалізації основних функцій ПП.

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F_1	*А*	*Х1*	50	27	0,27	7,29
F_2	*А*	*Х2*	55	29	0,21	6,09
	Б	*Х3*	70	15	0,36	5,4
F_3	*А*	*Х4*	2200	25	0,16	4

За даними з таблиці 4.6 за формулою:

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}], \qquad (4.12)$$

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 7.29 + 6.09 + 4 = 17{,}38;$$

$$K_{K2} = 7.29 + 5{,}4 + 4 = 16{,}69.$$

Як видно з розрахунків, кращим є 1-й варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Загальна трудомісткість обчислюється як:

$$T_О = T_Р \cdot К_П \cdot К_{СК} \cdot К_М \cdot К_{СТ} \cdot К_{СТ.М}, \quad (4.13)$$

де $T_Р$ – трудомісткість розробки ПП;

$К_П$ – поправочний коефіцієнт;

$К_{СК}$ – коефіцієнт на складність вхідної інформації;

$К_М$ – коефіцієнт рівня мови програмування;

$К_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$К_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення.

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $T_Р$ = 30 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $К_П$ = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної

та вихідної інформації для всіх семи завдань рівний 1: $К_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$= 0.8. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$Т_1 = 30 \cdot 1.7 \cdot 0.8 = 40,8 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $Т_Р$ = 25 людино-днів, $К_П$ = 0.8, $К_{СК}$ = 1, $К_{СТ}$= 0.8:

$$Т_2 = 25 \cdot 0.8 \cdot 0.8 = 16 \text{ людино-днів.}$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$Т_I = (40,8 + 16 + 5.6 + 16) \cdot 8 = 627,2 \text{ людино-годин.}$$
$$Т_{II} = (40,8 + 16 + 7.85 + 16) \cdot 8 = 645,2 \text{ людино-годин.}$$

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 29800 грн., один аналітик в області даних з окладом 35100. Визначимо середню зарплату за годину за формулою:

$$С_Ч = \frac{М}{T_m \cdot t} \text{ грн.,} \qquad (4.14)$$

де М – місячний оклад працівників;

T_m – кількість робочих днів тиждень;

t – кількість робочих годин в день.

$$С_{ч} = \frac{29800 + 29800 + 35100}{3 \cdot 21 \cdot 8} \approx 187{,}90 \text{ грн.} \quad (4.15)$$

Тоді, розрахуємо заробітну плату за формулою:

$$С_{ЗП} = С_{ч} \cdot Т_{i} \cdot К_{Д}, \quad (4.16)$$

де $С_{ч}$ – величина погодинної оплати праці програміста;

$Т_{i}$ – трудомісткість відповідного завдання;

$К_{Д}$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП} = 187{,}9 \cdot 627{,}2 \cdot 1.2 = 141421{,}06$ грн.

II. $С_{ЗП} = 187{,}9 \cdot 645{,}2 \cdot 1.2 = 145479{,}70$ грн.

Відрахування на єдиний соціальний внесок становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 141421{,}06 \cdot 0.22 = 31112{,}63$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 145479{,}70 \cdot 0.22 = 32005{,}53$ грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_{М}$)

Так як одна ЕОМ обслуговує одного програміста з окладом 29800 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$С_{Г} = 12 \cdot М \cdot К_{З} = 12 \cdot 29800 \cdot 0{,}2 = 71520 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$С_{ЗП} = С_{Г} \cdot (1 + К_{З}) = 71520 \cdot (1 + 0.2) = 85824 \text{ грн.}$$

Відрахування на соціальний внесок:

$$С_{ВІД} = С_{ЗП} \cdot 0.22 = 85824 \cdot 0{,}22 = 18881{,}28 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 25600 грн.

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.1 \cdot 0.25 \cdot 25600 = 7040 \text{ грн.},$$

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$К_А$– річна норма амортизації;

$Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.1 \cdot 25600 \cdot 0.06 = 1689,6 \text{ грн.},$$

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 105 – 12 – 15) \cdot 8 \cdot 0.65 =$$

$$= 1211,6 \text{ години},$$

де $Д_К$ – календарна кількість днів у році;

$Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування;

t –кількість робочих годин в день;

$К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1211,6 \cdot 0,4 \cdot 0,35 \cdot 4,27 = 724,29 \text{ грн.},$$

де $N_С$ – середньо-споживча потужність приладу;

$К_З$– коефіцієнтом зайнятості приладу;

$Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$С_Н = Ц_{ПР} \cdot 0.67 = 25600 \cdot 0,67 = 17152 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н, \quad (4.17)$$

$$С_{ЕКС} = 85824 + 18881,28 + 7040 + 1689,6 + 724,29 + 17152 =$$

$$= 131311,17 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 131311,17 / 1211,6 = 108,38 \text{ грн/год.}$$

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$С_М = С_{М-Г} \cdot Т, \quad (4.18)$$

I. $С_М = 108,38 \cdot 627,2 = 67975,94$ грн.

II. $С_М = 108,38 \cdot 645,2 = 69926,78$ грн.

Накладні витрати складають 67% від заробітної плати:

$$С_Н = С_{ЗП} \cdot 0,67, \quad (4.19)$$

I. $С_Н = 141421,06 \cdot 0,67 = 94752,11$ грн.

II. $С_Н = 145479,70 \cdot 0,67 = 97471,40$ грн.

Отже, вартість розробки ПП за варіантами становить:

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_{М} + С_{Н}, \quad (4.20)$$

I. $С_{ПП}$ = 141421,06 + 31112,63 + 67975,94 + 94752,11 = 335261,74 грн.

II. $С_{ПП}$ = 145479,70 + 32005,53 + 69926,78 + 97471,40 = 344883,41 грн.

4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕР j} = К_{Кj} / С_{Фj}, \quad (4.21)$$

$$К_{ТЕР1} = 17,38 / 335261,74 = 5,184 \cdot 10^{-5},$$

$$К_{ТЕР2} = 16,69 / 344883,41 = 4,8393 \cdot 10^{-5}.$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 5,184 \cdot 10^{-5}$.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР1} = = 5,184 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

- Вибір мови програмування – Python;

- Використання бібліотечних функцій для розробки програмного продукту;
- Використання бібліотечних метричних функцій для оцінки роботи моделі.

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, швидку реалізацію програми та доступний функціонал для роботи.

4.8 Висновки до четвертого розділу

В даній частині було проведено повний функціонально-вартісний аналіз програмного продукту. Також було знайдено оцінку основних функцій програмного продукту.

В результаті виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, було визначено та проведено оцінку основних функцій програмного продукту, а також знайдено параметри, які його характеризують.

На основі аналізу вибрано варіант реалізації програмного продукту.

ВИСНОВКИ

В ході виконання дипломної роботи було розглянуто різні методи напівкерованого навчання, зокрема графовий метод та імплементовано напівкероване навчання нейронної мережі на основі графів, що дозволило класифікувати точки на площині, які не використовувалися у навчанні графовим методом. Тобто комбінування графового методу на керованого навчання глибокої нейронної мережі надало алгоритму індуктивності, не зважаючи на те, що за замовчуванням графовий метод є трансдуктивним. Це може бути корисно, якщо об’єкти, які досліджуються можна представити у вигляді графу та якщо є потреба класифікувати дані не лише з навчальної вибірки.

У Розділі 1 наведено аргументи щодо актуальності використання штучного інтелекту в Україні, розглянуто типи машинного навчання та методи напівкерованого навчання: EM-алгоритм, самонавчання, спільне навчання, трансдуктивні опорні векторні машини; різні види будови нейронної мережі та їх застосування.

Розділ 2 присвячений постановці завдання напівкерованого навчання, етапам використання графового методу та різні підходи їх реалізації; метрикам оцінки точності моделі. Наведено постановку задачі дипломної роботи.

У Розділі 3 наведено інформацію про досліджуваний датасет та його використання у роботі, вибір програмного забезпечення, дослідження впливу природи наборів точок на ефективність роботи графового методу та нейронної мережі навченої на виході графового методу. Представлені усі необхідні графіки та оцінки для аналізу ефективності алгоритму з використанням бібліотеки TensorFlow, зокрема.

Було виявлено, що нечіткі межі класів негативно впливають на виконання графового методу на основі kNN алгоритму.

У Розділі 4 проведено повний функціонально-вартісний аналіз програмного продукту, знайдено оцінку основних функцій програмного продукту та параметри, які його характеризують. Обрано найкращий варіант реалізації.

Отже, актуальність роботи полягає в тому, що напівкероване навчання є необхідним у час великих даних, коли непозначені дані широко доступні, але їх маркування вимагає витрат часу, ресурсів і т. п. Крім того, як вже зазначалося, велику кількість систем та об’єктів можна описати у вигляді графу, тому графові методи напівкерованого навчання набувають популярності.

Розроблена модель показала гарні результати на наборах даних різного виду, а також показала доцільність комбінування напівкерованого графового методу та керованого навчання мереж для набуття моделлю індуктивності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. UC Health. About the daVinci Surgical System. – URL: https://www.uchealth.com/services/robotic-surgery/patient-information/davinci-surgical-system/

2. Підкамінний І.М., Романчук К.В. Перспективи використання штучного інтелекту у сфері медицини.

3. Краковецький О. Огляд національних стратегій розвитку штучного інтелекту. – 2018.

4. Про затвердження Концепції Єдиної судової інформаційно-телекомунікаційної системи. НАКАЗ 13.04.2018 № 168.

5. Про рішення Ради національної безпеки і оборони України від 14 травня 2021 року "Про Стратегію кібербезпеки України". Указ президента України №447/2021.

6. Abdel-Nasser Sharkawy. Principle of Neural Network and Its Main Types: Review. 2020. 19 p.

7. Types of Neural Networks and Definition of Neural Network. – URL: https://www.mygreatlearning.com/blog/types-of-neural-networks/

8. Недашківська Н. І. Згорткові нейронні мережі. Лекції. – 2021. – 36 с.

9. IBM. Convolutional Neural Networks. – URL: https://www.ibm.com/topics/convolutional-neural-networks

10. Martin T. Hagan, Howard B. Demuth, Mark Hudson Beale, Orlando De Jesús. Neural Network Design. 2nd Edtion. – 1012 p.

11. Zhu, X. Semi-supervised learning with graphs. Ph.D. thesis, Carnegie Mellon University. – 2005. – 174 p.

12. Zhu, X. Semi-supervised learning literature survey. Technical Report. 1530, University of Wisconsin-Madison. – 2008. – 60 p.

13. Haugh M. The EM Algorithm. – 2015. – 7 p.

14. Gandhi R. Support Vector Machine — Introduction to Machine Learning Algorithms. – 2018. – URL: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

15. Song Z., Yang X., Xu Z. King I. Graph-based Semi-supervised Learning: A Comprehensive Review. – 2021. – 24 p.

16. Jebara, T., Wang, J., Chang, S. F. Graph construction and b-matching for semi-supervised learning. In Proceedings of the 26th annual international conference on machine learning. – 2009. – pp. 441 – 448.

17. Liu, W., Wang, J., & Chang, S. F. Robust and scalable graph- based semisupervised learning. Proceedings of the IEEE, 100(9), 2624– 2638. – 2012.

18. Narkhede S. Understanding Confusion Matrix. – 2018. – URL: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

19. How to choose cross-entropy loss function in Keras? – 2021. – URL: https://androidkt.com/choose-cross-entropy-loss-function-in-keras/

20. Khandelwal R. Semi-Supervised Learning using Label Propagation. – 2021. – URL: https://medium.com/geekculture/semi-supervised-learning-using-label-propagation-2f2d69b8f3ae

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

Модуль label_propagation.py

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.semi_supervised import LabelPropagation
from mlxtend.plotting import plot_decision_regions


def read_data_csv(dataset_name):
    data =
pd.read_csv(f"../resources/{dataset_name}/{dataset_name}_ssl_p10_unsupervised
.csv", header=0,
                        names=['x1', 'x2', 'y']).to_numpy()
    unseen_data =
pd.read_csv(f"../resources/{dataset_name}/{dataset_name}_ssl_p10_supervised.c
sv", header=0,
                              names=['x1', 'x2', 'y']).to_numpy()
    return data, unseen_data


def delete_labels(data, probability=0.9):
    np.random.shuffle(data)
    rd = np.random.RandomState(1)
    random_unlabeled_points = rd.rand(len(data[:, 2])) < probability
    labels = np.copy(data[:, 2])
    x = data[:, :2]
    draw(x, labels)
    labels[random_unlabeled_points] = -1
    tot_unlabeled = labels[labels == -1]
    draw(x, labels)
    print(f"Total records: {len(x)} \tUnlabeled records:
{len(tot_unlabeled)}")
    return labels


def run(x, labels, true_labels, kernel="knn", neighbors=10, gamma=30):
    print(f"Neighbors: {neighbors} \t Gamma: {gamma}")
    model = LabelPropagation(kernel=kernel, n_neighbors=neighbors,
gamma=gamma, max_iter=2000, n_jobs=2)
    model.fit(x, labels)
    predicted_labels = model.predict(x[labels == -1])
    true_predicted_labels = true_labels[labels == -1]
    plot_decision_regions(x, true_labels.astype(np.int_), clf=model,
legend=2)
    plt.show()
    cm = confusion_matrix(true_predicted_labels, predicted_labels,
labels=model.classes_)
    print(f"Label propagation model: {len(labels[labels != -1])} labeled &
{len(labels[labels == -1])} "
          f"unlabeled points ({len(x)} total)")
    print(classification_report(true_predicted_labels, predicted_labels))
    print("Confusion matrix")
    print(cm)
    labels[labels == -1] = predicted_labels
    draw(x, labels)
```

```
    return labels


def draw(x, labels, x_unseen=None, labels_unseen=None):
    plt.scatter(x[labels == 1.0][:, 0], x[labels == 1.0][:, 1], c='#fa5',
label='1.0')
    plt.scatter(x[labels == 0.0][:, 0], x[labels == 0.0][:, 1], c='b',
label='0.0')
    plt.scatter(x[labels == -1.0][:, 0], x[labels == -1.0][:, 1], c='#aaa',
label='unlabeled')
    if x_unseen is not None and labels_unseen is not None:
        plt.scatter(x_unseen[labels_unseen == 1.0][:, 0],
x_unseen[labels_unseen == 1.0][:, 1], c='r', label='1.0')
        plt.scatter(x_unseen[labels_unseen == 0.0][:, 0],
x_unseen[labels_unseen == 0.0][:, 1], c='green', label='0.0')
    plt.xlabel('x1')
    plt.ylabel('x2')
    plt.legend()
    plt.show()
```

Модуль nn.py

```
import time

import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
from sklearn.metrics import classification_report, confusion_matrix

n_epochs = 100
batch_size = 100
optimizers = {"sgd": tf.keras.optimizers.SGD, "adam":
tf.keras.optimizers.Adam}
act_functions = {"relu": tf.nn.relu, "elu": tf.nn.elu, "leaky_relu":
tf.nn.leaky_relu}


def create_dataset(x, labels, true_labels):
    x_train = x / 255
    y_train = labels
    y_train = y_train.astype(np.int32)
    n = int(x_train.shape[0] * 0.2)
    x_test, x_train = x_train[:n], x_train[n:]
    y_test, y_train = y_train[:n], y_train[n:]
    y_valid = true_labels[:n]
    return x_train, y_train, x_test, y_test, y_valid


def loss_at_rate(history):
    lrs = 1e-8 * (10 ** (np.arange(500) / 20))
    plt.figure(figsize=(10, 6))
    plt.grid(True)
    plt.semilogx(lrs, history.history["loss"])
    plt.tick_params('both', length=10, width=1, which='both')
    plt.axis([1e-8, 1, -1, 1])
    plt.show()


def create_model(n_inputs, n_hidden1=128, n_hidden2=64,
                 n_outputs=2, activation=tf.nn.relu, dropout=False):
```

```
    inputs = tf.keras.layers.Input(shape=n_inputs)
    w_init = tf.keras.initializers.random_normal(mean=0., stddev=1.)
    hidden1 = tf.keras.layers.Dense(n_hidden2, activation=activation,
kernel_initializer=w_init)(inputs)
    if dropout:
        hidden1 = tf.keras.layers.Dropout(rate=0.5)(hidden1)
    hidden2 = tf.keras.layers.Dense(n_hidden1, activation=activation,
kernel_initializer=w_init)(hidden1)
    hidden3 = tf.keras.layers.Dense(n_hidden2, activation=activation,
kernel_initializer=w_init)(hidden2)
    outputs = tf.keras.layers.Dense(n_outputs, activation="softmax",
kernel initializer=w init)(hidden3)
    model = tf.keras.Model(inputs=inputs, outputs=outputs)
    return model


def training(x_train, y_train, x_test, y_test, n_inputs, lr, opt, activation,
dropout=False):
    model = create_model(n_inputs, activation=activation, dropout=dropout)
    model.compile(optimizer=optimizers[opt](learning_rate=lr),

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                  metrics=['accuracy'])
    log_dir = "./logs/fit/" + opt + "_" + activation + str(lr) + '_' +
str(dropout)
    tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,

update_freq='epoch')
    lr_schedule = tf.keras.callbacks.LearningRateScheduler(lambda epoch: 1e-8
* 10**(epoch / 20))
    stop_callback = tf.keras.callbacks.EarlyStopping(monitor="loss",
patience=6)
    history = model.fit(x_train, y_train, epochs=n_epochs,
batch_size=batch_size,
                        validation_data=(x_test, y_test),
callbacks=[tensorboard_callback, stop_callback])
    #loss_at_rate(history)
    model.save('./models/final_model.ckpt')
    return model


def predict(model, x_test, y_valid):
    start = time.time()
    predictions = np.argmax(model.predict(x_test), axis=1).astype(np.float32)
    end = time.time()
    print("Prediction lasted:", end - start)
    print(classification_report(y_valid, predictions))
    print("Confusion matrix")
    cm = confusion_matrix(y_valid, predictions, labels=np.array([0.0, 1.0]))
    print(cm)
```

Модуль main.py

```
import label_propagation as lp
import nn


def main():
    size = 2
```

```
    learning_rate = 2e-4
    opt = "adam"
    activation = "relu"
    dropout = False
    data, unseen_data = lp.read_data_csv("c_circles")
    x = data[:, :2]
    true_labels = data[:, 2]
    labels_with_unlabeled = lp.delete_labels(data)
    output_labels = lp.run(x, labels_with_unlabeled, data[:, 2])
    x_train, y_train, x_test, y_test, y_valid = nn.create_dataset(x,
output_labels, true_labels)
    model = nn.training(x train, y train, x test, y test, size,
learning_rate, opt, activation, dropout)
    nn.predict(model, x_test, y_valid)
    nn.predict(model, unseen_data[:, :2]/255, unseen_data[:, 2])


main()
```