МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)
19 травня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 - Комп'ютерних наук,
освітньо-наукової програми «Інформатика»
на тему: «Інформаційна технологія оцінювання відповідності навчального контенту випускової кафедри сучасним вимогам»
здобувача групи ІН.м-11н Надточій Юлії Олександрівни

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Юлія НАДТОЧІЙ
_____
(підпис)

Керівник,
в.о. завідувача кафедри,
кандидат технічних наук, доцент Анатолій ДОВБИШ _____
(підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр

зі спеціальності 122 - Комп'ютерних наук, освітньо-наукової програми «Інформатика»
здобувача групи ІН.м-11н Надточій Юлії Олександрівни

1. Тема роботи: «Інформаційна технологія оцінювання відповідності навчального контенту випускової кафедри сучасним вимогам»
затверджую наказом по СумДУ від «08» травня 2023 р. № 0475-VI
2. Термін здачі здобувачем кваліфікаційної роботи до 19 травня 2023 року
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Огляд методів інтелектуального аналізу та класифікації даних; 2) Постановка й. формування завдань дослідження; 3) Розробка інформаційного та програмного забезпечення системи оцінки якості навчального контенту; 4) Аналіз результатів.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Огляд методів інтелектуального аналізу та класифікації даних*		
2	*Аналіз проблеми дослідження та постановка задачі*		
3	*Розробка інформаційного та програмного забезпечення системи оцінки якості навчального контенту*		
4	*Аналіз отриманих результатів*		
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*		

Здобувач вищої освіти _____ Керівник _____

(підпис) (підпис)

АНОТАЦІЯ

Записка: 92 стор., 22 рис., 1 додаток, 31 джерел.

Обґрунтування актуальності теми роботи – Тема кваліфікаційної роботи є актуальною, оскільки присвячена розв'язанню важливої практичної задачі оцінці відповідності навчального контенту вимогам ринку праці шляхом розробки відповідних методів, моделей та інформаційної технології.

Об'єкт дослідження — процес комп'ютерного оцінювання навчального контенту випускової кафедри вимогам ринку праці.

Мета роботи — підвищення функціональної ефективності системи підтримки прийняття рішень для адаптації навчального контенту випускової кафедри до вимог ринку праці шляхом машинного навчання.

Методи дослідження — інформаційно-екстремальна інтелектуальна технологія аналізу даних, методи теорії інформації та машинного навчання.

Результати — досліджено питання якості освіти, методів оцінки якості освіти та навчального контенту, проведено аналіз існуючих підходів до класифікації даних. В ході дослідження розроблено програмне забезпечення інформаційно-аналітичної системи оцінки навчального контенту вимогам ринку праці згідно з положеннями інформаційно-екстремальної інтелектуальної технології. Сформовано вхідний математичний опис, розглянуто та реалізовано категорійну модель машинного навчання з оптимізацією параметрів за допомогою інформаційних критеріїв, порівняно лінійну та ієрархічну структуру даних та реалізовано функціонування системи на етапі екзамену

ІЄРАРХІЧНА СТРУКТУРА ДАНИХ, ДЕКУРСИВНЕ ДЕРЕВО, ІНФОРМАЦІЙНО-ЕКСТРЕМАЛЬНА ІНТЕЛЕКТУАЛЬНА ТЕХНОЛОГІЯ, СИСТЕМА ОЦІНКИ НАВЧАЛЬНОГО КОНТЕНТУ, МАШИННЕ НАВЧАННЯ.

ЗМІСТ

ВСТУП

У наш час кількість інформації, технологій та знань у різних галузях невпинно зростає, у зв'язку з цим існує необхідність в постійній оцінці та підвищенні якості освіти. При цьому оцінка якості вищої освіти відіграє особливу роль. В умовах мінливого ринкового середовища роботодавці пред'являють високі вимоги до фахівців, а отже і високі вимоги до вищих навчальних закладів. Якість освіти визначається необхідністю та можливість використання отриманих у процесі навчання знать та технологій при подальшій трудовій діяльності у певній галузі. Постає питання актуалізації навчального контенту у контексті ринку праці, а отже необхідність постійного оновлення навчальних програм при підготовці фахівців у вузах. Дані про якість освіти є основою розробки рекомендацій для поліпшення усіх складових навчання, що в свою чергу необхідно для підготовки висококваліфікованих кадрів.

Питання якості освіти та способів його оцінювання не нове. В світі вже давно користуються такими інструментами для визначення якості освіти як внутрішнє і зовнішнє оцінювання. Перше включає оцінку вишу національною чи міжнародною академічною спільнотою, роботодавцями та іншими зацікавленими суб'єктами. До зовнішньої оцінки відносяться рейтинги, що відображають рівень університету за сукупністю показників. Для оцінки якості освіти також використовуються аудит, акредитація, бенчмаркінг. Всі ці перевірки необхідні керівництву для контролю за станом навчального процесу, оцінки динаміки змін, виявлення відхилень та визначення оптимальної стратегії розвитку. Завдання підвищення якості вищої освіти повинно розглядатися не тільки на державного рівні, а у самих вищих навчальних закладах. З розвитком ринку освітніх послуг та підвищенням конкуренції, навчальним закладам необхідні додаткові конкурентні переваги, для привернення уваги нових абітурієнтів. Вищі навчальні заклади є досить вільними при формуванні освітніх планів, постійному вдосконаленні та актуалізації навчальних програм, з метою

стимулювання інновацій у сфері освіти. Успішним буде лише той навчальній заклад, що навчає речам, які можна використовувати при подальшому працевлаштуванні. Нікому не цікаві застаріла інформація та технології, тому існує необхідність швидкого реагування навчального закладу на розвиток технологій. Таким чином, вищим навчальним закладам необхідно система моніторингу відповідності навчальних програм вимогам ринку праці, для своєчасного оновлення навчального контенту, що в свою чергу підвищить рівень підготовки фахівців та дозволить навчальному закладу залишатися конкурентоспроможним на ринку освітніх послуг.

Для оцінки актуальності отриманих знань доцільно залучити самих випускників та роботодавців. При цьому необхідна система, яка буде здатна забезпечити зворотній зв'язок між вищим навчальним закладом та респондентами. Також постає завдання створення механізму, який буде здійснювати обробку і аналіз даних отриманих у процесі зворотного зв’язку. Для подібного аналізу доцільно використовувати технологію інформаційно-екстремального інтелектуального машинного навчання.

Метою даної роботи є створення інформаційно-аналітичної систем для оцінки відповідності якості навчальної програми вимогам ринку праці на основі алгоритмів інформаційно-екстремальної інтелектуальних технологій. На основі аналізу результатів роботи програми можна корегувати та вдосконалювати начальний план, при мінімальних часових та матеріальних затратах на обробку інформації. Введення в експлуатацію навчальних закладів подібної системи має важливе значення для вдосконалення системи навчання, адже дозволить вчасно реагувати на зміни вимог ринку праці, а отже, в перспективі сприятиме підвищенню рівня якості знань фахівців.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКИ ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Аналіз системи оцінки якості освіти в Україні

Сьогодні якісна освіта є не тільки одним з основних індикаторів якості життя, це також важливий інструмент для забезпечення соціального добробуту та економічного зростання [1]. У високотехнологічному інформаційному суспільстві якісна інформація набуває найбільшої цінності та є стратегічним продуктом держав. При цьому якість освіти є ключовою для забезпечення власної потреби в розвитку, а також потреби суспільства і держави в соціально активних громадянах і висококваліфікованих фахівцях [2]. Тому більшість країн світу оголошує проблему освіти національним пріоритетом, усвідомлюючи важливість володіння якісною інформацією.

В Україні знаходиться одна з найбільших мереж у університетів у світі, так на кожний мільйон населення припадає 6,7 університетів. При цьому серед населення, вік якого відповідає типовому віку навчання, 82,7 % задіяні у процесі отримання вищої освіти [3].

Але на основі аналізу світових рейтингів можна спостерігати неоднозначну картину. Згідно з деякими рейтингами Україна має достатньо високий рівень системи вищої освіти, адже потрапляє в топ 50 країн згідно показника частки населення, яке має вищу освіту або тільки отримує її. Але у провідні рейтинги університетів світу вдалося потрапити лише декільком українським вузам, які при цьому зайняли не дуже високі позиції[3].

В українській системі вищої професійної освіти наявні проблеми, які в сукупності з зовнішніми та внутрішніми викликами провокують ризики та мають негативний вплив розвиток. Такими проблемами є невідповідність навичок випускників вищих навчальних закладів вимогам роботодавців та невідповідність програми навчання поточним та перспективним вимогам ринку праці, а також відсутність налагодженого зв’язку вузів з наукою та бізнесом.

Вибірково проведені опитування показують, що значний відсоток населення і роботодавців залишаються незадоволеними сучасної якістю вищої освіти. Причиною є використання навчальними закладами застарілого навчального контенту та підходів до викладання, а також те, що у вузах не використовуються системи оцінки якості освіти.

На сьогодні в Україні не існує комплексної системи, яка б забезпечувала якість вищої освіти. Такі системи, як ліцензування та акредитація, державне інспектування вищих навчальних закладів ззовні та ректорський контроль на даний момент виступають для неї тимчасовими замінниками. Але ні самостійно ні навіть сукупно вони не здатні повною мірою забезпечити системний вплив на всі існуючі процеси по організації навчальної діяльності, та не здатні призвести до постійного покращення якості освіти. В Україні не існує національно визнаних внутрішніх та зовнішніх стандартів чи незалежно працюючих агентств, які б могли гарантувати якість вищої освіти. В університетах на даний момент використовуються різні моделі для проведення моніторингу, аудиту, та управління якістю, але вони не здатні сформують цілісну систему. Такі документи, як система стандартів вищої освіти, Ліцензійні умови надання освітніх послуг у сфері вищої освіти, Вимоги до акредитації напряму підготовки фахівців відповідних освітньо-кваліфікаційних рівнів лише формально можуть виконують функції зовнішніх стандартів, що забезпечують якість освіти. Але навіть сукупність усіх цих систем не здатна створити цілісної і адекватної системи , яка б забезпечувала якість освіти у відповідності до європейських рекомендацій[3].

Згідно з результатами дослідження Міжнародного фонду досліджень освітньої політики відносно впровадження систем якості освіти в університетах , лише 47% опитаних вузів мають спеціально створену структуру, яка виконує функцію управління якістю. Серед опитаних вузів 78% не зробили сертифікацію

систем управління якістю на відповідність вимогам міжнародного стандарту ISO 9001:2008.

Однією з головних проблем вищої освіти в Україні, є відсутність для випускників можливості повність реалізувати власній потенціал на вітчизняному ринку праці. З даною проблемою безпосередньо зіштовхуються здобувачі вищої освіти, але вона також є суттєвою для суспільства в цілому , бо унеможливлює розвиток конкурентоспроможних спеціалістів[3].

Отримуючи професійну освіту, яка є частиною загально-української системи освіти, будь-яка людина може здобути знання, уміння та навичкам у відповідній сфері діяльності. Надання вузом якісної підготовки випускнику, є обов'язковою умовою для його подальшої конкурентоспроможності на ринку праці. Системна нестача інвестицій залучених в освітній процес та інфраструктуру, призводить до того що деякі навчальні заклади використовую зношене на 60% обладнання. В той самий час співпраця між вузами та бізнесом є досить слабкою , а зміст та програми викладання не відповідають сучасним вимогам ринку праці.

Існує невідповідність між очікуваннями роботодавців та випускників та реальним станом якості освіти. Система управління та фінансування не здатна повною мірою забезпечити найкращих викладачів та навчальні заклади гідною винагородою. Люди, що йдуть навчатися в університет не достатньо зрілі, не усвідомлюють повної відповідальності, не володіють достатнім рівнем знать та навичок, не достатньо вмотивовані.

Якість освіти поступово стає багатофункціональним об'єктом управління. Це значить, що її забезпечення вимагає створення нових спеціалізованих систем управління якістю, яких раніше не існувало [4]. Так в Україні бракує системи, яка може надавати дані про оцінки якості освіти, тобто відповідність навчальних програм вимогам ринку праці від роботодавців та випускників. А також системи

моніторингу, що відображатиме зайнятість випускників вищих навчальних закладів у галузі, яку вони вивчали.

Проблемами оцінювання якості вищої освіти займалися та зробили значний внесок такі науковці, як: Архипова С. П. [5], Лукіна Т. О. [6], Ляшенко О. І. [7–9], Сафонова В. Є. [10], Субетто О. І. [12–15], Фоменко Н. А. [16]. Деякі автори [6; 10; 11] виділяють класифікаційні ознаки та підходи до оцінювання якості освіти. Отже, окреслене питання є досить актуальним і певною мірою досліджується, але існує ще багато недостатньо вивчених аспектів, які потребують подальшого розгляду і вироблення пропозицій для прийняття дієвих управлінських рішень.

1.2 Існуючі методи інтелектуального аналізу для класифікації даних

Інтелектуальний аналіз даних – це процес вилучення корисної інформації та зразків даних з величезних інформаційних масивів, який виконується за допомогою сучасних статистичних та математичних методів. Інтелектуальний аналіз даних включає в себе такі методи, як класифікація, дерева рішень, регресія, правила асоціації, штучний інтелект, нейронні мережі, кластеризація, генетичний алгоритм та інших[17].

Одним з найважливіший завдань, яке вирішується методом інтелектуального аналізу даних є класифікація. Дане завдання полягає у визначенні того набору класів, до якого належать об’єкти. А вирішується воно за допомогою аналітичних моделей (класифікаторів). Причинами всезагального використання класифікації є порівняльна простота алгоритмів та методів її реалізації, та висока інтерпретованістю результатів, якщо порівнювати з іншими технологіями аналізу даних.[18]

На сьогодні існує досить багато різних видів класифікаторів, деякі засновані на статистичних методах (логістична регресія, дискримінантний аналіз), інші на методах машинного навчання (метод найближчого сусіда, метод

k-найближчих сусідів, класифікатор Байеса, метод індукції дерев рішень, нейронні мережі, машини опорних векторів та ін.) . Наявність такої значної кількості методів пов’язана з особливостями завдань які необхідно розв’язати. При розв’язуванні задачі аналізу вибір класифікатора є одним з важливих факторів отримання вірного вішення.

Перевагою статистичних класифікаторів, є наявність математичної обґрунтованості, але вони є досить складними у використанні та вимагають знання ймовірнісного розподілу вихідних даних та оцінки його параметрів, а також мають фіксовану структуру моделі. До то ж, статистичні методи можуть лише оцінити ймовірність приналежності об'єкта до класу, але не пояснюють причини. З іншого боку класифікатори в основі яких лежить машинне навчання , не вимагають оцінки параметрів розподілу вихідних даних, а міра подібності в них формалізується за допомогою функції відстані. Такі класифікатори є метричними, їх використання та реалізація є більш простою, ніж для параметричних, а результати значно легше зрозуміти та інтерпретувати. Але в той же час метричні класифікатори можуть забезпечити рішення тільки в обмеженій кількості практично значущих випадків, рішення може бути неточним, чи не єдиним .

Задача класифікації нейронних мереж є компромісом між використанням параметричних та метричних методів. Нейронні мережі є непараметричними моделями, які не вимагають припущення про імовірнісний розподіл даних і не користуються міри відстані. Саме тому вони є універсальними класифікаторами, які дозволяють отримати результати, в той час коли параметричні та метричні класифікатори не можуть гарантувати прийнятного рішення[19].

Задача пошуку найближчого сусіда – це фундаментальна проблема комп’ютерних наук, яка широко використовується в комп’ютерному зорі, колективній фільтрації , у рекомендаційних системах і системах пошуку патернів. Формально завдання пошуку найближчого сусіда визначається

наступним чином: за набором S точок у просторі M та точці запиту q ∈ M знайти найближчу точку S до q. Ступінь схожості двох об'єктів обчислюється, використовуючи функцію відстані чи близькості. Вважають, що менше відстань між об'єктами, тим більше вони схожі (ближче) між собою. Коли функція близькості відповідає трьом аксіомам метрики, позитивність, симетричність; нерівність трикутника, то прийнято говорити про пошук у метричному просторі. Якщо хоча б одна з аксіом може порушуватися, то використовують термін " пошук у неметричному просторі ". Завдання пошуку найближчого сусіда полягає у пошуку найближчого об'єкта до запиту. Узагальненням цього завдання є завдання пошуку найближчих сусідів[20].

Метод k-найближчих сусідів використовується для вирішення завдання класифікації. Він відносить об'єкти до класу, якому належить більшість з його найближчих сусідів у багатовимірному просторі ознак. Це один із найпростіших алгоритмів навчання класифікаційних моделей. Метод широко застосовується в технологіях Data Mining. Незважаючи на відносну алгоритмічну простоту, метод показує хороші результати. Головним його недоліком є висока обчислювальна трудомісткість, яка збільшується квадратично зі зростанням кількості навчальних прикладів[20].

Дерева рішень (DT) – це непараметричний контрольований метод навчання, який використовують для класифікації та регресії. Його суть полягає в тому, щоб створити модель, яка передбачає значення цільової змінної, вивчаючи прості правила прийняття рішень, виведені з характеристик даних. Перевагами дерев є те , що їх можна досить просто зрозуміти та інтерпретувати.. Можлива перевірка моделі за допомогою статистичних тестів. Це дозволяє враховувати надійність моделі. Але з іншого боку рішення, можуть створювати занадто складні дерева, які погано узагальнюють дані, тобто перенавчатися. Щоб уникнути цієї проблеми, необхідні такі механізми, як обрізання, встановлення

мінімальної кількості вибірок, необхідних кінцевого вузла, або встановлення максимальної глибини дерева[21].

Наївний алгоритм Байєса – це алгоритм класифікації, заснований на теоремі Байєса з припущенням про незалежність ознак. Тобто даний алгоритм передбачає, що наявність будь-якої ознаки в класі ніяк не може бути пов’язана з наявністю будь-якої іншої ознаки. Навіть якщо ці ознаки взаємозалежні, або залежать від інших ознак, згідно з алгоритмом вони роблять незалежний внесок у ймовірність. Саме тому алгоритм називається "наївним". Моделі створені на основі даного алгоритму є досить простими і вкрай корисні при роботі з великими наборами даних.

Інформаційно-інтелектуальна екстремальна технологія (ІЕІТ) це технологія машинного навчання, що була розроблена провідними фахівцями Сумського державного університету під керівництвом професора Довбиша А.С. На становлення ІЕІТ вплинули ідеї А.А. Харкевича, Клода Шеннона, О.Г. Івахненка, І.В. Кузьміна, В.І. Костюка, О.А. Павлова та інших. За відносно короткий час з моменту появи, ІЕІТ знайшла використання у багатьох сферах людської діяльності: медицині [22], промисловості [23], кібербезпеці [24], тощо. Широке впровадження технології для різних цілей слугує підтвердженням її працездатність.

Базовою працею, яку присвячено ІЕІТ є [25]. Мірами функціональної ефективності систем на основі ІЕІТ зазвичай виступають міри Кульбака або Шеннона [26]. Одним з важливих питанньь при розробці системи з використанням ІЕІТ яке було досліджено у роботі [27] є вибір базового класу.. Побудові ієрархічної структури даних класів розпізнавання присвячено статтю [28]. Оптимізацію словника ознак досліджено у [29].

ІЕІТ також можна застосовувати й у навчальному процесі. Це питання розглядалося ще у [30]. Найпершими спробами використання ІЕІТ для оцінки

актуальності навчального контенту були спроби у роботі[31]. Проте питання досі залишаться відкритим і потребує подальшого дослідження.

1.3 Постановка задачі

Нехай задано алфавіт $\{X_m^o \mid m=\overline{1,M}\}$ з апріорно вірних класів розпізнавання для системи оцінки якості навчального контенту, які характеризують рівні відповідність навчальних дисциплін вимогам ринку праці, за національною шкалою. Для кожного із даних класів побудовано відповідну тривимірну навчальну матрицю $\| y_{m,i}^{(j)} \|$, яка включає значення оцінок отриманих для кожного навчального модуля, кожної дисципліни в результаті опитування. Тоді рядок $\{y_{m,i}^{(j)} \mid i=\overline{1,N}\}$, де N – кількість ознак, є структурованим вектором (реалізацією), а стовпчик матриці є випадковою навчальною вибіркою $\{y_{m,i}^{(j)} \mid j=\overline{1,n}\}$ i-ї ознаки з обсягом n.

Відповідно до принципів інформаційно-екстремальної інтелектуальної технології, необхідно виконати перетворення вхідної навчальної матриці Y в робочу бінарну X. Під час машинного навчання буде змінюватися саме бінарна матриця. За це відповідає вектор параметрів функціонування, який на пряму впливає на ефективність машинного навчання при розпізнавати реалізації довільного класу X_m^o в бінарному просторі Хеммінга:

$$g_m =< x_m, d_m, \delta >, \qquad (1.1)$$

де x_m – усереднений вектор реалізацій класу X_m^o, водночас є центром гіперсферичного вирішального правила; d_m- радіус гіперсферичного вирішального правила, центром якого є x_m; δ - параметр оптимізації системи контрольних допусків на ознаки розпізнавання.

Згідно принципів інформаційно-екстремальної інтелектуальної технології, наступні обмеження накладаються на параметри функціонування:

- оцінки знаходяться в інтервалі [0;100] балів відповідно до національної шкали;
- для радіусу гіперсферичного вирішального правила d_m допустима область значень визначається наступною нерівністю:

$$d_m < d(x_m \oplus x_c)$$

де $d(x_m \oplus x_c)$ - міжцентрова відстань між реалізацією x_m і найближчою до неї x_c сусіднього класу X_c^o;

- область значень параметра оптимізації системи контрольних допусків на ознаки розпізнавання δ визначається наступною нерівністю:

$$\delta < \delta_H / 2,$$

де δ_H - це нормоване поле допусків на ознаки розпізнавання.

На етапі функціонування системи в режимі екзамену, тобто класифікації оцінок респондентів, необхідною умовою є забезпечення перевірки прийняття класифікаційних рішень. Тому під час машинного навчання необхідно оптимізувати відповідні параметри (1.1).

$$\overline{E}^* = \frac{1}{M} \sum_{m=1}^{M} \max_{G_E \cap \{k\}} E_m^{(k)}, \qquad (1.2)$$

де $E_m^{(k)}$ – значення інформаційного критерію, на k-му кроці машинного навчання; G_E – робоча область для обчислення інформаційного критерію; $\{k\}$ – множина кроків навчання.

Для реалізації **поставленої** задачі необхідно виконати наступні кроки:

- реалізувати відповідні математичні моделі функціонування інтелектуальної компоненти в режимах машинного навчання та екзамену;

- реалізувати перевірку функціональної ефективності для системи оцінювання якості освіти за допомогою обраного інформаційного критерію;
- оцінити функціональну ефективність розпізнавання оцінок респондентів на етапі екзамену;

Отже, задача інтелектуально-інформаційної системи оцінювання якості контенту зводиться до пошуку максимально граничного значення інформаційного критерію (1.2), шляхом оптимізації внутрішніх параметрів машинного навчання.

2 ОПИС МЕТОДУ ДОСЛІДЖЕННЯ

2.1 Основні принципи та положення IEIT

Фундаментальна концепція технології інформаційно-екстремального інтелектуального машинного навчання представляє собою перетворення нечіткого розбиття простору ознак на чітку множину класів еквівалентності оптимізуючи просторово-часові параметри функціонування інформаційної системи ітераційним методом. Характерною особливістю методів, що використовують IEIT є те, що під час оптимізації контрольних допусків відбувається перетворення початкового нечіткого розбиття реалізацій в чітке[].

Відмінністю методів IEI-технології є те, що трансформація вхідного нечіткого розподілу реалізацій образів в чітке здійснюється в процесі оптимізації системи контрольних допусків, що приводить до цілеспрямованої зміни значень ознак розпізнавання і дозволяє побудувати безпомилкові за багатовимірною навчальною матрицею вирішальні правила.[]

Тобто IEI-технологія поєднує в собі одночасну нормалізацію образів розпізнавання відносно еталонного вектору та сам процес навчання, під час якого формуються вирішальні правила.

Нехай відомий алфавіт класів розпізнавання $\{X_m^0 \mid m = \overline{1,M}\}$. У випадку приянятя гіпотези про нечітку компактность реалізацій образу є нечітким розбиттям $\tilde{\Re}^{|M|}$, розбиття простору ознак на класи розпізнавання є нечітким розбиттям $\tilde{\Re}^{|M|}$, а при цьому елементи розбиття є нечіткими класами розпізнавання. Для даного випадку виконуються вирази[]:

$$
\begin{aligned}
&1)\ \left(\forall X_m^0 \in \tilde{\Re}^{|M|}\right)\left[X_m^0 \neq \emptyset\right];\\
&2)\ \left(\exists X_k^0 \in \tilde{\Re}^{|M|}\right)\left(\exists X_l^0 \in \tilde{\Re}^{|M|}\right)\left[X_k^0 \neq X_l^0 \rightarrow X_k^0 \cap X_l^0 \neq \emptyset\right];\\
&3)\ \left(\forall X_k^0 \in \tilde{\Re}^{|M|}\right)\left(\forall X_l^0 \in \tilde{\Re}^{|M|}\right)\left[X_k^0 \neq X_l^0 \rightarrow KerX_k^0 \cap KerX_l^0 \neq \emptyset\right];\\
&4)\ \bigcup_{X_m^0 \in \Re} X_m^0 \subseteq \Omega_{\text{Б}}; k \neq l; k,l,m = \overline{1,M}.
\end{aligned}
\tag{2.1}
$$

У бінарному просторі ознак формою оптимального контейнера класу розпізнавання умовно вважається псевдогіперсфера, яка містить усі вершини гіперпаралелепіпеда. Це дозволяє далі розглядати такі параметри оптимізації контейнера в радіальному базисі простору ознак, як еталонний вектор, наприклад, $x_m \in X_m^o$, вершина якого визначає геометричний центр контейнера K_m^o, і радіус псевдосферичного контейнера, який у просторі Хеммінга визначається за формулою[]:

$$d_m = \sum_{i=1}^{N} (x_{m,i} \oplus \lambda_i), \qquad (2.2)$$

де $x_{m,i}$ – *i*-та координата еталонного вектора x_m; λ_i – *i*-та координата деякого вектора λ, вершина якого належить контейнеру $K_m^o \in X_m^o$.

За IEI-технологією відновлення оптимального контейнера в радіальному базисі, наприклад, K_m^o здійснюється шляхом його цілеспрямованої послідовної трансформації в гіперсферичний габарит, радіус d_m якого збільшується на кожному кроці навчання за рекурентною процедурою[]:

$$d_m(k) = [d_m(k-1) + h \mid d_m(k) \in G_m^d], \qquad (2.3)$$

де k – змінна числа збільшень радіуса контейнера K_m^o; h – крок збільшення радіуса; G_m^d – область допустимих значень радіуса d_m.

Нехай класи X_k^o і X_l^o є “найближчими сусідами”, тобто мають серед усіх класів найменшу міжцентрову відстань $d(x_k \oplus x_l)$, де x_k, x_l – еталонні вектори відповідних класів. збільшення радіуса; G_m^d – область допустимих значень радіуса d_m.

Для того, щоб уникнути ситуації коли один із класів поглинає ядро сусіднього класу (тобто такого, який має мінімальну міжцентрову відстань до даного) вирази (2.1) доповнюються наступною умовою [23]:

$$\left(\forall X_k^o \in \tilde{\Re}^{|M|}\right)\left(\forall X_l^o \in \tilde{\Re}^{|M|}\right)[X_k^o \neq X_l^o \to (d_k^* < d(x_k \oplus x_l))\,\&$$

$$\&\,(d_l^* < d(x_k \oplus x_l))], \qquad (2.4)$$

де d_k^*, d_l^* – оптимальні радіуси контейнерів K_k^o і K_l^o відповідно.

Таким чином, за IEIT технологією вирішальні правила будуються у багатоциклічній процедурі пошуку максимального граничного усередненого інформаційного критерію оптимізації машинного навчання [23]:

$$g_\xi^* = \arg\max_{G_\xi}\{\max_{G_{\xi-1}}\{...\{\max_{G_1 \cap G_E}\frac{1}{M}\sum_{m=1}^{M}E_m\}...\}\}, \qquad (2.5)$$

де E_m – інформаційний критерій оптимізації параметрів машинного навчання; G_ξ – допустима область значень ξ-го параметра машинного навчання; G_E – робоча (допустима) область визначення функції інформаційного критерію.

2.2 Оцінка функціональної ефективності

Важливим кроком етапу навчання є вибір та побудова критеріїв оптимізації параметрів функціонування системи з метою максимізації її інформаційної спроможності [10]. Функціональна ефективність системи характеризується ступенем того, що робота системи задовольняє поставленим перед нею критеріям мети. Введення критеріїв мети здійснюється для запобігання невизначеності в описі інформаційної системи, до того ж, мірою такої невизначеності виступає кількість інформації. Тому, враховуючи інформаційний характер критеріїв мети, виникає необхідність у проведенні аналізу критеріїв функціональної ефективності (КФЕ).

Найбільшу перевагу у використанні з метою оцінки функціональної ефективності системи отримали ентропійна міра за Шенноном та інформаційна міра Кульбака [14].

Нормований ентропійний КФЕ за Шенноном навчання системи розпізнаванню реалізації класу X_m^0 відповідає виразу (2.6):

$$\mathrm{E}_m^{(k)} = \frac{I_m^{(k)}}{I_{max}^{(k)}} = \frac{H_m^{(k)} - H_m^{(k)}(\gamma)}{H_m^{(k)}}, \qquad (2.6)$$

де $I_m^{(k)}$ – величина умовної інформації на k-му кроці алгоритму навчання;

$I_{max}^{(k)}$ – максимальна кількість умовної інформації, що може бути отримана на кроці k;

$H_m^{(k)} = -\sum_{l=1}^{M} p(\gamma_{l,k}) \log_2 p(\gamma_{l,k})$ – апріорна ентропія на кроці k;

$H_m^{(k)}(\gamma) = -\sum_{l=1}^{M}\sum_{m=1}^{M} p(\gamma_{l,k}) p\left(\frac{\mu_{m,k}}{\gamma_{l,k}}\right) \log_2 p\left(\frac{\mu_{m,k}}{\gamma_{l,k}}\right)$ – апостеріорна ентропія, визначає залишкову невизначеність у результаті виконання кроку k;

$p(\gamma_{l,k})$ – апріорна ймовірність прийняття гіпотези $\gamma_{l,k}$ на k-му кроці;

$p\left(\frac{\mu_{m,k}}{\gamma_{l,k}}\right)$ – апостеріорна ймовірність прийняття гіпотези $\mu_{m,k}$ на k-му кроці, у разі, якщо було прийнято $\gamma_{l,k}$.

Для поширеної на практиці задачі, коли рішення є двохальтернативним (M=2) і гіпотези є рівноймовірними, здійснюється модифікація ентропіного КФЕ шляхом підстановки апріорної та апостеріорної ентропії у вираз (2.6) та заміни умовної ентропії за формулою Байєса:

$$E_m^{(k)} = 1 + \frac{1}{2}\left(\frac{\alpha_m^{(k)}(d)}{\alpha_m^{(k)}(d) + D_{2,m}^{(k)}(d)} \log_2 \frac{\alpha_m^{(k)}(d)}{\alpha_m^{(k)}(d) + D_{2,m}^{(k)}(d)} + \right.$$

$$+ \frac{\beta_m^{(k)}(d)}{\beta_m^{(k)}(d) + D_{1,m}^{(k)}(d)} \log_2 \frac{\beta_m^{(k)}(d)}{\beta_m^{(k)}(d) + D_{1,m}^{(k)}(d)} + \frac{D_{1,m}^{(k)}(d)}{\beta_m^{(k)}(d) + D_{1,m}^{(k)}(d)}$$

$$\left. \log_2 \frac{D_{1,m}^{(k)}(d)}{\beta_m^{(k)}(d) + D_{1,m}^{(k)}(d)} + \frac{D_{2,m}^{(k)}(d)}{\alpha_m^{(k)}(d) + D_{2,m}^{(k)}(d)} \log_2 \frac{D_{2,m}^{(k)}(d)}{\alpha_m^{(k)}(d) + D_{2,m}^{(k)}(d)} \right) \qquad (2.7)$$

де $\alpha_m^{(k)}(d)$ – помилка 1-го роду прийняття рішення на кроці k;

$\beta_m^{(k)}(d)$ – помилка 2-го роду прийняття рішення на кроці k;

$D_{1,m}^{(k)}(d)$ – 1-ша достовірність прийняття рішення на кроці k;

$D_{2,m}^{(k)}(d)$ – 2-га достовірність прийняття рішення на кроці k;

d – відстань від центру класу розпізнавання до контейнера.

Так як наведені точнісні характеристики визначають величину віддаленості вершин еталонних векторів від центрів гіперповерхонь-контейнерів класів, то критерій (2.7) є нелінійним, взаємно-неоднозначним функціоналом від описаних точнісних характеристик, а його визначення вимагає пошуку робочої області в ході навчання.

Враховуючи, що інформаційний критерій (2.7) залежить від точнісних характеристик, то за репрезентативного обсягу навчальної вибірки слід застосовувати наступні оцінки (2.8):

$$D_{1,m}^{(k)}(d) = \frac{K_{1,m}^{(k)}}{n_{min}}; \qquad \alpha_m^{(k)}(d) = \frac{K_{2,m}^{(k)}}{n_{min}};$$

$$\beta_m^{(k)}(d) = \frac{K_{3,m}^{(k)}}{n_{min}}; \qquad D_{2,m}^{(k)}(d) = \frac{K_{4,m}^{(k)}}{n_{min}}, \qquad (2.8)$$

де $K_{1,m}^{(k)}$ – кількість подій, що визначають належність реалізацій до контейнеру $K_{1,k}^0$, якщо вони дійсно належать класу X_1^0;

$K_{2,m}^{(k)}$ – кількість подій, що визначають неналежність реалізацій до контейнеру $K_{1,m}^0$, якщо вони насправді належать класу X_1^0;

$K_{3,m}^{(k)}$ – кількість подій, що визначають належність реалізацій до контейнеру $K_{1,m}^0$, якщо вони насправді належать класу X_2^0;

$K_{4,m}^{(k)}$ – кількість подій, що визначають неналежність реалізацій до контейнеру $K_{1,m}^0$, якщо вони дійсно належать класу X_2^0;

n_{min} – мінімально можливий репрезентативний обсяг навчальної вибірки.

У результаті підстановки оцінок (2.8) до виразу (2.7) отримано робочу формулу для розрахунку ентропійного інформаційного критерію (2.9):

$$\mathrm{E}_{1,m}^{(k)} = 1 + \frac{1}{2}\left(\frac{K_{1,m}^{(k)}}{K_{1,m}^{(k)} + K_{3,m}^{(k)}} \log_2 \frac{K_{1,m}^{(k)}}{K_{1,m}^{(k)} + K_{3,m}^{(k)}} + \frac{K_{2,m}^{(k)}}{K_{2,m}^{(k)} + K_{4,m}^{(k)}} \log_2 \frac{K_{2,m}^{(k)}}{K_{2,m}^{(k)} + K_{4,m}^{(k)}} + \right.$$

$$\left. + \frac{K_{3,m}^{(k)}}{K_{1,m}^{(k)} + K_{3,m}^{(k)}} \log_2 \frac{K_{3,m}^{(k)}}{K_{1,m}^{(k)} + K_{3,m}^{(k)}} + \frac{K_{4,m}^{(k)}}{K_{2,m}^{(k)} + K_{4,m}^{(k)}} \log_2 \frac{K_{4,m}^{(k)}}{K_{2,m}^{(k)} + K_{4,m}^{(k)}}\right) \quad (2.9)$$

Інформаційна міра, що представлена добутком логарифмічного відношення повної ймовірності правильного прийняття рішення $P_{t,m}^{(k)}$ до повної ймовірності помилкового прийняття рішення $P_{f,m}^{(k)}$ на відповідні відхилення, є варіантом відомої диференціальної інформаційної міри Кульбака [10], та має наступний вигляд:

$$\Lambda = \log_2 \frac{P_{t,m}^{(k)}}{P_{f,m}^{(k)}} = \log_2 \frac{p(\mu_m)p\left(\frac{\gamma_{1,k}}{\mu_m}\right) + p(\mu_c)p\left(\frac{\gamma_{2,k}}{\mu_c}\right)}{p(\mu_m)p\left(\frac{\gamma_{2,k}}{\mu_m}\right) + p(\mu_c)p\left(\frac{\gamma_{1,k}}{\mu_c}\right)}, \quad (2.10)$$

де $p(\mu_m)$ – апріорна ймовірність появи реалізацій класу X_m^0;

$p(\mu_c)$ – апріорна ймовірність появи реалізації класу-сусіда X_c^0;

$\gamma_{1,k}$ – гіпотеза вказує, що реалізації класу X_m^0 до належать контейнеру $K_{m,k}^0$;

$\gamma_{2,k}$ – гіпотеза-альтернатива.

Враховуючи, що за правилом Бернуллі-Лапласа $p(\mu_m) = p(\mu_c) = 0.5$, модифікація інформаційної міри Кульбака набуває вигляду:

$$E_{K\,m}^{(k)} = \log_2 \frac{P_{t,m}^{(k)}}{P_{f,m}^{(k)}} * \left[P_{t,m}^{(k)} - P_{f,m}^{(k)}\right] = \begin{vmatrix} P_{t,m}^{(k)} = 0.5D_{1,m}^{(k)}(d) + 0.5D_{2,m}^{(k)}(d) \\ P_{f,m}^{(k)} = 0.5\alpha_m^{(k)}(d) + 0.5\beta_m^{(k)}(d) \end{vmatrix} =$$

$$= 0.5 \log_2 \left(\frac{D_{1,m}^{(k)}(d) + D_{2,m}^{(k)}(d)}{\alpha_m^{(k)}(d) + \beta_m^{(k)}(d)}\right) * [(D_{1,m}^{(k)}(d) + D_{2,m}^{(k)}(d)) - (\alpha_m^{(k)}(d) +$$

$$+\beta_m^{(k)}(d))] = \log_2 \left(\frac{2 - \left(\alpha_m^{(k)}(d) + \beta_m^{(k)}(d)\right)}{\alpha_m^{(k)}(d) + \beta_m^{(k)}(d)}\right) * [1 - (\alpha_m^{(k)}(d) + \beta_m^{(k)}(d))] \quad (2.11)$$

Нормований варіант інформаційної міри Кульбака представлений наступним виразом (2.12):

$$\mathrm{E}_{K,m}^{(k)} = \frac{\mathrm{E}_{K\,m}^{(k)}}{\mathrm{E}_{K\,max}^{(k)}}, \qquad (2.12)$$

де $\mathrm{E}_{K\,max}^{(k)}$ – значення критерію за умов, що $D_{1,m}^{(k)}(d) = D_{2,m}^{(k)}(d) = 1$ і $\alpha_m^{(k)}(d) == \beta_m^{(k)}(d) = 0$ для виразу (2.11).

Нормування критерію виконується необов'язково для задач з пошуку екстремуму параметрів навчання, які відповідають максимальному значенню критерію функціональної ефективності в межах робочої області. Таку процедуру необхідно застосувати у разі порівнянні результатів досліджень або оцінці схожості потенційної і реальної інформаційних систем.

Для отримання робочої формули критерію Кульбака, оцінки (2.8) підставлено до виразу (2.11):

$$E = \frac{1}{n}\log_2\left\{\frac{2n+10^{-r}-\left[K_2^{(k)}+K_3^{(k)}\right]}{\left[K_2^{(k)}+K_3^{(k)}\right]+10^{-r}}\right\}\left[n-\left(K_2^{(k)}+K_3^{(k)}\right)\right] \qquad (2.17)$$

де r – кількість числових знаків у мантисі критерію $E_m^{(k)}$.

Отже, наведені модифікації відомих статистичних і дистанційних критеріїв оптимізації параметрів функціонування інтелектуальної системи інформаційна критерії можуть бути застосовані у системах, що здатні навчатися з метою побудови безпомилкових вирішальних правил. При цьому використання статистичних критеріїв має перевагу у тому, що в процесі навчання можна отримати чітке розбиття класів із нечіткого.

2.3 Лінійний алгоритм машинного навчання

У загальному випадку базовий алгоритм навчання дозволяє оптимізувати геометричні параметри контейнерів класів розпізнавання, обчислити інформаційний критерій функціональної ефективності навчання системи та знайти глобальний максимум критерію у робочій області визначення його функції.

Для базового алгоритму вхідною інформацією для навчання є масив реалізацій образу $\{y_m^{(j)} \mid m=\overline{1,M};\ j=\overline{1,n}\}$; система полів контрольних допусків $\{\delta_{K,i}\}$ і рівні селекції $\{\rho_m\}$ координат еталонних векторів-реалізацій (за замовчуванням дорівнюють 0,5 для всіх класів розпізнавання).

До етапів реалізації базового алгоритму належать [23]:

1) Формування бінарної навчальної матриці $\| x_{m,i}^{(j)} \|$.

2) Формування масиву еталонних двійкових векторів-реалізацій $\{x_{m,i} \mid m=\overline{1,M},\ i=\overline{1,N}\}$, елементи яких визначаються за правилом

$$x_{m,i}=\begin{cases}1, & if \quad \frac{1}{n}\sum_{j=1}^{n} x_{m,i}^{(j)} > \rho_m;\\ 0, & if \quad else,\end{cases} \qquad (2.18)$$

де ρ_m – рівень селекції координат вектора $x_m \in X_m^o$.

3) Розбиття множини еталонних векторів на пари найближчих «сусідів»: $\Re_m^{|2|}=<x_m\ ,\ x_l>$, де x_l – еталонний вектор сусіднього класу X_l^o, за такою схемою алгоритму:

а) структурується множина еталонних векторів, починаючи з вектора x_1 базового класу X_1^o, який характеризує найбільшу функціональну ефективність ІС;

б) будується матриця кодових відстаней між еталонними векторами розмірності $M \times M$;

в) для кожного рядка матриці кодових відстаней знаходиться мінімальний елемент, який належить стовпчику вектора, найближчого до вектора, що визначає рядок. За наявності декількох однакових мінімальних елементів вибирається з них будь-який, оскільки вони є рівноправними;

г) формується структурована множина елементів попарного розбиття $\{\Re_m^{|2|} \mid m=\overline{1,M}\}$, яка задає план навчання.

4) Оптимізація кодової відстані d_m відбувається за рекурентною процедурою $d_m(k)=[d_m(k-1)+h \mid d_m(k)\in G_m^d]$. При цьому береться $E_m(0)=0$.

5) Процедура закінчується при знаходженні максимуму КФЕ в робочій області його визначення: $E_m^*=\max\limits_{\{d\}} E_m$, де $\{d\}=\{d_1,...,d_k,...,d_{\max}\}\in[0;d(x_m\oplus x_l)-1]$ – множина радіусів концентрованих гіперсфер, центр яких визначається вершиною еталонного вектора $x_m\in X_m^o$. При цьому множина $\{d\}$ є так само множиною кроків навчання інформаційної системи.

Категорійну модель машинного навчання за базовим алгоритмом зображено на рисунку 2.1 [24]. Розглянемо дану категорійну модель навчання інформаційної системи за базовим алгоритмом у вигляді діаграми відображення множин. При обґрунтуванні гіпотези нечіткої компактності має місце нечітке розбиття $\tilde{\Re}^{|M|}\subset\Omega$. Введемо оператор θ нечіткої факторизації простору ознак: θ: $Y\to\tilde{\Re}^{|M|}$ і оператор класифікації ψ: $\tilde{\Re}^{|M|}\to I^{|l|}$, який перевіряє основну статистичну гіпотезу про належність реалізацій $\{x_m^{(j)} \mid j=\overline{1,n}\}$ нечіткому класу X_m^o. Тут l– кількість статистичних гіпотез. Оператор γ: $I^{|l|}\to\Im^{|q|}$ формує множину точнісних характеристик шляхом оцінки статистичних гіпотез $\Im^{|q|}$, де $q=l^2$ – кількість точнісних характеристик. Оператор $\varphi:\Im^{|q|}\to E$ обчислює множину значень інформаційного КФЕ, який є функціоналом точнісних характеристик. Контур оптимізації геометричних параметрів нечіткого розбиття $\tilde{\Re}^{|M|}$ шляхом пошуку максимуму КФЕ навчання розпізнаванню реалізацій класу X_m^o замикається оператором r: $E\to\tilde{\Re}^{|M|}$.

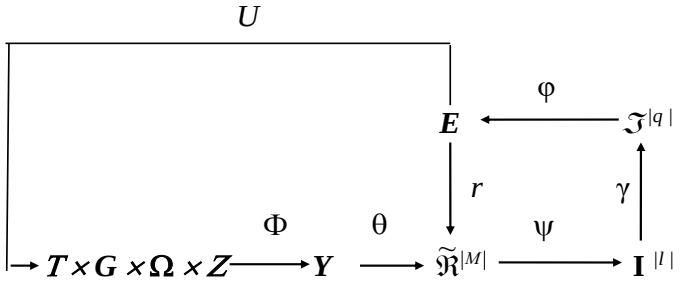

Рисунок 2.1 – Категорійна модель машинного навчання за базовим алгоритмом

На рисунку 2.1 прийнято такі позначення:

$T \times G \times \Omega \times Z$ – джерело інформації (T – множина моментів часу формування векторів-реалізацій класів розпізнавання; G – фактори, що впливають на функціонування системи; Ω – простір діагностичних ознак; Z – простір технічних станів системи, які визначають алфавіт класів розпізнавання);

Y – вхідна навчальна матриця;

$I^{(l)}$ – множина, що перевіряє основну статистичну гіпотезу;

$\mathfrak{I}^{|q|}$ – множина точнісних характеристик;

E – множина значень інформаційного критерію;

D – система контрольних допусків;

$\tilde{\Re}^{|M|}$ – розбиття простору ознак на класи розпізнавання;

θ – оператор нечіткої факторизації простору ознак: $\theta : Y \to \tilde{\Re}^{|M|}$

ψ – оператор класифікації який перевіряє основну статистичну гіпотезу про належність реалізацій $\{ x_m^{(j)} | j=\overline{1,n} \}$ нечіткому класу $\psi : \tilde{\Re}^{|M|} \to \mathbf{I}^{|l|}, \ X_m^o$;

γ – оператор, який шляхом оцінки статистичних гіпотез формує множину точнісних характеристик $\mathfrak{I}^{|q|}$: γ: $I^{|l|} \to \mathfrak{I}^{|q|}$;

φ – оператор, який обчислює множину значень інформаційного КФЕ, який є функціоналом точнісних характеристик: $\varphi : \mathfrak{I}^{|q|} \to E$;

r – оператор, який замикає контур оптимізації геометричних параметрів нечіткого розбиття $\tilde{\Re}^{|M|}$ r: $E \to \tilde{\Re}^{|M|}$;

U – оператор, який регламентує процес навчання: U: $T \times G \times \Omega \times Z$.

Отже, базовий алгоритм навчання реалізує ітераційну процедуру пошуку глобального максимуму інформаційного коефіцієнта функціональної ефективності в робочій області визначення його функції:

$$d_m^* = \arg\max_{\{d\}} E_m^* . \qquad (2.19)$$

Таким чином, основною функцією базового алгоритму навчання є обчислення на кожному кроці навчання інформаційного коефіцієнта функціональної ефективності і пошук його глобального максимуму в робочій області визначення функції критерію для визначення оптимальних геометричних параметрів розбиття простору ознак на класи розпізнавання.

2.4 Інформаційно-екстремальне машинне навчання системи за ієрархічною структурою даних

Саме можливість використання геометричного підходу при побудові класифікаційних правил є основною перевагою інформаційно-екстремальної технології при порівнянні з нейроподібними методами інтелектуального аналізу даних. Насамперед ця особливість допомагає побудувати майже інваріантні класифікаційні правила, які будуть не чутливі до довільного положення об’єкту інтересу в кадрі розпізнавання. Тим не менш залишається не вирішеною проблематика багатовимірності алфавіту класів, які в процесі функціонування системи можуть досягати довільних значень. Відомо, що ступінь перетину класів розпізнавання збільшується при збільшенні кількості класифікаційних ознак. Як один із можливих варіантів подолання цієї проблеми є перехід від лінійного класифікатору, до структурованого, наприклад ієрархічного.

У рамках інформаційно-екстремальної технології структурний класифікатор прийнято будувати як дерево [30], де кожна наступна страта містить в собі атрибути верхніх, у нашому випадку це класи розпізнавання. Тобто

ідея ідентифікації буде полягати у переході від більшої абстракції до меншої. При цьому кожен клас, який ієрархічно знаходиться вище, кожного разу повертається в свою страту більш нижнього ярусу. Оскільки цей принцип відрізняється від класичного рекурсивного дерева, то таку структуру будемо називати декурсивною (рис. 2.4).

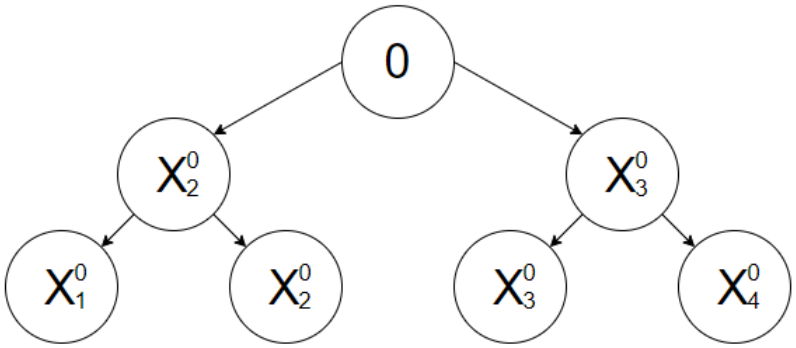

Рисунок 2.2 – Ієрархічна структура даних у вигляді декурсивного дерева

Запропонована структура дозволяє реорганізувати лінійний класифікатор для навіть для непарної кількості класів таким чином щоб попарно виконувати класифікацію. Такий підхід дозволяє підвищити ймовірність побудови безпомилкових, за навчальними даними, класифікаційних рішень через зменшення перетину класів.

Категорійну модель машинного навчання системи діагностування з ієрархічним класифікатором, відповідно до роботи [30], представимо у вигляді орієнтованого графу. На відмінну від лінійного класифікатору (рис.2.1), ієрархічний (рис.2.5) має додатковий контур оптимізації, який замикається через структурну множину *H*.

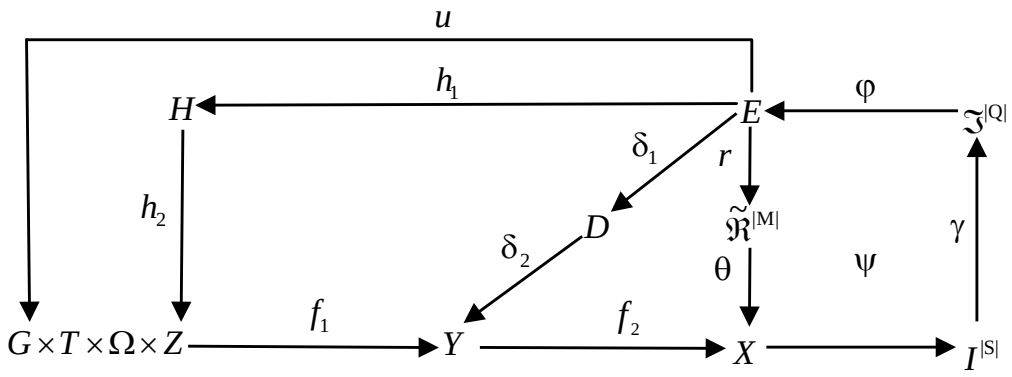

Рисунок 2.3 – Категорійна модель ієрархічного навчання

Відповідно до рисунку 2.3 оператор h_1 обирає необхідний ярус і страту з множини *H,* ці дії задають план машинного навчання. У свою чергу h_2 змінює простір можливих технічних станів об’єкти ідентифікації, що впливає на алфавіт класів розпізнавання.

Виходить, що кінцева модель ієрархічного машинного навчання, у рамках IEI-технології, полягає в побудові багаторівневої структури. Кожен рівень якої складається з лінійного класифікатору, який визначає подальший рух кадрів розпізнавання. Тим не менш, при такому підході з’являється питання розташування алфавіту класів в ієрархічному дереві. Найпростіший спосіб структуризації – це експертна оцінка. Тобто деяка людина, за допомогою своїх суб’єктивних знань може розмістити класи, наприклад, за рівнем їх абстракції. Більш загальні вище, більш конкретні нижче. Проте це можна зробити і автоматично. Експериментально було підтверджено [3], що чим менша дисперсія класу розпізнавання, тим більш однозначний на ньому об’єкт. Цим принципом можна скористатися для встановлення ієрархічних зв’язків серед класів розпізнавання.

3 РОЗРОБКА АЛГОРИТМІЧНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ

3.1 Формування вхідного математичного опису

Оцінювання навчального контенту проводилося за національною шкалою вимірювання, в даному випадку було виділено чотири класи розпізнавання, кожен з яких характеризує певний рівень якості. Кожний з чотирьох класів («Відмінно», «Добре», «Задовільно», «Незадовільно») характеризується центром розсіювання та радіусом контейнера:

Клас 1 («Відмінно») – $X_1^0: C_1 = 92, d_1 = 8$

Клас 2 («Добре) – $X_2^0: C_2 = 81, d_2 = 8$

Клас 3 («Задовільно») – $X_3^0: C_3 = 69, d_3 = 8$

Клас 4 («Незадовільно») – $X_4^0: C_4 = 57, d_4 = 8$

Графічне зображення розташування границь класів у просторі зображено на рисунку 3.1, де Клас X_1^0 позначено червоним кольором, клас X_2^0 – чорним, клас X_3^0 – зеленим, а Клас X_4^0 – блакитним.

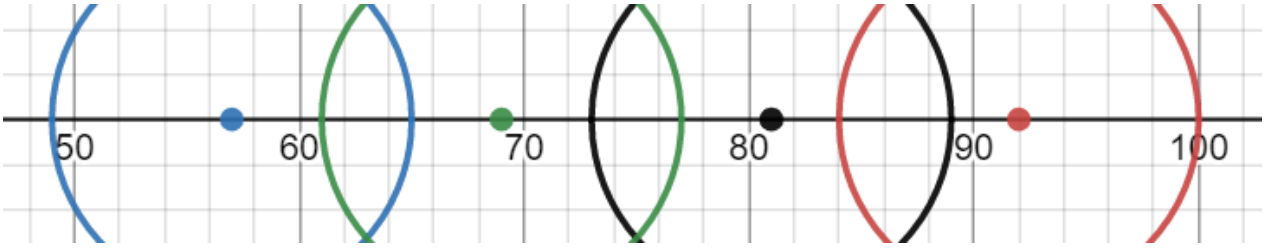

Рисунок 3.1 – Границі класів розпізнавання у просторі

Вхідні матриці для класів розпізнавання були створені імітаційним шляхом. Для кожного класу було згенеровано 40 реалізацій, в кожній реалізації – 147 характеристик і при цьому 30% реалізацій знаходяться на перетинах з іншими класами.

Оскільки було вирішено реалізувати машинне навчання як з лінійною, так і з ієрархічною структурою даних, то опису ієрархічної структури було реалізоване декурсивне бінарне дерево.

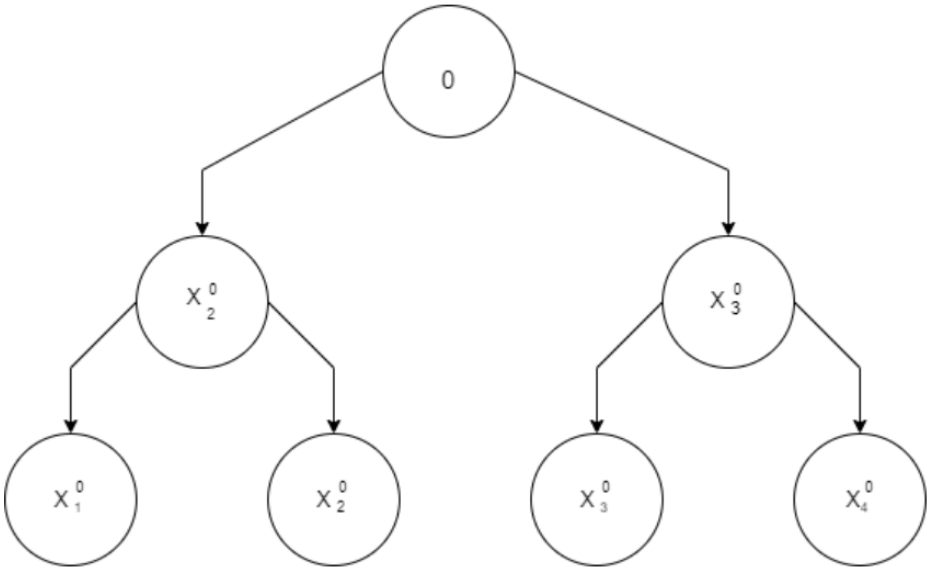

Рисунок 3.2 – Ієрархічна структура даних у вигляді декурсивного бінарного дерева

Як видно з рисунку 3.2 бінарне декурсивне дерево розбиває заданий алфавіт на страти, кожна з яких містить по два найближчих сусідніх класи. Це дозволяє в кожної фінальної страти класів розпізнавання застосовувати лінійний алгоритм інформаційно-екстремального машинного навчання. А отже навчання необхідно провести для трьх фінальних страт: (X_2^0, X_3^0), (X_1^0, X_2^0), (X_3^0, X_4^0).

3.2. Короткий опис програмної реалізації

Система оцінки відповідності якості навчального контенту вимогам ринку праці була реалізовано на мові програмування Java . Нижче наведено опис основних класів та методів, які були використані. Лістінг програми приведено у Додатку.

Клас Main необхідний для завантаження вхідних даних з оцінкам дисциплін з файлів та є точкою запуску лінійного та ієрархічного методів навчання.

Таблиця 3.1Опис основних методів класу Main

Методи	
Назва	**Призначення**
public static void main(String[] args)	Завантаження даних, запуск процесів навчання з лінійною та ієрархічною структурою даних, запуск процесу екзамену з ієрархічною структурою
public static int[][] readFile(int [] size, String filename)	Читання даних про оцінки класів з файлу

Клас MainAlgorithm реалізує механізм навчання лінійної структури за базовим алгоритмом з оптимізацією контрольних допусків класів розпізнавання.

Таблиця 3.2 Опис основних методів класу MainAlgorithm

Методи	
Назва	**Призначення**
public void run ()	Запуск процесів навчання
public void findOptDelta()	Метод для пошуку оптимального delta
learning(int delta)	Метод, що запускає алгоритм навчання в залежності від delta
public void getPermits()	Метод для отримання значень контрольних допусків

public void getAllBinaryMatrix()	Метод для отримання бінарних матриць на основі контрольних допусків
getEtalonVector()	Метод для отримання двійкових усереднених векторів ознак розпізнавання
getPair()	Метод для пошуку сусідніх класів
public void getCodeDistance()	Метод для пошуку кодової відстані до своїх та найближчих реалізацій
findAllCharacteristics()	Метод для обрахування точністних характерстик, таких як помилки першого і другого роду, першої і другої достовірності, значення коефіцієнта функціональної ефективності
public void getOptRadius()	Метод для пошуку оптимального радіуса для контейнерів розпізнавання

Клас HierarchicalTree реалізує механізм навчання ієрархічної структури даних за базовим алгоритмом з оптимізацією контрольних допусків класів розпізнавання і режим екзамену.

Таблиця 3.3 Опис основних методів класу MainAlgorithm

Методи	
Назва	**Призначення**
public void run()	Запуск процесів навчання
public void findAllOptDelta(Tree tree)	Рекурсивний метод для обходу дерева при пошуку оптимального delta
public Tree findOptDelta(int[][][] matrix)	Метод для пошуку оптимального delta для страти
private Tree baseAlgorithm(int[][][] strata, int delta)	Метод, що запускає алгоритм навчання для страти в залежності від delta
private double[][] findAllKulback(float d1[][], float d2[][], float alfa[][], float betta[][]) {	Метод для отримання значень інформаційного інформаційного критерію
private float[][] getPermits(int[][][] strata, int delta)	Метод для отримання значень контрольних допусків для страти
private int[][][] getAllBinaryMatrix	Метод для отримання бінарних матриць на основі параметра поля контрольних допусків
private int[][][] getSK(int[][][] binaryMatrix, int[][] etalonVectors)	Метод для пошуку кодової відстані до своїх та найближчих реалізацій
private int[] getOptRadius(int[][][] codeDistance, int maxR, double kulback[][], float d1[][], float d2 [][])	Метод для пошуку оптимального радіуса для контейнерів розпізнавання

public void make_exam(int[][] matrix)	Метод, що запускає процедуру екзамену
private void exam(Tree tree, int[][] matrix)	Рекурсивний метод, що дозволяє на основі функції належності відности тестові реалізації до одного з класів страти
private int[][] getAllEtalonVector(int[][][] binaryMatrix)	Метод для отримання еталонних векторів
public float[][] getD1(int sum_sk[][][], int maxR, int[][][] codeDistance)	Метод для отримання значень першої достовірності
public float[][] getAlfa(int sum_sk[][][], int maxR, float d1[][])	Метод для отримання значень помилки першого роду
public float[][] getD2(int sum_sk[][][], int maxR, float betta[][]) {	Метод для отримання значень другої достовірності
public float[][] getBetta(int sum_sk[][][], int maxR, int[][][] codeDistance)	Метод для отримання значень помилки другого роду

3.3 Аналіз результатів

Розглянемо результати отримані в результаті машинного навчання для лінійної структури даних.

При пошуку оптимального базового класу для лінійної структури, було виявлено, що при будь-якому еталонному класі, робоча область не існує (рис. 3.3-3.6).

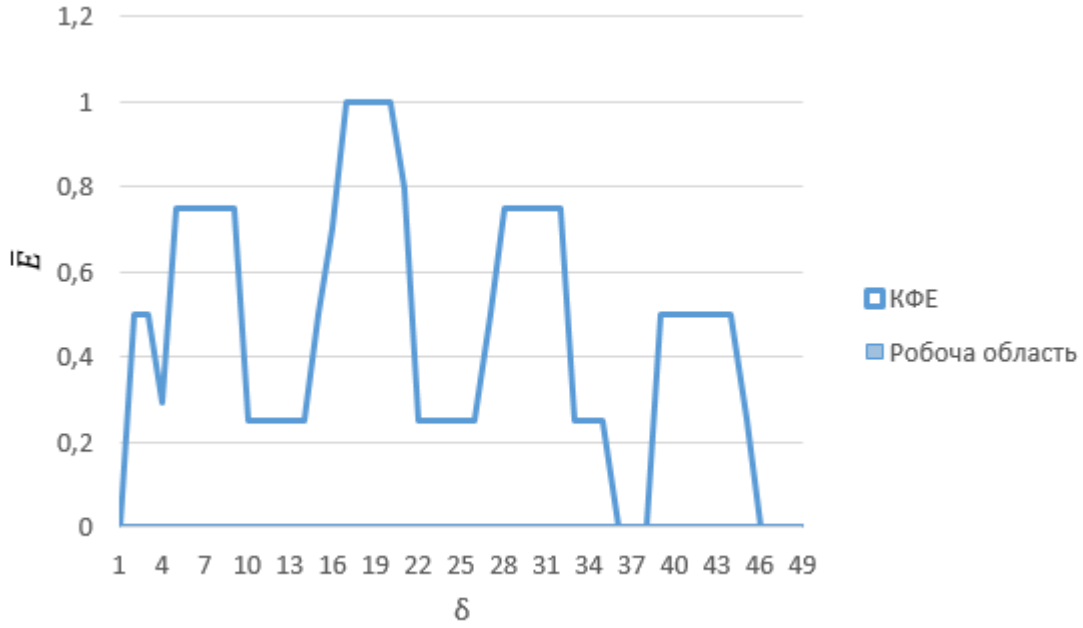

Рисунок 3.3 – Графік залежності інформаційного критерію від параметра поля контрольних допусків для еталонного класу X_1^0

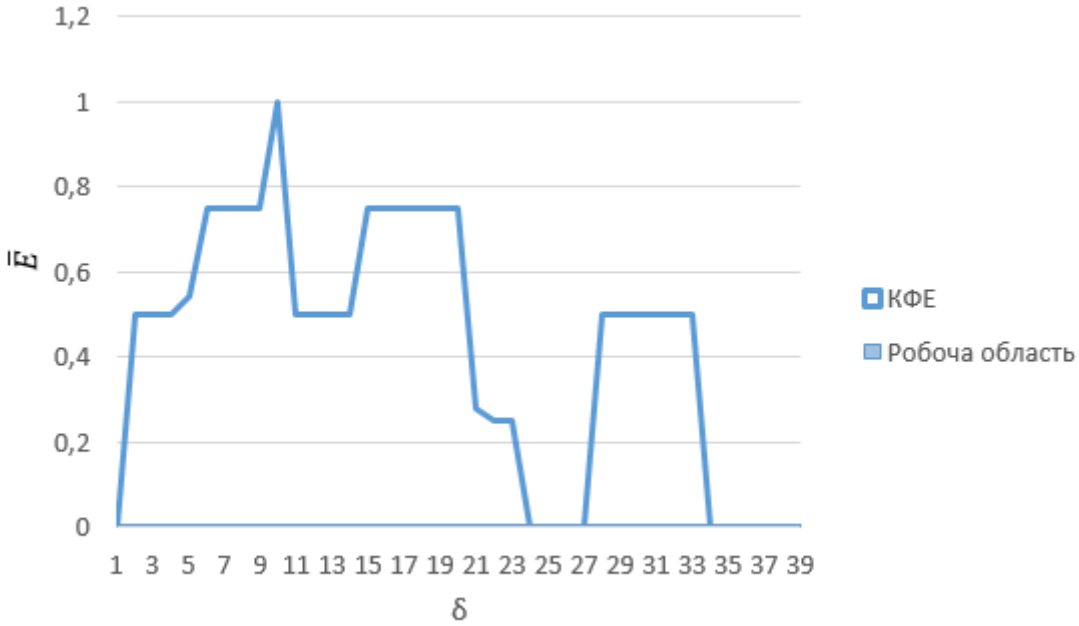

Рисунок 3.4 – Графік залежності інформаційного критерію від параметра поля контрольних допусків для еталонного класу X_2^0

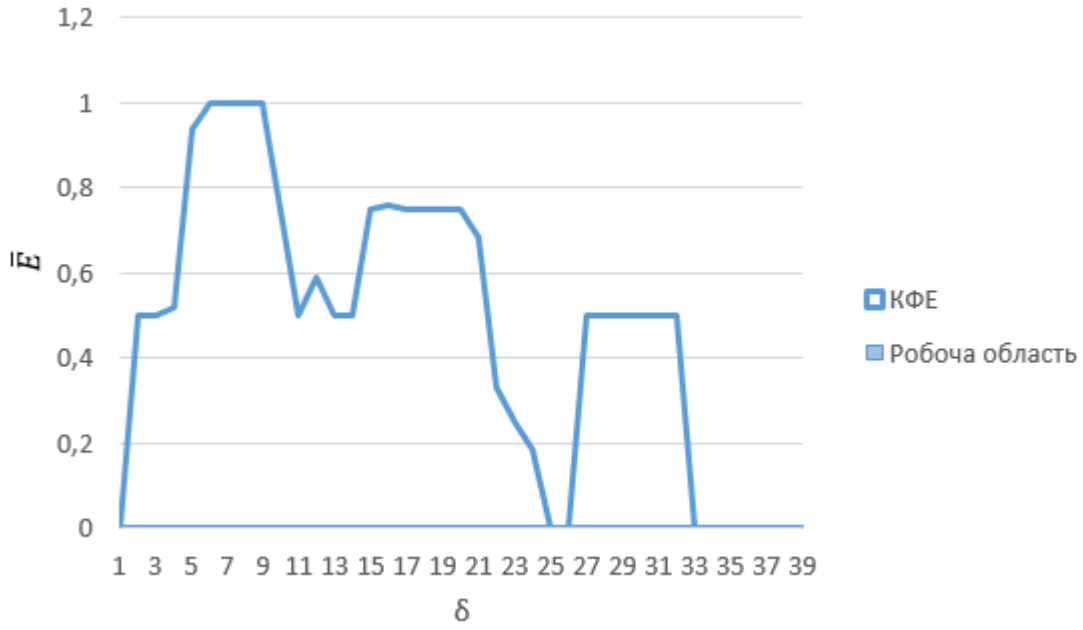

Рисунок 3.5 – Графік залежності інформаційного критерію від параметра поля контрольних допусків для еталонного класу X_3^0

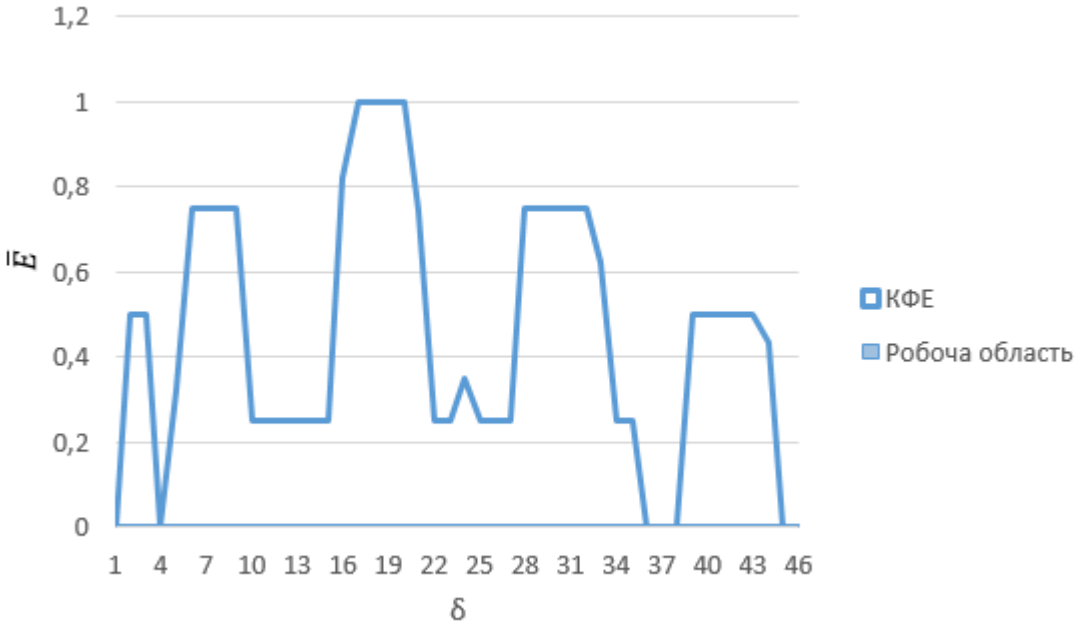

Рисунок 3.6 – Графік залежності інформаційного критерію від параметра поля контрольних допусків для еталонного класу X_4^0

На основі аналізу отриманих даних було встановлено, що робоча область не існує, так як гіперсфреричні контейнери одних класів лежать в середині інших. Для вирішення даної проблеми було вирішено розглянути всі можливі варіанти діапазонів для радіусів контейнерів. При цьому візьмемо за еталонний клас X_3^0. В результаті дослідження вдалося отримати значення інформаційного критерію в робочій області і побудувати вирішальні правила (рис 3.7). Оптимальне значення поля контрольних допусків $\delta = 5$, а діапазони радіусів контейнерів відповідно $d_1 = [0; 1]$, $d_1 = [1; 24], d_1 = [0; 75]$ та $d_4 = [1; 26]$.

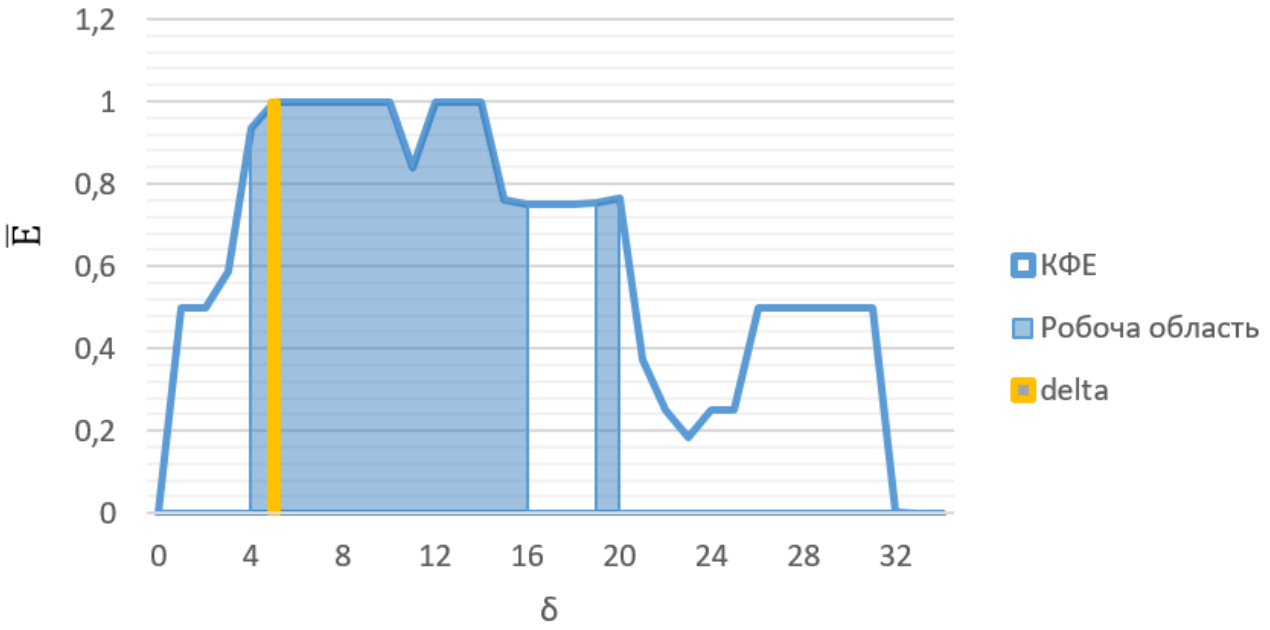

Рисунок 3.7 – Графік залежності інформаційного критерію від параметра поля контрольних допусків для еталонного класу X_3^0

Але, при подібних вирішальних правилах, процес екзамену значно ускладняються, адже виникає питання побудови нової функції належності

Розглянемо результати машинного навчання отримані для ієрархічної структури.

Для страти (X_2^0, X_3^0), значення оптимального параметру поля контрольних допусків $\delta = 5$, а радіуси контейнерів розпізнавання відповідно $d_2 = 117, d_3 = 78$.

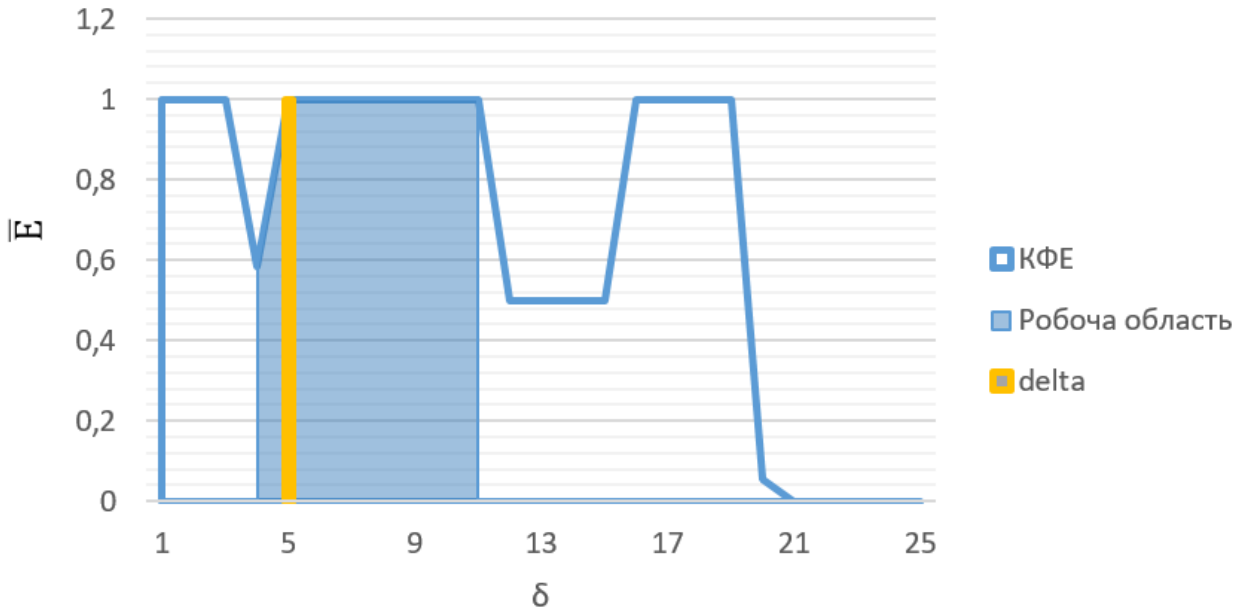

Рисунок 3.8 – Графік залежності інформаційного критерію від параметра поля контрольних допусків для страти (X_2^0, X_3^0)

На рисунках 3.9-3.10 відображено графіки залежності інформаційного критерію від радіусів контейнерів класів розпізнавання при значенні δ = 5 для страти (X_2^0, X_3^0).

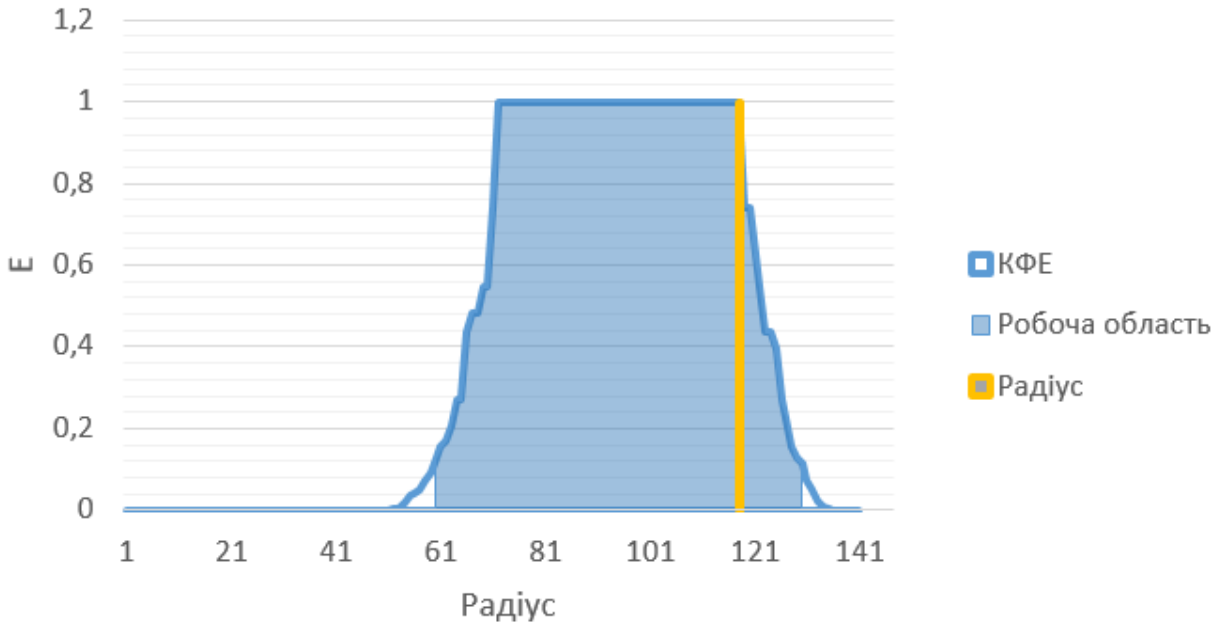

Рисунок 3.9 – Графік залежності інформаційного критерію від радіусу контейнера класу X_2^0 для страти (X_2^0, X_3^0)

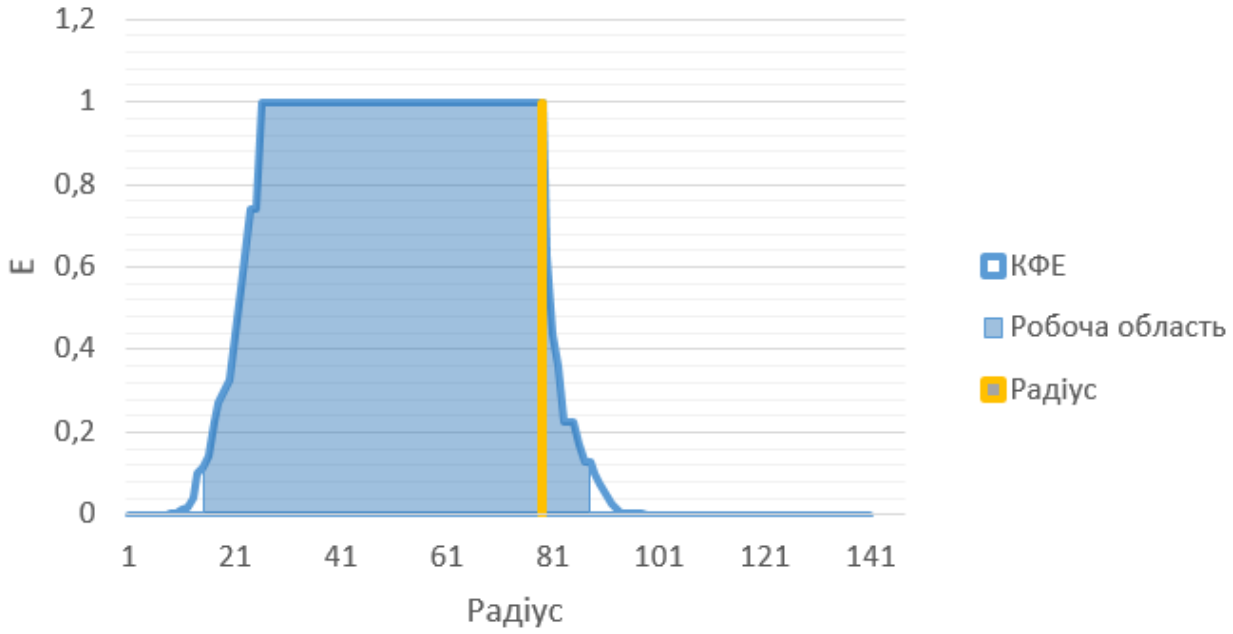

Рисунок 3.10 – Графік залежності інформаційного критерію від радіусу контейнера класу X_3^0 для страти (X_2^0, X_3^0)

Для страти (X_1^0, X_2^0), значення оптимального параметру поля контрольних допусків $\delta = 4$, а радіуси контейнерів розпізнавання відповідно $d_1 = 81, d_2 = 65$.

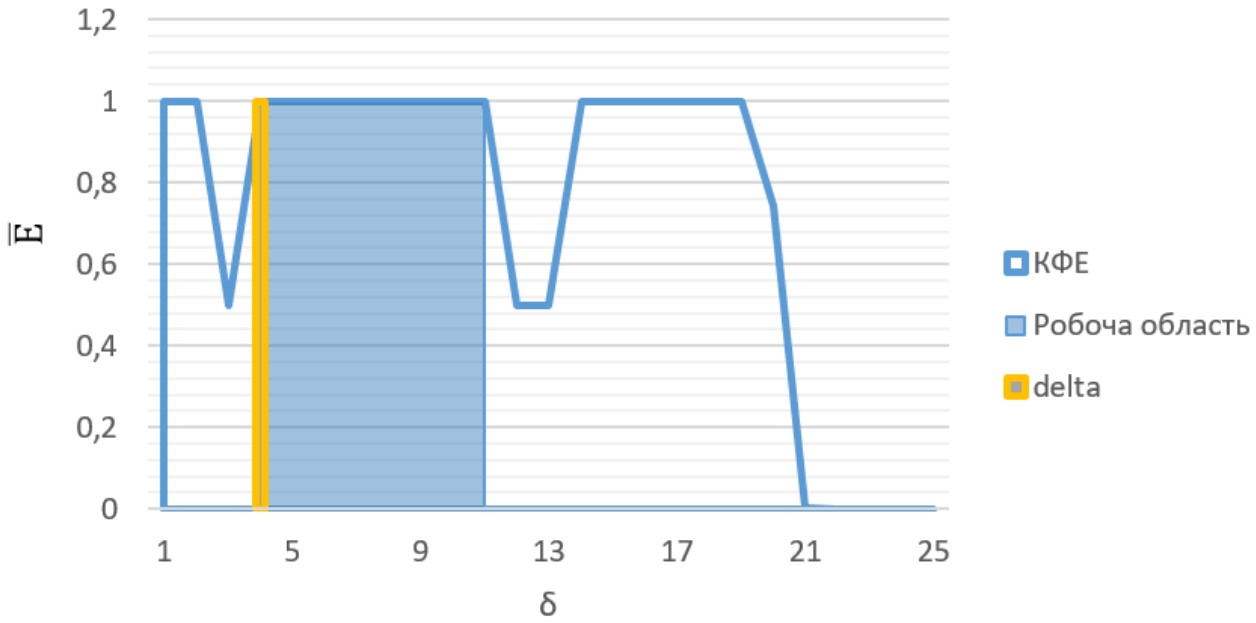

Рисунок 3.11 – Графік залежності інформаційного критерію від параметра поля контрольних допусків для страти (X_1^0, X_2^0)

На рисунках 3.12-3.103відображено графіки залежності інформаційного критерію від радіусів контейнерів класів розпізнавання при значенні δ = 4 для страти (X_1^0, X_2^0).

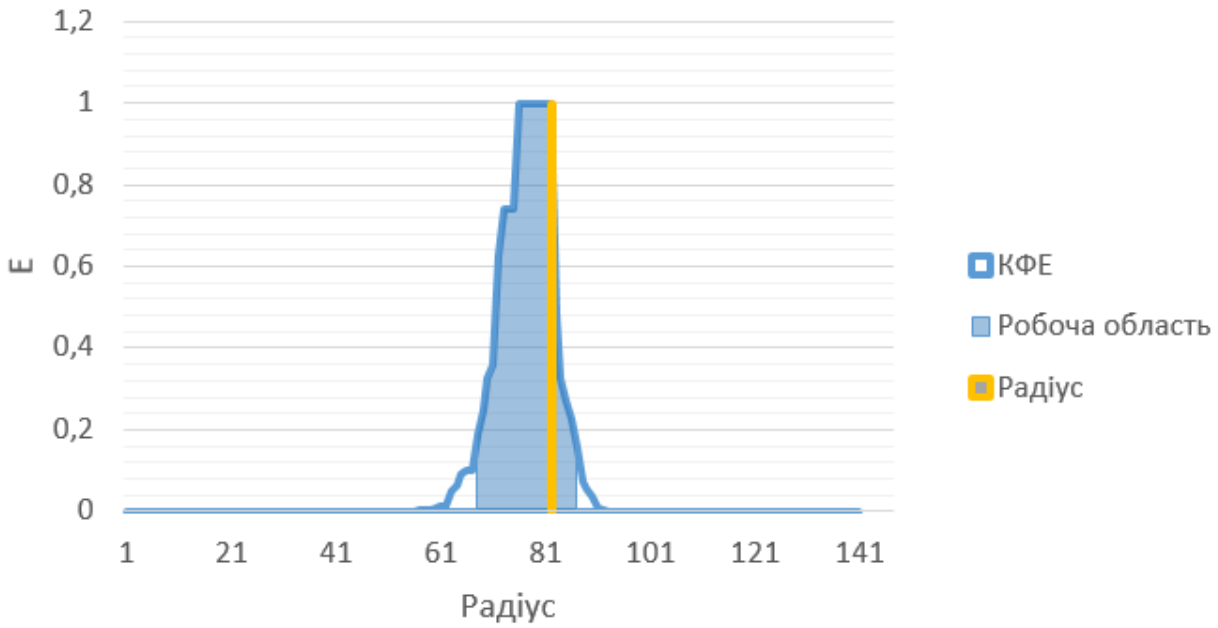

Рисунок 3.12 – Графік залежності інформаційного критерію від радіусу контейнера класу X_1^0 для страти (X_1^0, X_2^0)

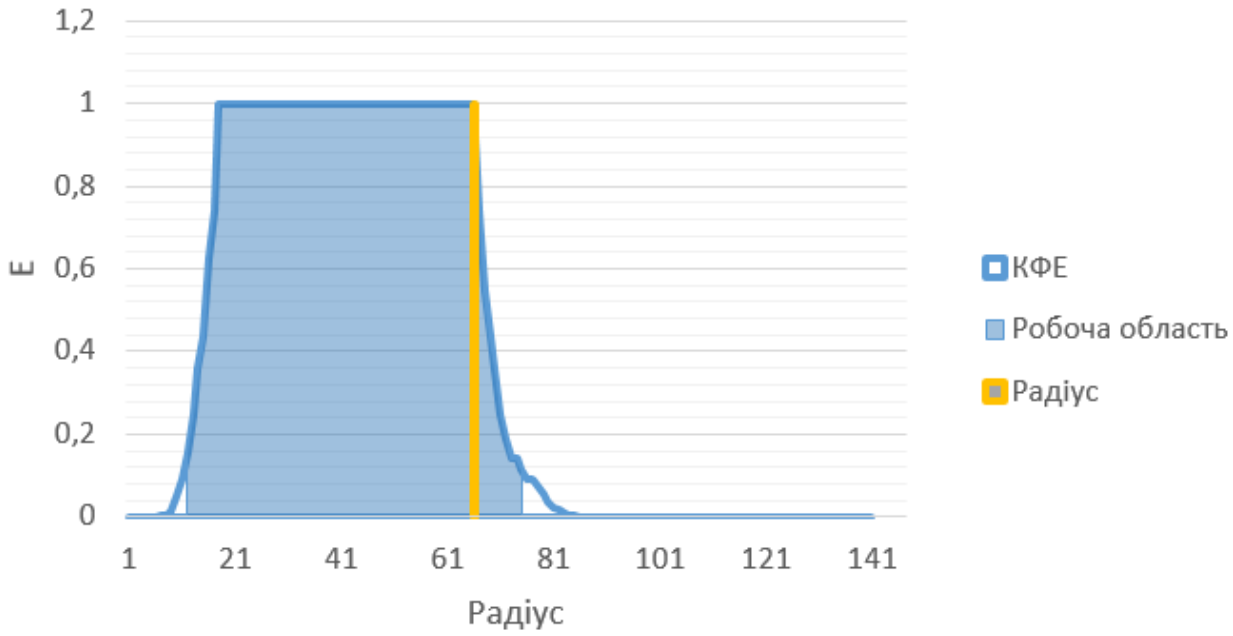

Рисунок 3.13 – Графік залежності інформаційного критерію від радіусу контейнера класу X_2^0 для страти (X_1^0, X_2^0)

Для страти (X_3^0, X_4^0), значення оптимального параметру поля контрольних допусків $\delta = 5$, а радіуси контейнерів розпізнавання відповідно $d_3 = 120, d_4 = 71$.

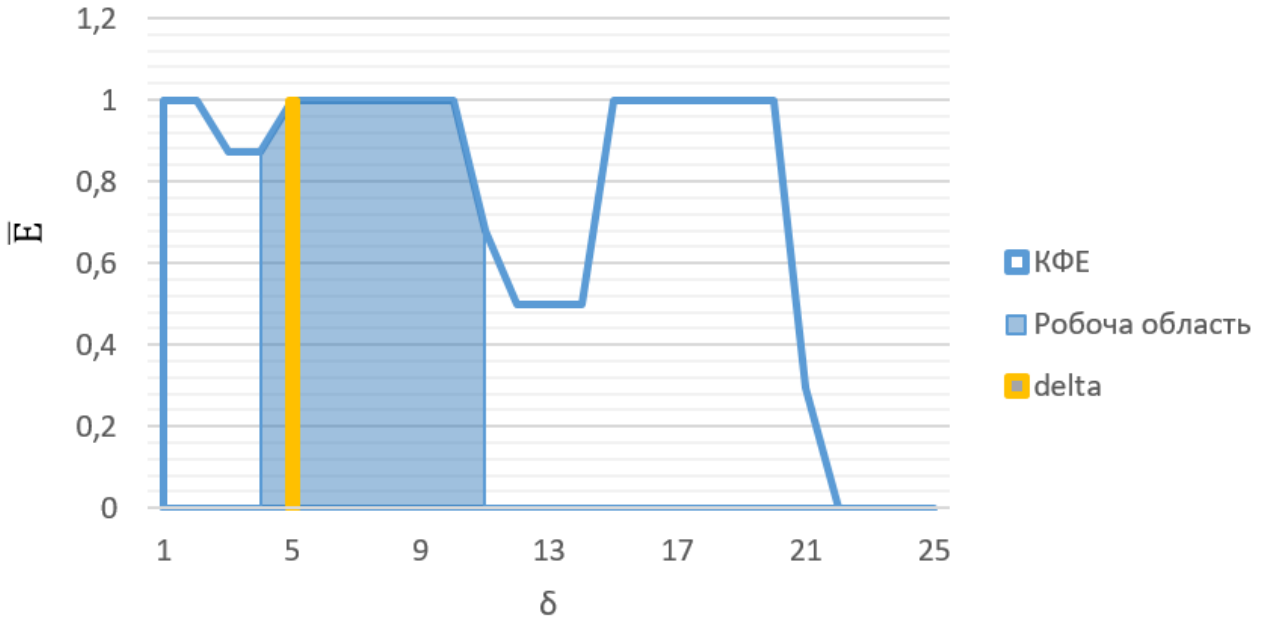

Рисунок 3.14 – Графік залежності інформаційного критерію від параметра поля контрольних допусків для для страти (X_3^0, X_4^0)

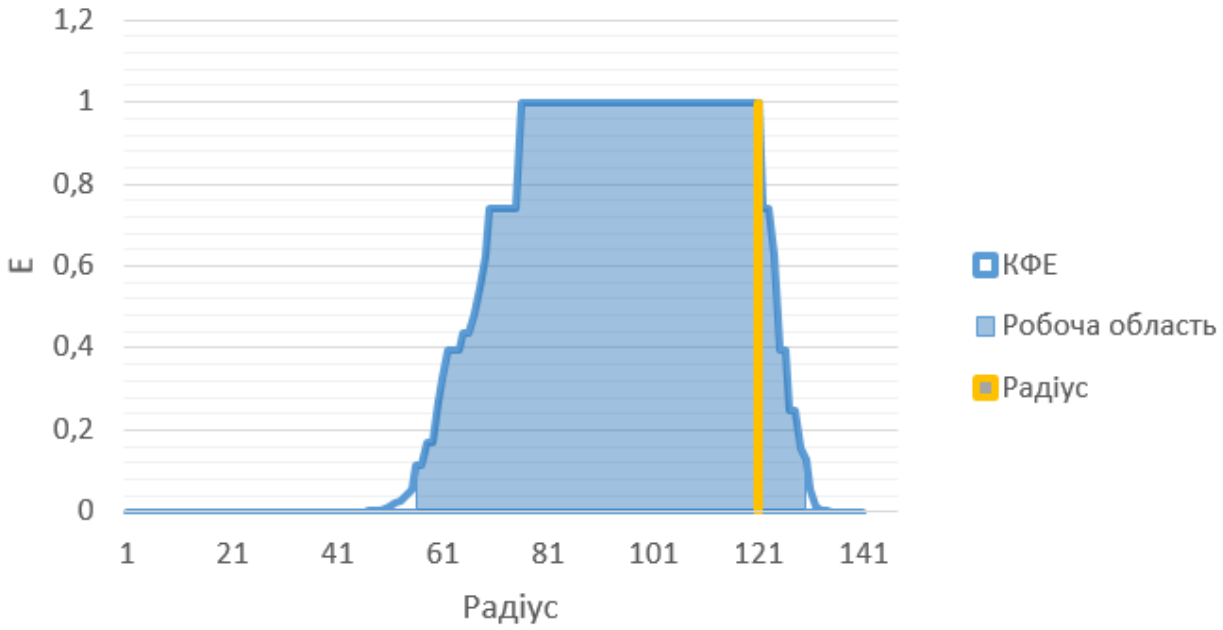

Рисунок 3.15 – Графік залежності інформаційного критерію від радіусу контейнера класу X_3^0 для страти (X_3^0, X_4^0)

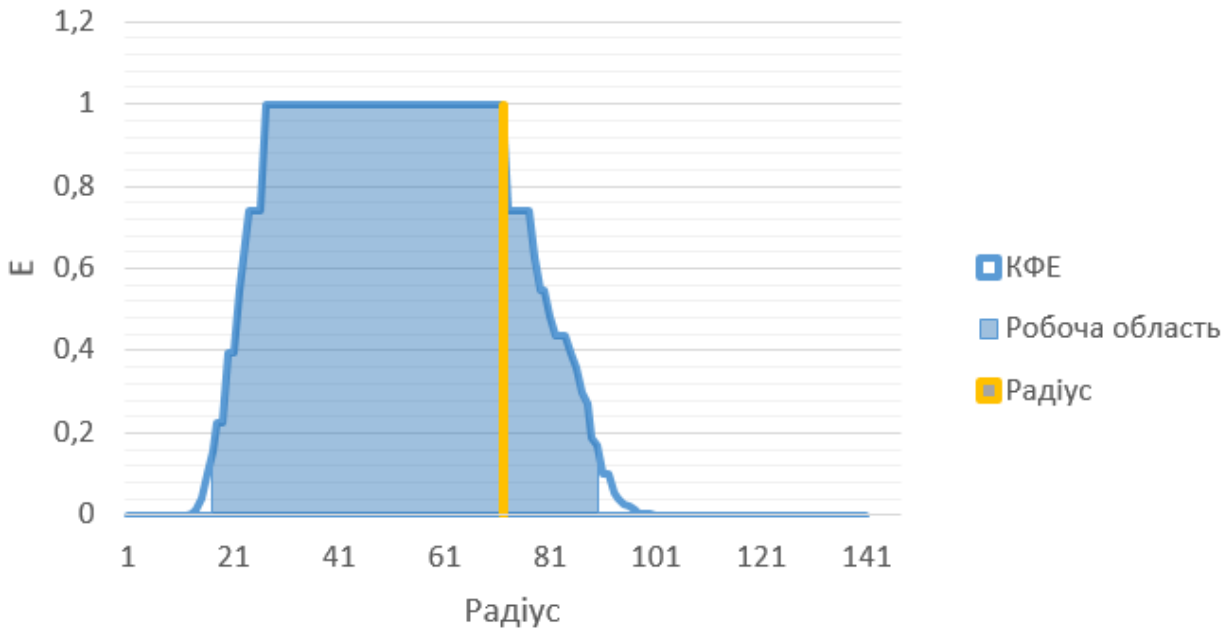

Рисунок 3.16 – Графік залежності інформаційного критерію від радіусу контейнера класу X_4^0 для страти (X_3^0, X_4^0)

В результаті машинного навчання було створено безпомилкові вирішальні правила. На рисунку 3.17 на декурсивному дереві вказані оптимальні для кожної страти параметри функціонування системи прийняття рішень, такі як параметр поля контрольних допусків на ознаки розпізнавання і радіуси гіперсферичних контейнерів класів розпізнавання. Висока достовірність розпізнавання забезпечується саме тим, що кожна страта має власній оптимальний параметр поля контрольних допусків.

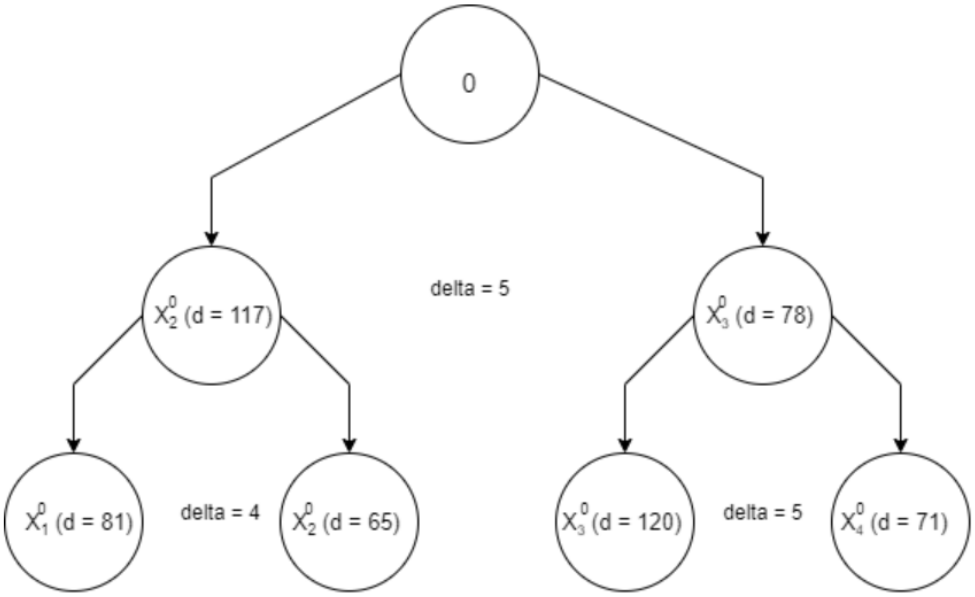

Рисунок 3.17 – Вирішальні правила для машинного навчання

На етапі екзамену було проведено два незалежних експерименти У ході виконання алгоритму екзамену розраховано функції належності та здійснена класифікація тестових даних. Результати експериментів графічно представлені на рисунках 3.18-3.19.

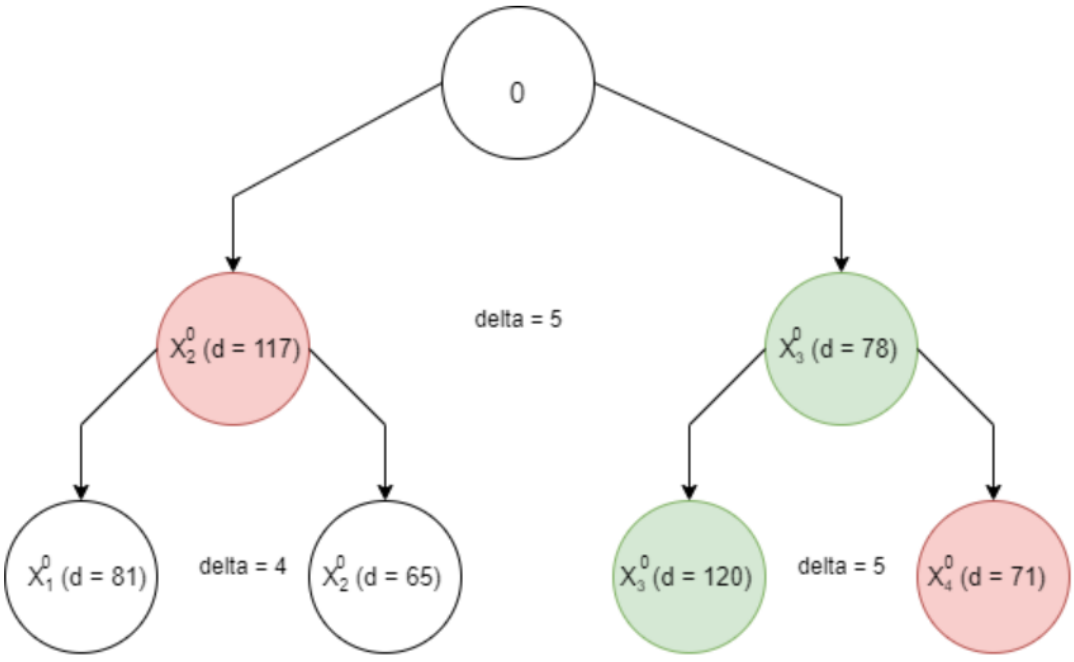

Рисунок 3.18 – Графічне представлення результатів першого експерименту

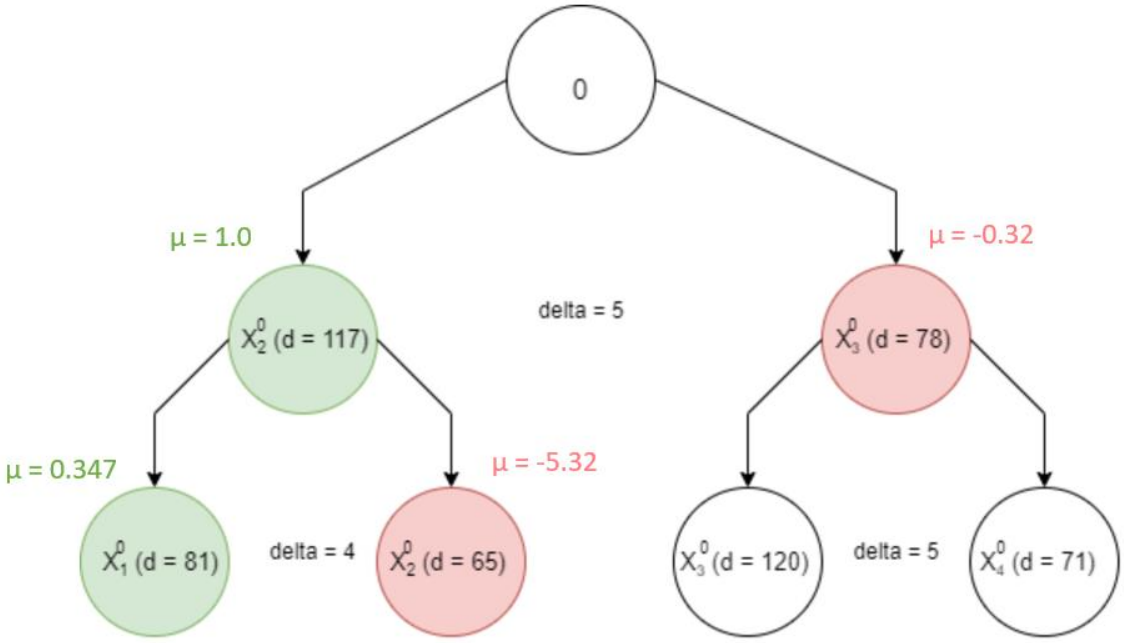

Рисунок 3.19 – Графічне представлення результатів другого експерименту

За результатами обчислення функції належності вхідні вибірки було віднесено до третього та першого класів відповідно, що співпадає з переважною більшістю значень елементів у тестових матрицях

ВИСНОВКИ

У ході виконання кваліфікаційної роботи магістра проаналізовано сучасний стан досліджень у сфері оцінювання якості освіти, розглянуто основні засоби оцінювання та оновлення навчальних планів і програм. Визначено наукову проблему, яка полягає в розробці алгоритмів функціонування інформаційно-аналітичних систем для можливості швидкої адаптації існуючого навчального контенту до сучасних вимог ринку праці. .Проведено огляд методів машинного навчання, обрано та обґрунтовано методологію дослідження в межах інформаційно-екстремальної технології, яка грунтується на максимізації функціональної ефективності системи прийняття рішень, здатної навчатися

Під час реалізації системи оцінки відповідності якості навчального контенту вимогам ринку праці було використано інформаційно-екстремальну інтелектуальну технологію машинного навчання за лінійною та ієрархічною структурою даних з метою пошуку оптимальної структури. На етапі проектування знань системи було розглянуто категоріальні модель машинного навчання, які дозволяють генерувати безпомилкові і точні правила в результаті навчання, вибираються критерії знань для оптимізації навчання, встановлюється вхідний математичний опис і визначаються основні класи розпізнавання. Вхідний математичній опис був створений імітаційним шляхом на основі встановлених правил.

Програмна реалізація було виконано за допомогою об’єктивно-орієнтованої мови програмування Java. В результаті експлуатації системи отримано безпомилкові вирішальні правила, тому розроблена система оцінки відповідності контенту вимогам ринку праці має високу достовірність розпізнавання. Отримані правила можна використовувати як стартові для тестування оцінки якості освіти в режимі екзамену з подальшою можливістю перенавчання, що підтверджується тестовими прикладами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Конкретні п'ять цілей освітніх систем. Звіт Європейської комісії 31.01.2001 р. – URL: http://europa.eu.int. (дата звернення: 10.01.2022)

2. Свіжевська С.А. Акредитація як іміджева стратегія підвищення якості вищої освіти. // Вісник НТУУ «КПІ». Філософія. Психологія. Педагогіка. – Вип. 1’2014. – С. 110-115.

3. СТРАТЕГІЯ РОЗВИТКУ ВИЩОЇ ОСВІТИ В УКРАЇНІ на 2021–2031 роки 2020 р. – URL: https://mon.gov.ua/storage/app/media/rizne/2020/09/25/rozvitku-vishchoi-osviti-v-ukraini-02-10-2020.pdf (дата звернення: 10.04.2023)

4. Бахрушин В.Є. Якість вищої освіти та сучасні підходи до її вимірювання / В.Є.Бахрушин, О.М.Горбань // Освіта і управління. — 2012. — Т. 14, № 4. С. 7-11.

5. Архипова С. П. Якість освіти у контексті вимог сучасності / С. П. Архипова // Вісник Черкаського університету : наук. журнал. – 2008. – Вип. 135. – С. 11–14.

6. Лукіна Т. Моніторинг якості освіти: теорія і практика /Т. Лукіна. – К. : Шкільний світ, 2006. – 128 с.

7. Ляшенко О. І. Якість освіти як основа функціонування і розвитку сучасних систем освіти / О. І. Ляшенко // Моніторинг якості освіти: теорія та практика : матеріали Всеукр. наук.-практ. конф. – К. : КМПУ імені Б. Д. Грінченка, 2009. – С. 9–12.

8. Ляшенко О. І. Якість освіти: проблеми оцінювання, моніторингута управління / О. І. Ляшенко // Розвиток педагогічної і психологічної науки в Україні, 1992–2002 : зб. наук. праць до 10-річчяАПН України ; Академія педагогічних наук України. – Х. : ОВС,2002. – Частина І. – С. 243–249.

9. Моніторинг якості освіти: світові досягнення та українські перспективи ; за заг. ред. О. І. Локшиної. – К. : К.І.С, 2004. – 128 с.

10. Сафонова В. Вдосконалення методологічних підходів до оцінки якості вищої освіти як загальнодержавного пріоритету / В. Сафонова // Економіст. – 2011. – No 1. – С. 54–59.

11. Селезнева Н. А. Качество высшего образования как объект системного исследования : лекция-доклад / Н. А. Селезнева. – М.,2004. – 127 с.

12. Субетто А. И. Оценочные средства и технологии аттестации качества подготовки специалистов в вузах: методология, методика,практика / А. И. Субетто, Н. А. Селезнева. – СПб.-М. : Исследовательский центр проблем качества подготовки специалистов,2004. – 360 с.

13. Субетто А. И. Квалитология образования / А. И. Субетто. – СПб.-М. : Исследовательский центр проблем качества подготовкиспециалистов, 2000. – 226 с.

14. Субетто А. И. От квалиметрии человека – к квалиметрии образования / А. И. Субетто. – М. : Исследовательский центр проблем качества подготовки специалистов, 1993. – 248 с.

15. Субетто А. И. Квалиметрия человека и образования: генезис,становление, развитие, проблемы и перспективы / А. И. Субетто //Материалы XI симпозиума «Квалиметрия в образовании: методология, методика и практика». – М. : Исследовательский центр проблем качества подготовки специалистов, 2006. – 97 с.

16. Фоменко Н. А. Теоретичні засади дослідження якості туристської освіти / Н. А. Фоменко // Гуманіт. вісн. ДВНЗ Переяслав-Хмельн. держ. пед. ун-ту ім. Г. Сковороди : зб. наук. праць. – 2011. – Вип. 23. – С. 253–259.Стаття надій

17. Методи інтелектуального аналізу даних [Електронний ресурс]. – Режим доступу : http://buklib.net/books/24506/

18. Савчук Т. Класифікація даних за допомогою нейронних мереж / Пупко О. //Вісник КрНУ імені Михайла Остроградського. Випуск 2/2022 (133)

19. Лысцов Н.А., Мартышкин А.И. НЕЙРОННЫЕ СЕТИ: ПРИМЕНЕНИЕ И ПЕРСПЕКТИВЫ // Научное обозрение. Педагогические науки. – 2019. –№ 3-2. – С. 35-38.

20. Sarma T.H. et al. An improvement to k-nearest neighbor classifier. 2013.

21. Zhu B., Shoaran M. Tree in Tree: from Decision Trees to Decision Graphs. 2021.

22. Руденко, М.С. Розпізнавання онкологічних захворювань [Текст] / М.С. Руденко, С.С. Мартиненко // Інформатика, математика, механіка (IMM2009): матеріали та програма IV міжвузівської науково-технічної конференції викладачів, співробітників, аспірантів і студентів, 21-24 квітня 2009 р. / Відпов. за вип. С.І. Проценко. - Суми : СумДУ, 2009. - С.

23. Проценко, С.І. Інформаційна технологія розпізнавання електронограм на просвічуючому електронному мікроскопі [Текст]: робота на здобуття кваліфікаційного ступеня магістра; спец.: 122 - комп`ютерні науки (інформатика) / С.І. Проценко; наук. керівник А.С. Довбиш. - Суми: СумДУ, 2021. - 76 с.

24. Теницька, А.О. Система виявлення кібератак. Інформаційна технологія функціонування системи виявлення атак в режимі моніторингу [Текст]: робота на

здобуття кваліфікаційного ступеня бакалавра; спец.: 125 - кібербезпека / А.О. Теницька; наук. кер. А.С. Довбиш. - Суми: СумДУ, 2021. - 65 с.

25. Довбиш А. С. Основи проектування інтелектуальних систем: навч. посіб. / А. С. Довбиш – Суми: Видавництво СумДУ, 2009. — 171 с.

26. Оптимізація параметрів навчання комп’ютеризованої системи діагностування емоційно-психічного стану людини / І. В. Шелехов, Д. В. Прилепа // Радіоелектронні і комп’ютерні системи. - 2014. - № 1. - С. 116–121.

27. Шелехов, І.В. Вибір базового класу при розпізнаванні зображень [Текст] / І.В. Шелехов, К.В. Барило // Вісник Сумського державного університету. Серія Технічні науки. - 2010. - №3, Т.2. - С. 95-102.

28. Оптимізація ієрархічної структури даних інтелектуальної системи функціонального діагностування технічного стану складної машини / А. С. Довбиш, В. І. Зимовець, М. В. Бібик // Вісник Національного технічного університету “ХПІ”. Серія: Системний аналіз, управління та інформаційні технології. - 2018. - № 44. - С. 42-49.

29. Довбиш А. С. Оптимізація словника ознак розпізнавання системикерування, що навчається / А. С. Довбиш, І. В. Шелехов, О. В.Коробченко // Адаптивні системи автоматичного управління :міжвідомчий науково-технічний збірник. – 2015. – № 2(27). – С. 44–50. .

30. Шелехов, І.В. Інформаційно-екстремальна оптимізація словника ознак розпізнавання [Текст] / І.В. Шелехов, М.М. Бірюкова // Вісник Сумського державного університету. Серія Технічні науки. - 2012. - № 3. - С. 46-54.

31. Довбиш А. С. Оптимізація словника ознак розпізнавання дляінформаційно-екстремального гіпереліпсоїдного класифікатора / А. С.Довбиш, В. В. Москаленко

// Вісник Нац. техн. ун-ту “ХПІ”: зб. наук. пр.Темат. вип.: Системний аналіз, управління та інформаційні технології. – Харків: НТУ “ХПІ”. – 2012. – № 30. – С. 65-78.

ДОДАТОК

```
public class Main {

public static void main(String[] args) throws IOException {

int delta = 8;
int etalon1 = 2;

int etalon2 = 0;

int size [] = getAmountCount("class1.txt");
int matrix1 [][] = readFile(size, "class1.txt");
int matrix2 [][] = readFile(size, "class2.txt");
int matrix3 [][] = readFile(size, "class3.txt");
int matrix4 [][] = readFile(size, "class4.txt");

int matrix_test [][] = readFile(size, "c3.txt");

int matrixOfClasses [][][] = {matrix1, matrix2, matrix3, matrix4};

MainAlgorithm data = new MainAlgorithm(matrixOfClasses, delta, etalon1);
data.learing();

HierarchicalTree tree = new HierarchicalTree(matrixOfClasses, etalon2);

tree.run();

}
public static int[] getAmountCount(String filename) throws FileNotFoundException{
int [] size = new int[2];
File file = new File(filename);
Scanner scanner = new Scanner(file);
```

```
int numbersInLine = 0;
int lines = 0;

while (scanner.hasNextLine()) {
lines++;
String[] array = scanner.nextLine().split(" ");
numbersInLine = numbersInLine + array.length;
}

scanner.close();
size[1] = numbersInLine/lines;
size[0] = lines;
return size;
}

public static int[][] readFile(int [] size, String filename) throws FileNotFoundException{
int [][] matrix = new int[size[0]][size[1]];
File file = new File(filename);
Scanner scanner = new Scanner(file);
int lines = 0;

while (scanner.hasNextLine()) {

String[] array = scanner.nextLine().split(" ");
for (int i = 0; i < array.length; i++) {
matrix[lines][i] = Integer.parseInt(array[i]);
}
lines++;
}

scanner.close();
return matrix;
```

```
}
}
```

Class MainAlgrithm

```
package com.julia;
public class MainAlgorithm {
private int matrixOfClasses[][][];
private float [][] standart;
private int amountOfClasses;
private int amountOfRealisations;
private int amountOfAttributes;
private float permits[][];
private int delta;
private int etalonClass;
private int binaryMatrix [][][];
private float coefficient = (float) 0.5;
private int etalonVectors[][];
private int pairs[][];
private int matrixOfPairs[][];
private int matrixCodeDistance[][];
private int codeDistance [][];
private int codeDistancePairs [][];
private float d1 [][];
private float d2 [][];
private float alfa [][];
private float betta [][];
private float d1_new [][];
private float d2_new [][];
private float alfa_new [][];
private float betta_new [][];
private int sumCodeDistance [][];
private int sumCodeDistancePair [][];
```

```
private int sumCodeDistance_new[][];
private int sumCodeDistancePair_new[][];
private double kulback [][];
private double kulback_new [][];
private int optR [];
private int radius [][];
private boolean result;

public MainAlgorithm( int matrixOfClasses[][][], int delta, int etalonClass) {
this.matrixOfClasses = matrixOfClasses;
this.amountOfClasses = matrixOfClasses.length;
this.amountOfAttributes = matrixOfClasses[0][0].length;
this.amountOfRealisations = matrixOfClasses[0].length;
this.permits = new float [2][amountOfAttributes];
this.delta = delta;
this.etalonClass = etalonClass;
this.binaryMatrix = new int [amountOfClasses][amountOfRealisations][amountOfAttributes];
this.etalonVectors = new int [amountOfClasses][amountOfAttributes];
this.pairs = new int [amountOfClasses][2];
this.matrixOfPairs = new int [amountOfClasses][amountOfClasses];
this.matrixCodeDistance= new int [amountOfClasses][amountOfClasses];
this.codeDistance = new int [amountOfClasses][amountOfRealisations];
this.codeDistancePairs = new int [amountOfClasses][amountOfRealisations];
this.d1 = new float [amountOfClasses][amountOfAttributes];
this.d2  = new float [amountOfClasses][amountOfAttributes];
this.alfa = new float [amountOfClasses][amountOfAttributes];
this.betta = new float [amountOfClasses][amountOfAttributes];
this.sumCodeDistance = new int [amountOfClasses][amountOfAttributes];
this.sumCodeDistancePair = new int [amountOfClasses][amountOfAttributes];
this.kulback = new double[amountOfClasses][amountOfAttributes];
this.optR = new int[amountOfClasses];
}
```

```
public void getPermits(){
int sum = 0;
for (int x = 0; x < amountOfAttributes; x++) {
for (int y = 0; y < amountOfRealisations; y++) {
sum += matrixOfClasses[etalonClass][y][x];
}
permits[0][x] = ((float) sum / amountOfRealisations) - delta;
permits[1][x] = ((float) sum / amountOfRealisations) + delta;
sum = 0;
}
}

public int[][] getBinaryMatrix(int [][] matrix){
int [][] binaryMatrix = new int[amountOfRealisations][amountOfAttributes];
for (int y = 0; y < amountOfRealisations; y++) {
for (int x = 0; x < amountOfAttributes; x++) {
if(matrix[y][x] > permits[0][x] && matrix[y][x] < permits[1][x]){
binaryMatrix[y][x] = 1;
}
else{
binaryMatrix[y][x] = 0;
}
}
}
return binaryMatrix;
}

public void getAllBinaryMatrix(){
for (int n = 0; n < amountOfClasses; n++) {
binaryMatrix[n] = getBinaryMatrix(matrixOfClasses[n]);
```

```
}
}

public void learing_new(int delta){
this.delta = delta;
getRadiusDistance();
getPermits();

getAllBinaryMatrix();
getEtalonVector();
getMatrixOfPairs();
getPair();
getCodeDistance();

findAllCharacteristics_new();
getOptRadiusForInterval();
}

public void learingForInterval(){
findOptDeltaForInterval();
if(delta != 0) {
print_new();
}

}

public void run(){
findOptDelta();
if(delta == 0){
learingForInterval();
};
}
```

```
public void learing(int delta){
this.delta = delta;
getPermits();
getAllBinaryMatrix();
getEtalonVector();
getMatrixOfPairs();

getPair();
getCodeDistance();
findAllCharacteristics();
getOptRadius();
}

public void print(){
for (int i = 0; i < permits.length; i++) {
System.out.println("Permit "+i);
for (int j = 0; j < permits[0].length; j++) {
System.out.print(permits[i][j]+" ");
}
System.out.println("");
}

System.out.println("-----");
for (int i = 0; i < amountOfClasses; i++) {
System.out.println("Binary matrix "+i);
for (int j = 0; j < amountOfRealisations; j++) {
for (int k = 0; k < amountOfAttributes; k++) {
System.out.print(binaryMatrix[i][j][k]+" ");
}
System.out.println("");
```

```
}
}

System.out.println("-----");
for (int i = 0; i < amountOfClasses; i++) {
System.out.println("Etalon vector "+i);
for (int j = 0; j < amountOfAttributes; j++) {
System.out.print(etalonVectors[i][j]+" ");
}
System.out.println("");
}

System.out.println("-----");
System.out.println("matrix of pairs");
for (int i = 0; i < amountOfClasses; i++) {
// System.out.println("Etalon vector "+i);
for (int j = 0; j < amountOfClasses; j++) {
System.out.print(matrixOfPairs[i][j]+" ");
}
System.out.println("");
}

System.out.println("-----");
System.out.println("pairs");
for (int i = 0; i < amountOfClasses; i++) {
// System.out.println("Etalon vector "+i);
for (int j = 0; j < 2; j++) {
System.out.print(pairs[i][j]+" ");
}
System.out.println("");
}
```

```
System.out.println("-----");
System.out.println("Code distance own");
for (int i = 0; i < amountOfClasses; i++) {
// System.out.println("Etalon vector "+i);
for (int j = 0; j < amountOfRealisations; j++) {
System.out.print(codeDistance[i][j] + " ");
}
System.out.println("");
}


System.out.println("-----");
System.out.println("code distance pair");
for (int i = 0; i < amountOfClasses; i++) {
// System.out.println("Etalon vector "+i);
for (int j = 0; j < amountOfRealisations; j++) {
System.out.print(codeDistancePairs[i][j] + " ");
}
System.out.println("");
}

System.out.println("-----");
System.out.println("Optimal radius");
for (int i = 0; i < amountOfClasses; i++) {
System.out.println(optR[i]);
}
System.out.println("-----");
System.out.println("Kulbach");
for (int i = 0; i < amountOfClasses; i++) {
System.out.print(kulback[i][optR[i]]+" ");
}
}
```

```
public void print_new(){
for (int i = 0; i < permits.length; i++) {
System.out.println("Permit "+i);
for (int j = 0; j < permits[0].length; j++) {
System.out.print(permits[i][j]+" ");
}
System.out.println("");
}
System.out.println("-----");
for (int i = 0; i < amountOfClasses; i++) {
System.out.println("Binary matrix "+i);
for (int j = 0; j < amountOfRealisations; j++) {
for (int k = 0; k < amountOfAttributes; k++) {
System.out.print(binaryMatrix[i][j][k]+" ");
}
System.out.println("");
}
}

System.out.println("-----");
for (int i = 0; i < amountOfClasses; i++) {
System.out.println("Etalon vector "+i);
for (int j = 0; j < amountOfAttributes; j++) {
System.out.print(etalonVectors[i][j]+" ");
}
System.out.println("");
}

System.out.println("-----");
System.out.println("matrix of pairs");
for (int i = 0; i < amountOfClasses; i++) {
```

```
// System.out.println("Etalon vector "+i);
for (int j = 0; j < amountOfClasses; j++) {
System.out.print(matrixOfPairs[i][j]+" ");
}
System.out.println("");
}

System.out.println("-----");
System.out.println("pairs");
for (int i = 0; i < amountOfClasses; i++) {
for (int j = 0; j < 2; j++) {
System.out.print(pairs[i][j]+" ");
}
System.out.println("");
}

System.out.println("-----");
System.out.println("Code distance own");
for (int i = 0; i < amountOfClasses; i++) {
// System.out.println("Etalon vector "+i);
for (int j = 0; j < amountOfRealisations; j++) {
System.out.print(codeDistance[i][j] + " ");
}
System.out.println("");
}


System.out.println("-----");
System.out.println("code distance pair");
for (int i = 0; i < amountOfClasses; i++) {
// System.out.println("Etalon vector "+i);
for (int j = 0; j < amountOfRealisations; j++) {
```

```
System.out.print(codeDistancePairs[i][j] + " ");
}
System.out.println("");
}
System.out.println("-----");
System.out.println("Optimal radius");
for (int i = 0; i < amountOfClasses; i++) {
System.out.println(radius[optR[i]][0]+" "+radius[optR[i]][1]);
}

System.out.println("-----");
System.out.println("Kulbach");
for (int i = 0; i < amountOfClasses; i++) {
System.out.print(kulback_new[i][optR[i]]+" ");
}
}

public void getEtalonVector(){
int sum = 0;
for (int n = 0; n < amountOfClasses; n++) {
for (int x = 0; x < amountOfAttributes; x++) {
for (int y = 0; y < amountOfRealisations; y++) {
sum += binaryMatrix[n][y][x];
}
if((float)sum/amountOfRealisations >= coefficient){
etalonVectors[n][x] = 1;
}
else {
etalonVectors[n][x] = 0;
}
sum = 0;
}
```

```
}

}

public void getMatrixOfPairs(){
for (int i = 0; i < amountOfClasses; i++) {
for (int j = 0; j < amountOfClasses; j++) {
matrixOfPairs[i][j] = getDistance(etalonVectors[i], etalonVectors[j]);
}
if (i != amountOfClasses - 1){
pairs[i][0] = i + 1;
pairs [i][1] = matrixOfPairs[i][i+1];
}
else {
pairs[i][0] = 0;
pairs [i][1] = matrixOfPairs[i][0];
}
}
}

public int getDistance(int [] vector1, int [] vector2){
int distance = 0;
for (int i = 0; i < amountOfAttributes; i++) {
if(vector1[i] != vector2[i]){
distance++;
}
}
return distance;
}

public void getPair(){
for (int i = 0; i < amountOfClasses; i++) {
```

```
for (int j = 0; j < amountOfClasses; j++) {
if(i != j && matrixOfPairs[i][j] < pairs[i][1]){
pairs[i][1] = matrixOfPairs[i][j];
pairs [i][0] = j;
}
}
}
}

public void  getCodeDistance() {
//int[][][] codeDistance = new int[binaryMatrix.length][2][binaryMatrix[0].length];
for (int i = 0; i < amountOfClasses; i++) {
for (int j = 0; j < amountOfRealisations; j++) {
codeDistance[i][j] = getDistance(etalonVectors[i], binaryMatrix[i][j]);
codeDistancePairs[i][j] = getDistance(etalonVectors[i], binaryMatrix[pairs[i][0]][j]);

}
}
//return codeDistance;
}

public void getSumCodeDistance() {
int sumSk1[][] = new int[amountOfClasses][amountOfAttributes];
int sumSk2[][] = new int[amountOfClasses][amountOfAttributes];
for (int n = 0; n < amountOfClasses; n++) {

for (int r = 0; r < amountOfAttributes; r++) {
for (int i = 0; i < amountOfRealisations; i++) {
if (codeDistance[n][i] <= r) {
sumSk1[n][r] += 1;
}
if (codeDistancePairs[n][i] <= r) {
```

```
sumSk2[n][r] += 1;
}
}
sumCodeDistance[n][r] = sumSk1[n][r];
sumCodeDistancePair[n][r] = sumSk2[n][r];
}
}
}

public void findAllCharacteristics() {
getSumCodeDistance();
for (int n = 0; n < amountOfClasses; n++) { //для какого класса
for (int r = 0; r < amountOfAttributes; r++) {//радиуси 0 - 15 откуда взять значние
d1[n][r] = (float) sumCodeDistance[n][r] / amountOfRealisations;
alfa[n][r] = 1 - d1[n][r];
betta[n][r] = (float) sumCodeDistancePair[n][r] / amountOfRealisations;
d2[n][r] = 1 - betta[n][r];
kulback[n][r] = findKulback( d1[n][r],  d2[n][r], alfa [n][r], betta [n][r])/ findKulback(1,1,0, 0);
}
}
}

public void getRadiusDistance(){

int n = 0;
for (int i = 0; i < amountOfAttributes; i++) {
for (int j = i+1; j < amountOfAttributes; j++) {
n++;

}

}
```

```
this.radius = new int [n][2];
this.sumCodeDistance_new = new int [amountOfClasses][n];
this.sumCodeDistancePair_new = new int [amountOfClasses][n];
this.d1_new = new float[amountOfClasses][n];
this.d2_new = new float[amountOfClasses][n];
this.alfa_new = new float[amountOfClasses][n];
this.betta_new = new float[amountOfClasses][n];
this.kulback_new = new double[amountOfClasses][n];

n = 0;
for (int i = 0; i < amountOfAttributes; i++) {
for (int j = i+1; j < amountOfAttributes; j++) {
//System.out.println(n+": "+i+" "+j);
radius[n][0] = i;
radius[n][1] = j;
n++;

}

}

}
public void getSumCodeDistanceForInterval() {
int sumSk1[][] = new int[amountOfClasses][radius.length];
int sumSk2[][] = new int[amountOfClasses][radius.length];
for (int n = 0; n < amountOfClasses; n++) {
for (int r = 0; r < radius.length; r++) {
for (int i = 0; i < amountOfRealisations; i++) {
if (codeDistance[n][i] < radius[r][1] && codeDistance[n][i] >= radius[r][0]) {
sumSk1[n][r] += 1;
}
if (codeDistancePairs[n][i] < radius[r][1] && codeDistancePairs[n][i] >= radius[r][0]) {
```

```
sumSk2[n][r] += 1;
}
}
sumCodeDistance_new[n][r] = sumSk1[n][r];
sumCodeDistancePair_new[n][r] = sumSk2[n][r];
}
}
}

public void findAllCharacteristics_new() {
getRadiusDistance();
getSumCodeDistanceForInterval();
for (int n = 0; n < amountOfClasses; n++) {
for (int r = 0; r < radius.length; r++) {
d1_new [n][r] = (float) sumCodeDistance_new[n][r] / amountOfRealisations;
alfa_new[n][r] = 1 - d1_new[n][r];
betta_new[n][r] = (float) sumCodeDistancePair_new[n][r] / amountOfRealisations;
d2_new[n][r] = 1 - betta_new[n][r];
kulback_new [n][r] = findKulback( d1_new[n][r],  d2_new[n][r], alfa_new [n][r], betta_new [n][r])/
findKulback(1,1,0, 0);
}
}
}

public void getOptRadiusForInterval() {

double max_k = 0;
int max_r = 0;
for (int n = 0; n < amountOfClasses; n++) { //для какого класса
for (int r = 0; r < radius.length; r++) {//радиуси 0 - 15 откуда взять значние
if (kulback_new[n][r] > max_k && d1_new[n][r] > coefficient && d2_new[n][r] > coefficient) {
max_k = kulback_new[n][r];
```

```
max_r = r;
}
}
optR[n] = max_r;
max_k = 0;
max_r = 0;
}
}
```

```
/**
* Метод для пошуку оптимальних радіусів контейнерів
*/
public void getOptRadius() {
double max_k = 0;
int max_r = -1;
for (int n = 0; n < amountOfClasses; n++) {
for (int r = 0; r < amountOfAttributes; r++) {
if (kulback[n][r] >= max_k && d1[n][r] >= coefficient && d2[n][r] >= coefficient) {
max_k = kulback[n][r];
max_r = r;
}
}
if(max_r == -1){

max_r = 1;
}
optR[n] = max_r;
max_k = 0;
max_r = -1;
```

```
}

}


/**
* Метод для розрахунку поточного значення КФЕ
*/
public double findKulback(double d1, double d2, double alfa, double betta) {
return 0.5 * (Math.log((d1 + d2 + 0.01) / (alfa + betta + 0.01)) / Math.log(2)) * ((d1 + d2) - (alfa +
betta));
}

public void findOptDelta(){
double [] E_max = new double[100];
double [][] dd1 = new double[E_max.length][amountOfClasses];
double [][] dd2 = new double[E_max.length][amountOfClasses];
for (int i = 0; i < 100; i++) {
learing(i);
double E_m = 0;
for (int j = 0; j < amountOfClasses; j++) {
E_m += kulback[j][optR[j]];
dd1[i][j] = d1[j][optR[j]];
dd2[i][j] = d2[j][optR[j]];
}
E_m = E_m/amountOfClasses;
E_max[i] = E_m;
//System.out.println("delta = "+ i+ ", E_max = "+ E_max[i]);
}
double max = 0;
int delta = 0;
for (int i = 0; i < 100; i++) {
```

```
if (E_max[i] > max) {
int d = 0;
for (int j = 0; j < amountOfClasses; j++) {
if (dd1[i][j] > 0.5 && dd2[i][j] > 0.5) {
d++;
}
}
if (d == amountOfClasses) {
max = E_max[i];
delta = i;
}
}
}
this.delta = delta;
learing(delta);
if (delta == 0){
System.out.println("Робоча область не існує");
}
else{
System.out.println("Оптимальна delta "+ delta+ "\nE_max при оптимальній delta "+E_max[delta]);
}
System.out.println("E_max || E_max у робочій області (середнє значення E_max для всіх класів)"
);
for (int i = 0; i < 100; i++) {
System.out.println(E_max[i]);
//System.out.println(i+ " "+ E_max[i]+" "+dd1[i][0]+" "+dd2[i][0]+" "+dd1[i][1]+" "+dd2[i][1]+"
"+dd1[i][2]+" "+
// dd2[i][2]+" "+dd1[i][3]+" "+dd2[i][3]);
if(dd1[i][0] > 0.5 && dd2[i][0]> 0.5 && dd1[i][1]> 0.5 && dd2[i][1]> 0.5 && dd1[i][2]> 0.5 &&
dd2[i][2] > 0.5 && dd1[i][3] > 0.5 && dd2[i][3]> 0.5){
System.out.println(E_max[i]);
```

```
}
}

}

public void findOptDeltaForInterval(){
double [] E_max = new double[100];
double [][] dd1 = new double[E_max.length][amountOfClasses];
double [][] dd2 = new double[E_max.length][amountOfClasses];
for (int i = 0; i < 100; i++) {
learing_new(i);
double E_m = 0;
for (int j = 0; j < amountOfClasses; j++) {
E_m += kulback_new[j][optR[j]];
dd1[i][j] = d1_new[j][optR[j]];
dd2[i][j] = d2_new[j][optR[j]];
}
E_m = E_m/amountOfClasses;
E_max[i] = E_m;
System.out.println("delta = "+ i+ ", E_max = "+ E_max[i]);
}
double max = 0;
int delta = 0;
for (int i = 0; i < 100; i++) {
if (E_max[i] > max) {
int d = 0;
for (int j = 0; j < amountOfClasses; j++) {
if (dd1[i][j] > 0.5 && dd2[i][j] > 0.5) {
d++;
}
}
if (d == amountOfClasses) {
```

```
max = E_max[i];
delta = i;
}
}
}
this.delta = delta;
learing_new(delta);

System.out.println("Оптимальна delta "+ delta+ "\nE_max при оптимальній delta "+E_max[delta]);
System.out.println("E_max || E_max у робочій області (середнє значення E_max для всіх класів)"
);
for (int i = 0; i < 100; i++) {
System.out.print(E_max[i]+" ");
//System.out.println(i+ " "+ E_max[i]+" "+dd1[i][0]+" "+dd2[i][0]+" "+dd1[i][1]+" "+dd2[i][1]+"
"+dd1[i][2]+" "+
// dd2[i][2]+" "+dd1[i][3]+" "+dd2[i][3]);
if(dd1[i][0] > 0.5 && dd2[i][0]> 0.5 && dd1[i][1]> 0.5 && dd2[i][1]> 0.5 && dd1[i][2]> 0.5 &&
dd2[i][2] > 0.5 && dd1[i][3] > 0.5 && dd2[i][3]> 0.5){
System.out.print(E_max[i]);

}
else {}//System.out.println("Робоча область не існує");
System.out.println(" ");
}

//дописати для побудови граф
for (int i = 0; i < radius.length; i++) {
if(radius[i][1] - radius[i][0] ==1){
System.out.println(radius[i][0]+" "+radius[i][1]);

}
}
```

```
}

public void examen(int [][] examMatrix){

//
int binary_exam[][] = getBinaryMatrix(examMatrix);
System.out.println("----Binary exam-----");
for (int i = 0; i < binary_exam.length; i++) {
for (int j = 0; j < binary_exam[0].length; j++) {
System.out.print(binary_exam[i][j]+" ");
}
System.out.println(" ");
}
//
int sk_exam[][] = new int[amountOfClasses][amountOfRealisations];

System.out.println("----Sk_exam------");
for (int i = 0; i < amountOfClasses; i++) {
for (int j = 0; j < amountOfRealisations; j++) {
sk_exam[i][j] = getDistance(etalonVectors[i], binary_exam[j]);
System.out.print(sk_exam[i][j] + " ");
}
System.out.println(" ");
}
System.out.println("---------------");

double[] test_w_max = new double[amountOfClasses];
double summa = 0;
for (int i = 0; i < amountOfClasses; i++) {
for (int j = 0; j < amountOfRealisations; j++) {
if(sk_exam [i][j] >= radius[optR[i]][0] && sk_exam [i][j] <= radius[optR[i]][1]){
summa += 1;
```

```
}
}
test_w_max[i] = summa / amountOfRealisations;
System.out.println(test_w_max[i]);
summa = 0;
}


double max = -1;
int exam_class = -1;
int min_radius = amountOfAttributes;

for (int i = 0; i < amountOfClasses; i++) {
if (test_w_max[i] >= max && test_w_max[i] > 0){
max = test_w_max[i];
exam_class = i;
min_radius = radius[optR[i]][1] - radius[optR[i]][0];

}
}
System.out.println("Exam class: "+exam_class);
}
}
package com.julia;

public class HierarchicalTree {
private int matrixOfClasses[][][];
private float[][] standart;
private int amountOfClasses;
private int amountOfRealisations;
private int amountOfAttributes;
```

```
private float permits[][];
private int delta;
private int etalonClass;
private int binaryMatrix[][][];
private float coefficient = (float) 0.5;
private int etalonVectors[][];
private int pairs[][];
private int matrixOfPairs[][];
private Tree tree;
int n;
//private float [][][]matrixGray;
//private float [][][] matrixRGB;
//BufferedImage source [];
//BufferedImage sourceGray [];

public HierarchicalTree(int matrixOfClasses[][][], int etalonClass) {
// озможно дерево нужно создавать именно в єтом классе
this.matrixOfClasses = matrixOfClasses;
this.amountOfClasses = 2;
this.amountOfAttributes = matrixOfClasses[0][0].length;
this.amountOfRealisations = matrixOfClasses[0].length;
this.permits = new float[2][amountOfAttributes];
//this.delta = ;
this.etalonClass = etalonClass;
this.binaryMatrix = new int[amountOfClasses][amountOfRealisations][amountOfAttributes];
this.etalonVectors = new int[amountOfClasses][amountOfAttributes];
this.pairs = new int[amountOfClasses][2];
this.matrixOfPairs = new int[amountOfClasses][amountOfClasses];


int[][][] strata1 = new int[2][amountOfRealisations][amountOfAttributes];
int[][][] strata2 = new int[2][amountOfRealisations][amountOfAttributes];
```

```
int[][][] strata3 = new int[2][amountOfRealisations][amountOfAttributes];

strata1[0] = matrixOfClasses[1];
strata1[1] = matrixOfClasses[2];

strata2[0] = matrixOfClasses[0];
strata2[1] = matrixOfClasses[1];

strata3[0] = matrixOfClasses[2];
strata3[1] = matrixOfClasses[3];


//иєрархическая структура дерева
Tree root = new Tree(strata1,
new Tree(strata2),
new Tree(strata3));

this.tree = root;
}


public void run() {
System.out.println("Ієрархічна структура");
findAllOptDelta(tree);

}


private Tree baseAlgoritmus(int[][][] strata, int delta) {
Tree newTree;
float permits[][] = new float[2][strata[0].length];
int[][][] binaryMatrix = new int[strata.length][strata[0].length][strata[0][0].length];
```

```
int[][] etalonVectors = new int[strata.length][strata[0].length];
int[][][] codeDistance = new int[strata.length][2][strata[0].length];
int optRadius[] = new int[strata.length];
permits = getPermits(strata, delta);
binaryMatrix = getAllBinaryMatrix(strata, permits);
etalonVectors = getAllEtalonVector(binaryMatrix);
codeDistance = getSK(binaryMatrix, etalonVectors);
int[][][] sum_sk = getSumCodeDistance(codeDistance, strata[0][0].length);
float d1[][] = getD1(sum_sk, strata[0][0].length, codeDistance);
float alfa[][] = getAlfa(sum_sk, strata[0][0].length, d1);
float betta[][] = getBetta(sum_sk, strata[0][0].length, codeDistance);
float d2[][] = getD2(sum_sk, strata[0][0].length, betta);
double kulbach[][] = findAllKulback(d1, d2, alfa, betta);
optRadius = getOptRadius(codeDistance, strata[0][0].length, kulbach, d1, d2);

newTree = new Tree(permits, binaryMatrix, etalonVectors, codeDistance, optRadius, d1, alfa, betta,
d2, kulbach, sum_sk);
newTree.delta = delta;
return newTree;
}


public float[][] getPermits(int[][][] strata, int delta) {
int sum = 0;
int amountOfRealisations = strata[0].length;
int amountOfAttributes = strata[0][0].length;
float permits[][] = new float[2][amountOfAttributes];
for (int x = 0; x < amountOfAttributes; x++) {
for (int y = 0; y < amountOfRealisations; y++) {
sum += strata[etalonClass][y][x];
}
permits[0][x] = ((float) sum / amountOfRealisations) - delta;
```

```
permits[1][x] = ((float) sum / amountOfRealisations) + delta;
sum = 0;
}
return permits;
}


public int[][][] getAllBinaryMatrix(int[][][] strata, float[][] permits) {
int[][][] binaryMatrix = new int[strata.length][strata[0].length][strata[0][0].length];
for (int n = 0; n < strata.length; n++) {
binaryMatrix[n] = getBinaryMatrix(strata[n], permits);
}
return binaryMatrix;
}

public int[][] getBinaryMatrix(int[][] matrix, float[][] permits) {
int[][] binaryMatrix = new int[matrix.length][matrix[0].length];
for (int y = 0; y < matrix.length; y++) {
for (int x = 0; x < matrix[0].length; x++) {
if (matrix[y][x] > permits[0][x] && matrix[y][x] < permits[1][x]) {
binaryMatrix[y][x] = 1;
} else {
binaryMatrix[y][x] = 0;
}
}
}
return binaryMatrix;
}

public int[][] getAllEtalonVector(int[][][] binaryMatrix) {
int[][] etalonVectors = new int[binaryMatrix.length][binaryMatrix[0].length];
for (int n = 0; n < binaryMatrix.length; n++) {
```

```
etalonVectors[n] = getEtalonVector(binaryMatrix[n]);
}
return etalonVectors;
}

public int[] getEtalonVector(int[][] binaryMatrix) {
int[] etalonVector = new int[binaryMatrix[0].length];
int sum = 0;
for (int x = 0; x < binaryMatrix[0].length; x++) {
for (int y = 0; y < binaryMatrix.length; y++) {
sum += binaryMatrix[y][x];
}
if ((float) sum / binaryMatrix.length >= coefficient) {
etalonVector[x] = 1;
} else {
etalonVector[x] = 0;
}
sum = 0;
}
return etalonVector;
}

public int[][][] getSK(int[][][] binaryMatrix, int[][] etalonVectors) {
int[][][] codeDistance = new int[binaryMatrix.length][2][binaryMatrix[0].length];
for (int i = 0; i < binaryMatrix.length; i++) {
for (int j = 0; j < binaryMatrix[0].length; j++) {
codeDistance[i][i][j] = getS(etalonVectors[i], binaryMatrix[i][j]);
codeDistance[i][getPair(i)][j] = getS(etalonVectors[i], binaryMatrix[getPair(i)][j]);

}
}
return codeDistance;
```

```
}

private int getPair(int value) {
if (value == 1)
return 0;
else
return 1;
}

private int getS(int[] vector1, int[] vector2) {
int sum = 0;
for (int k = 0; k < vector1.length; k++) {
if (vector1[k] != vector2[k]) {
sum += 1;
}
}
return sum;
}

public int[][][] getSumCodeDistance(int[][][] codeDistance, int maxR) {
int sum_sk[][][] = new int[codeDistance.length][2][amountOfAttributes];
for (int n = 0; n < codeDistance.length; n++) {

for (int r = 0; r < amountOfAttributes; r++) {
for (int i = 0; i < amountOfRealisations; i++) {
if (codeDistance[n][n][i] <= r) {
sum_sk[n][n][r] += 1;
}
if (codeDistance[n][getPair(n)][i] <= r) {
sum_sk[n][getPair(n)][r] += 1;
}
}
```

```
}
}
return sum_sk;
}

public float[][] getD1(int sum_sk[][][], int maxR, int[][][] codeDistance) {
float d1[][] = new float[sum_sk.length][maxR];
for (int n = 0; n < sum_sk.length; n++) {

for (int r = 0; r < maxR; r++) {
d1[n][r] = (float) sum_sk[n][n][r] / codeDistance[0][0].length;
}
}

return d1;
}

public float[][] getAlfa(int sum_sk[][][], int maxR, float d1[][]) {
float alfa[][] = new float[sum_sk.length][maxR];
for (int n = 0; n < sum_sk.length; n++) {

for (int r = 0; r < maxR; r++) {
alfa[n][r] = 1 - d1[n][r];
}
}
return alfa;
}

public float[][] getBetta(int sum_sk[][][], int maxR, int[][][] codeDistance) {
float betta[][] = new float[sum_sk.length][maxR];
for (int n = 0; n < sum_sk.length; n++) {
```

```
for (int r = 0; r < maxR; r++) {
betta[n][r] = (float) sum_sk[n][getPair(n)][r] / codeDistance[0][0].length;
}
}
return betta;
}

public float[][] getD2(int sum_sk[][][], int maxR, float betta[][]) {
float d2[][] = new float[sum_sk.length][maxR];
for (int n = 0; n < sum_sk.length; n++) {
for (int r = 0; r < maxR; r++) {
d2[n][r] = 1 - betta[n][r];
}
}
return d2;
}

/**
* Метод для пошуку оптимальних радіусів контейнерів
*/
public int[] getOptRadius(int[][][] codeDistance, int maxR, double kulback[][], float d1[][], float d2
[][]){

int[] optR = new int[codeDistance.length];
double max_k = 0;
int max_r = -1;
for (int n = 0; n < codeDistance.length; n++) {
for (int r = 0; r < amountOfAttributes; r++) {
if (kulback[n][r] >= max_k && d1[n][r] > coefficient && d2[n][r] > coefficient) {
max_k = kulback[n][r];
max_r = r;
}
```

```
}
if(max_r == -1){

max_r = 1;
}
optR[n] = max_r;
max_k = 0;
max_r = 0;
}

return optR;

}

private double[][] findAllKulback(float d1[][], float d2[][], float alfa[][], float betta[][]) {
double e[][] = new double[d1.length][d1[0].length];
for (int n = 0; n < d1.length; n++) {
for (int r = 0; r < d1[0].length; r++) {
e[n][r] = findKulback(d1[n][r], d2[n][r], alfa[n][r], betta[n][r]) / findKulback(1, 1, 0, 0);
}
}
return e;
}

/**
* Метод для розрахунку поточного значення КФЕ
*/
private double findKulback(double d1, double d2, double alfa, double betta) {
return 0.5 * (Math.log((d1 + d2 + 0.01) / (alfa + betta + 0.01)) / Math.log(2)) * ((d1 + d2) - (alfa +
betta));
}
```

```
/**
* Метод для пошуку оптимального delta
*/

public void findAllOptDelta(Tree tree) {
System.out.println("Страта");
Tree root = findOptDelta(tree.value);
tree.permits = root.permits;
tree.binaryMatrix = root.binaryMatrix;
tree.etalonVectors = root.etalonVectors;
tree.codeDistance = root.codeDistance;
tree.optRadius = root.optRadius;
tree.d1 = root.d1;
tree.alfa = root.alfa;
tree.betta = root.betta;
tree.d2 = root.d2;
tree.kulbach = root.kulbach;
tree.delta = root.delta;



System.out.println("Оптимальна дельта = "+tree.delta);
System.out.println("Class 0 ");
System.out.println("Opt radius = " + tree.optRadius[0]);
for (int i = 0; i < tree.kulbach[0].length; i++) {
System.out.print(tree.kulbach[0][i]+ " ");
if (tree.d1[0][i] > coefficient && tree.d2[0][i] > coefficient){
System.out.print(tree.kulbach[0][i]);
}
System.out.println("");
```

```
}

System.out.println("Class 1 ");
System.out.println("Opt radius = " + tree.optRadius[1]);
for (int i = 0; i < tree.kulbach[1].length; i++) {
System.out.print(tree.kulbach[1][i]+ " ");
if (tree.d1[1][i] > coefficient && tree.d2[1][i] > coefficient){
System.out.print(tree.kulbach[1][i]);
}
System.out.println("");
}


if (tree.left != null) {
findAllOptDelta(tree.left);
}

if (tree.right != null) {
findAllOptDelta(tree.right);
}
}

public Tree findOptDelta(int[][][] matrix) {
Tree studing;
double[] E_max = new double[100];
double[][] dd1 = new double[E_max.length][matrix.length];
double[][] dd2 = new double[E_max.length][2];
for (int i = 0; i < 100; i++) {
studing = baseAlgoritmus(matrix, i);


double E_m = 0;
```

```
for (int j = 0; j < matrix.length; j++) {
E_m += studing.kulbach[j][studing.optRadius[j]];
dd1[i][j] = studing.d1[j][studing.optRadius[j]];
dd2[i][j] = studing.d2[j][studing.optRadius[j]];
}
E_m = E_m / matrix.length;
E_max[i] = E_m;

}

double max = 0;
int delta = 0;
for (int i = 0; i < 100; i++) {

if (E_max[i] > max) {
int d = 0;
for (int j = 0; j < matrix.length; j++) {
if (dd1[i][j] > 0.5 && dd2[i][j] > 0.5) {
d++;
}
}

if (d == matrix.length) {
max = E_max[i];
delta = i;
}
}
}

studing = baseAlgoritmus(matrix, delta);
```

```
System.out.println("Дані для дельта");
for (int i = 0; i < 100; i++) {
System.out.print(E_max[i]+" ");
if(dd1[i][0] > 0.5 && dd2[i][0]> 0.5 && dd1[i][1]> 0.5 && dd2[i][1]> 0.5 ){
System.out.print(E_max[i]);
}
System.out.println("");
}
System.out.println("-----------------------");

return studing;
}

private void exam(Tree tree, int[][] matrix){
int binary_exam[][] = getBinaryMatrix(matrix, tree.permits);

int sk_exam[][] = new int[tree.value.length][tree.value[0].length];

System.out.println("----Sk_exam------");
for (int i = 0; i < tree.value.length; i++) {
for (int j = 0; j < tree.value[0].length; j++) {
sk_exam[i][j] = getS(tree.etalonVectors[i], binary_exam[j]);
}
}
System.out.println("---------------");


double[][] test_w = new double[tree.value.length][tree.value[0].length];
double[] test_w_max = new double[tree.value.length];
double summa = 0;

for (int i = 0; i < tree.value.length; i++) {
```

```
for (int j = 0; j < tree.value[0].length; j++) {
test_w[i][j] = 1 - ((double) sk_exam[i][j] / tree.optRadius[i]);
summa += test_w[i][j];
}
test_w_max[i] = summa / tree.value[0].length;
summa = 0;
}

double max = -1;
int exam_class = -1;

for (int i = 0; i < tree.value.length; i++) {
if (test_w_max[i] > max && test_w_max[i] >= 0) {
max = test_w_max[i];
exam_class = i;
}
}
System.out.println("Exam class: "+exam_class);
System.out.println("Exam value 0: "+test_w_max[0] + ", Exam value 1: "+ test_w_max[1]);

if (tree.left != null && exam_class == 0) {
exam(tree.left, matrix);
}
if (tree.right != null && exam_class == 1) {
exam(tree.right, matrix);
}
}

public void make_exam(int[][] matrix) {
exam(tree, matrix);
}
}
```