МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КОМПЛЕКСНА КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

«Інформаційно-аналітична система адаптації навчального контенту до вимог ринку праці. Функціонування системи в режимі моніторингу»

Завідувач випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Довбиш А.С.**

Студента групи ІНм-81н **Сальник К.О.**

СУМИ 2020

Сумський державний університет

(назва вузу)

Факультет *ЕлІТ* Кафедра *Комп'ютерних наук*

Спеціальність *«Інформатика»*

Затверджую:
зав. кафедрою _____
_____
"_____" _____ 20___р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТОВІ

Сальник Крістіні Олександрівні

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи)
Інформаційно-аналітична система адаптації навчального контенту до вимог ринку праці. Функціонування системи в режимі моніторингу

затверджую наказом по інституту від "____" _____ 20 ___ р. № _____

2. Термін здачі студентом закінченого проекту (роботи)

3. Вхідні данні до проекту (роботи)

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Аналіз проблеми та постановки задачі дослідження
2) Інформаційно-екстремальна інтелектуальна технологія
3) Інформаційне та програмне забезпечення системи оцінки якості навчального контенту

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_____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка
1.	*Аналіз проблеми та постановки задачі дослідження*		
2.	*Інформаційно-екстремальна інтелектуальна технологія*		
3.	*Інформаційне та програмне забезпечення системи оцінки якості навчального контенту*		
4.	*Оформлення пояснювальної записки до дипломної роботи*		

Студент – дипломник _____
(підпис)

Керівник проекту _____
(підпис)

РЕФЕРАТ

Записка: 74 стор., 15 рис., 4 табл., 1 додаток, 15 джерел.

Мета роботи – розробка інформаційно-аналітичної системи адаптації навчального контенту випускової кафедри до вимог ринку праці та забезпечення її функціонування для постійного контролю якості освітніх програм в режимі моніторингу.

Об'єкт дослідження – процес оцінки якості отриманих знань випускниками за допомогою машинного навчання.

Предмет досліджень – інформаційна та програмна складові системи, формування вхідного опису, машинне навчання з оптимізацією параметрів та алгоритм функціонування системи в режимі екзамену.

Методи дослідження – інформаційно-екстремальна інтелектуальна технологія.

Результати – в ході дослідження розроблено програмне забезпечення інформаційно-аналітичної системи оцінки навчального контенту згідно положень інформаційно-екстремальної інтелектуальної технології. Сформовано вхідний математичний опис, розглянуто та реалізовано категорійну модель машинного навчання з оптимізацією параметрів за допомогою інформаційних критеріїв, реалізовано функціонування системи на етапі екзамену та розроблено інтерфейс програми.

ІНФОРМАЦІЙНО-ЕКСТРЕМАЛЬНА ІНТЕЛЕКТУАЛЬНА ТЕХНОЛОГІЯ, ОЦІНКА НАВЧАЛЬНОГО КОНТЕНТУ, ІНФОРМАЦІЙНО-АНАЛІТИЧНА СИСТЕМА, РЕЖИМ МОНІТОРИНГУ, КАТЕГОРІЙНА МОДЕЛЬ, ІНФОРМАЦІЙНИЙ КРИТЕРІЙ, ВХІДНИЙ МАТЕМАТИЧНИЙ ОПИС

ЗМІСТ

ВСТУП

У зв’язку зі збільшенням бази наукових знань та технологій для різних процесів виробництва існує потреба у вдосконаленні навчального процесу, саме тому постає необхідність постійного моніторингу якості та актуальності навчального контенту. Якість знань, отриманих студентами, характеризується їх фундаментальністю та затребуваністю у трудовій діяльності. На основі даних про якість освітніх програм формуються рекомендації щодо структури та плану навчального процесу і навчально-методичного забезпечення у закладах вищої освіти. Такі рекомендації необхідні у навчально-педагогічній сфері навчального закладу з метою підготовки кваліфікованих кадрів.

Існує ряд інструментів визначення якості освіти, що використовуються в усьому світі. До них відносяться: внутрішнє та зовнішнє оцінювання, акредитація, аудит, бенчмаркінг. Такі заходи проводяться з метою перевірки наявності необхідних ресурсів, відповідності навчального процесу сучасним тенденціям розвитку освіти, контролю діяльності вищого начального закладу та якості підготовки кадрів на всіх етапах і рівнях навчання. Крім того, сучасні критерії оцінки якості освіти припускають свободу для навчальних закладів у формуванні освітніх планів, увазі до якості підготовки фахівців, постійному вдосконаленні навчальних програм з метою їх актуалізації, стимулювання інновацій у сфері освіти.

Сучасні системи управління навчанням та освітнім контентом, такі як Moodle або Lotus Workplace Collaborative Learning, є досить потужними програмними комплексами, що полегшують навчальний процес, але вони не вирішують питання актуалізації навчального контенту відповідно до вимог ринку праці. Тож постає необхідність залучити до вирішення задачі методи інтелектуального аналізу та машинного навчання, що базуються на роботі з аналітичними даними. Такими методами є структурні, ознакові, кореляційні, нейронні мережі, але вони не забезпечують достатньої точності розпізнавання

класів, тому є потреба розширення класу об’єктів для побудови математичного опису в межах інформаційно-екстремальної інтелектуальної технології.

Метою дослідження є розробка інформаційно-аналітичної системи та забезпечення її функціонування для постійного моніторингу якості освітніх програм за алгоритмами інформаційно-екстремальної інтелектуальної технології. Об’єктивний аналіз результатів роботи системи послужить додатковим показником для визначення навчального плану та заходів для його вдосконалення. Введення даної системи в начальний процес має важливе організаційне, навчальне, розвивальне значення і сприяє актуалізації знань та удосконаленню професійних навичок студентів.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКИ ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Аналіз системи оцінки якості навчальної діяльності

У період глобальних змін особливо гостро постає проблема якості вищої освіти, забезпечення системи її оцінки та постійного вдосконалення. Якість освітнього процесу складає цілий комплекс характеристик навчальної діяльності, які передбачають ефективне практичне застосування отриманих знань та формування у здобувачів освіти професійної свідомості [1]. Тому важливим завданням є забезпечення постійного контролю основних показників якості освіти та розробка рекомендацій щодо поліпшення усіх складових підготовки фахівців. Однією із таких складових є удосконалення системи моніторингу та передбачення потреб ринку праці з метою адаптації навчального контенту до сучасних вимог.

Тісна взаємодія вищої освіти та ринку праці в умовах трансформаційних процесів в економічних системах є важливим напрямком змін освітнього середовища, розширення доступу до навчання і вибору професії, підвищення рівня економічного і соціального розвитку країни [2]. Упродовж останніх десятиріч спостерігаються відчутні зміни у відношенні вищої освіти та ринку праці. У ході стрімкого технічного прогресу виникає зростання вимог до знань працівників у сфері техніки та інформаційних технологій, крім того досвід з обміну студентами та міжнародної співпраці спонукає уряди держав приділити увагу проблемі якості освіти, адже Європейський Союз та відкритий ринок вимагають нових підходів до якості навчального процесу [3].

Тож, перед вищими навчальними закладами постає необхідність вирішення низки питань із забезпечення високого рівня якості підготовки студентів, наприклад, впровадження системи збору інформації про якість освітнього процесу впродовж всього навчального періоду, від абітурієнта до працюючого фахівця, обробки та аналізу накопичених даних та прийняття відповідних рішень [4]. Основними проблемами, що вимагають уваги є:

- невідповідність знань та умінь молодих фахівців до вимог на виробництві через розрив сформованих зав'язків між виробництвом, наукою та освітою;
- забезпечення якісного оцінювання рівня конкурентоздатності випускників навчального закладу та виявлення способів з його поліпшення;
- невідповідність кваліфікації фахівців до вимог ринку праці та роботодавців;
- проблема із залучення роботодавців у процес формування плану для професійної підготовки фахівців;
- проблема наступності у системі дворівневої підготовки майбутніх фахівців.

Про якість освіти, що надається вищим навчальним закладом, свідчать такі показники [4], як кількість організацій-партнерів навчального закладу їх рівень; відображення отриманих результатів у сфері бізнесу; внесення бізнесовими організаціями навчального закладу до рейтингів; тривалість навчальної діяльності закладу освіти; рівень заробітної плати випускників закладу; постійне оновлення навчальних програм у відповідності до потреб на ринку праці та розвитку виробничих процесів; кількість абітурієнтів та випускників; кількість запитів від організацій та підприємств-працедавців щодо працевлаштування випускників вищого навчального закладу та інші.

Існуючий взаємозв'язок між вищою освітою та ринком праці є не достатньо ефективним, вітчизняний ринок освітніх послуг не може у повній мірі задовольнити потребу у кваліфікованих кадрах та швидко реагувати на зміни, що відбуваються на ринку праці [2]. Тому, ситуація, що склалася на ринках освітніх послуг і праці, потребує введення інноваційних заходів щодо налагодження їх взаємодії.

З 80-х років XX ст. система освіти стала набагато складнішою та диверсифікованою, що призвело до автономізації вищих начальних закладів. Зміни, що відбулись у сфері науки та техніки передбачали створення такої

системи, що дозволила б приймати рішення на місцевому рівні. Крім того, однією з основних ідей Болонського процесу є надання більшої самостійності вищим навчальним закладам за умови впровадження системи зовнішнього оцінювання якості (external quality assessment) [3], що допомагає здійснити підтримку та підвищення якості, а також поетапно впровадити механізми забезпечення достатньої якості.

Основними задачами системи оцінки якості навчальної діяльності [5] є:

- забезпечення якісних освітніх програм згідно з національними та міжнародними стандартами враховуючи зміни у європейському просторі;
- заохочення та допомога для вищих навчальних закладів у діяльності з покращання якості навчання;
- впровадження обміну інформацією щодо якості та рекомендацій з підвищення якості в межах країни та в світі;
- співпраця з іншими навчальними закладами у напрямку забезпечення якості.

Для досягнення вказаних цілей європейські агенції із забезпечення якості використовують наступні процедури з оцінювання [3,5], які отримали широке розповсюдження у всьому світі:

- оцінювання – комплекс заходів з внутрішнього і зовнішнього систематичного аналізу, яка застосовується для визначення якості діяльності вищого навчального закладу та створення рекомендацій щодо її поліпшення та модернізаціїї;
- акредитація – процедура, що дозволяє встановити відповідність якості діяльності вищого навчального закладу прийнятим стандартам з метою надання закладу певного статусу або підтвердити його право продовжувати освітню діяльність;
- аудит – захід з оцінювання механізмів забезпечення якості, коли вони використовуються в різних вищих навчальних закладах з метою перевірки їх ефективності;

- бенчмаркінг – підхід до оцінювання, за якого вищий навчальний заклад визначає рівень власної освітньої діяльності та порівнює з передовими національними і закордонними практиками з метою самовдосконалення.

На рисунку 1.1 відображено результат аналізу частоти використання інститутами з забезпечення якості освіти описаних заходів з оцінювання.

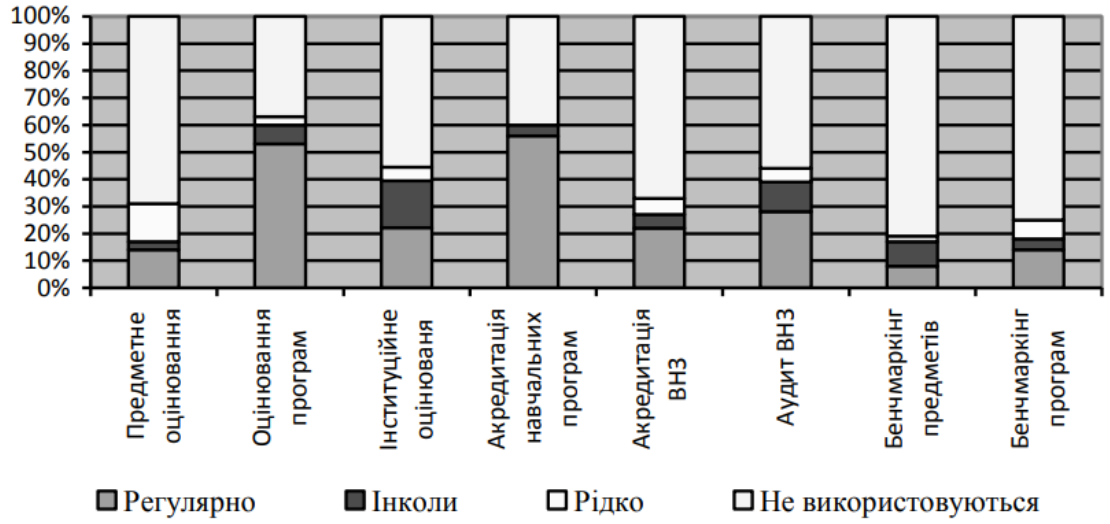


Рисунок 1.1 – Частота використання різних типів оцінки якості європейськими агенціями [3]

За наведеними даними можна зробити висновок, що акредитація та оцінювання навчальних програм – це два заходи, які найбільш часто використовуються в європейському забезпеченні якості, з подальшим зменшенням частоти – аудит, акредитація та оцінювання всіх видів навчальної діяльності закладу, оцінювання дисциплін, тестування начальних програм та дисциплін [5].

На сьогодні існує велика кількість програмних продуктів управління навчальною діяльністю із широким спектром функцій, таких як, розробка та поширення онлайн-матеріалів, забезпечення сумісного доступу до навчального контенту, організація дистанційного навчання, управління групами студентів, оцінювання якості викладання та знань студентів, опитування і тестування та інше. В багатьох навчальних системах інтегровані технології, що дозволяють легко систематизувати та поширити навчальні

матеріали: wiki (web-сайт, що дозволяє учасникам начального процесу самостійно наповнювати його вміст, створюючи спільний проект чи базу знань), блоги (web-сайти, призначені для регулярного обміну інформацією), RSS (формат, призначений для розповсюдження інформації, анонсів статей, відео та аудіо матеріалів та інше).

Learning Management System (системи керування навчанням) – системи, що об'єднують засоби організації навчальної діяльності та адміністрування, а також методи дистанційного електронного навчання (e-Learning) [6], що використовують хмарні рішення, технології електронних матеріалів, мультимедіа та Internet, з метою полегшення доступу до навчальних ресурсів та сервісів, обміну інформацією та роботи в групі.

Архітектуру систем електронного навчання складає зрозумілий інтерфейс, з яким працює студент, та робоча платформа, що побудована на основі бази даних, та містить усі дані і ресурси навчальної системи, інформацію про студентів та структуру навчального процесу. Такі системи можуть знаходитись у вільному доступі (системи з відкритим кодом), до них належать: Moodle, Claronline, OLAT, Efront learning, Dokeos, Sakai, Ilias, Colloquia, Dokeos, Atutor, Adobe Acrobat Connect Pro; або обмеженому доступі (комерційні системи), прикладами таких систем є: WebCT, SumTotalSystem, Desire2Learn, KnowkedgePlanet, Blackboard або Angel [7].

Learning Content Management System (системи керування навчальним контентом) – системи, що дозволяють здійснювати контроль та розробку навчальних планів і програм, адмініструвати та відстежувати дистанційне навчання, створювати та розміщувати навчальні матеріали. Такі системи корисні як для методичних працівників, які розробляють план курсу, його наповнення та графік навчання, так і для студентів, за допомогою чого вони можуть отримати доступ до розкладу та журналу оцінок або зареєструватись на навчання. До таких систем відносяться програмні продукти Lotus Workplace Collaborative Learning та Talentsoft Learning Content Management System.

1.2 Методи інтелектуального аналізу даних

У результаті вдосконалення програмного та апаратного забезпечення, розвитку засобів накопичення та збереження даних, технологій та алгоритмів їх оброблення, створено значну кількість баз даних у різних сферах життя. Дослідження структур баз даних та інформаційних технологій породили новий підхід до зберігання та маніпулювання цими даними для подальшого прийняття рішень. Таким підходом є інтелектуальний аналіз даних (Data Mining) [8], він являє собою процес вилучення корисної інформації та зразків даних з величезних інформаційних масивів і виконується за допомогою сучасних статистичних та математичних методів. До методів інтелектуального аналізу даних відносяться: класифікація, дерева рішень, регресія, правила асоціації, штучний інтелект, нейронні мережі, кластеризація, генетичний алгоритм та інші.

Класифікація (Classification) – метод, що має найбільше застосування, полягає у знаходженні набору ознак, що характеризують досліджувані класи об’єктів, та створенні правил, за якими новий об’єкт можна віднести до певного класу. Процес класифікації даних передбачає етапи навчання та самої класифікації об’єктів. Дані класифікаційного тесту використовуються для оцінки точності правил класифікації, і якщо вона є прийнятною, то правила можуть застосовуватись до нових наборів даних. Під час алгоритму навчання попередньо класифіковані приклади використовуються для визначення набору ознак, які необхідні для правильної побудови алгоритму класифікатора. Виділяють декілька основних типів класифікаційних моделей, до яких відносяться: класифікація за методом опорних векторів (Support Vector Machines), баєсів класифікатор (Bayesian Classification), нейронні мережі (Neural Networks), метод індукції дерев рішень (Decision Tree Induction) [9].

Кластеризація (Clustering) – це більш складний процес аналізу даних, використовується для поділу об’єктів на групи, при цьому їх класи наперед невідомі. Метою кластеризації є створення кластерів таким чином, щоб

об’єкти в межах одного кластеру були схожі один на одного за певним критерієм. Для вирішення задач кластеризації застосовуються такі методи як: метод ієрархічного поділу (Partitioning Methods), кластеризація на основі щільності (Density Based Methods), карти Кохонена (Self-Organizing Map) [8].

Асоціація (Associations) – полягає у побудові асоціативних закономірностей, де, на відміну від попередніх методів, здійснюється аналіз не на основі ознак об’єктів, а на основі подій, що одночасно відбуваються. Відомими алгоритмами з пошуку асоціативних правил є багаторівнева асоціація (Multilevel Association Rule), багатовимірна асоціація (Multivariate Association Rule), кількісна асоціація (Quantitative Association Rule).

Прогнозування (Predication) – метод регресійного аналізу, що здійснює моделювання зв’язку між однією або кількома значеннями залежних і незалежних показників з метою відновлення пропущених значень чи передбачення наступних. До задач прогнозування зазвичай застосовуються методи регресії, наприклад, лінійна регресія (Linear Regression), багатовимірна лінійна регресія (Multivariate Linear Regression), нелінійна регресія (Nonlinear Regression), багатовимірна нелінійна регресія (Multivariate Nonlinear Regression) [9].

Генетичні алгоритми (Genetic algorithm) – застосовуються для рішення складних комбінаторних задач та задач оптимізації. На початковому етапі здійснюється кодування вихідних закономірностей бази даних, що мають назву хромосоми. Популяція (набір) хромосом, сформована у результаті зіставлення хромосом, проходить процедури репродукції, мутації та генетичної композиції з метою створення нових екземплярів. Таким чином відбувається моделювання еволюційного розвитку і продовжується декілька циклів до досягнення критерію зупинки.

Нейронні мережі (Neural networks) – клас систем, що побудовані по аналогії нервової тканини із нейронів. Нейронні мережі найкраще визначають закономірності або тенденції в структурах даних і добре підходять для прогнозування та реалізації передачі даних. Найбільш поширеним типом

нейронної мережі є багатошаровий перцептрон, він імітує роботу нейронів в мережі, що має ієрархічну структуру. Нейрони вищого рівня з'єднуються за допомогою своїх входів із виходами нейронів нижчого рівня, які в свою чергу отримують значення вхідних параметрів (сигналів). Під час проходження сигналів через мережу, їх значення змінюються залежного від ваг між нейронними сполученнями. Таким чином, на найвищому рівні отримується результат, що розглядається як відповідь мережі. Навчання такої системи полягає у пошуку оптимальних значень ваг сполучень, що приводять до отримання правильних відповідей. Недоліком нейронної мережі є великий обсяг навчальної вибірки, а також структура мережевих зав'язків, яка нагадує «чорну скриньку» [8,9].

Описані алгоритми є лише невеликою частиною сукупності відомих методів інтелектуального аналізу даних, адже він поєднує у собі широкий спектр математичних інструментів, від звичайного статистичного аналізу до новітніх кібернетичних методів і технологій, що дозволяють вирішити низку задач, які виникають на практиці в різних галузях науки і техніки.

1.3 Формалізована постановка задачі

Метою дослідження є розробка інформаційно-аналітичної системи адаптації навчального контенту випускової кафедри до вимог ринку праці та забезпечення її функціонування в режимі моніторингу. Розглянемо формалізовану постановку задачі згідно положень інформаційно-екстремальної інтелектуальної технології.

Нехай задано алфавіт класів $\{X_m^0 \mid m = \overline{1,M}\}$, де M – кількість класів розпізнавання, що визначають оцінку якості навчальних матеріалів з ряду дисциплін, за національною шкалою оцінювання. Навчальна матриця, що відображає властивості M класів розпізнавання, формується за наперед визначеними ваговими коефіцієнтами та позначається наступним чином: $\left\| y_{m,i}^{(j)} \right\|, i = \overline{1,N},\ j = \overline{1,n}$, де N – кількість ознак розпізнавання; n – кількість

векторів-реалізацій. Відомий вектор структурованих параметрів машинного навчання з відповідними обмеженнями: $g = \langle g_1, \dots, g_{\xi_1}, \dots, g_{\Xi_1} \rangle$.

За допомогою оптимізованих параметрів вектора функціонування інтелектуальної системи на етапі машинного навчання необхідно побудувати вирішальні правила для прийняття рішень на етапі екзамену про належність реалізацій екзаменаційної матриці до одного з класів з заданого алфавіту.

Вхідними даними на етапі екзамену є:

1) M – кількість класів розпізнавання, які характеризують різні функціональні стани системи;

2) $\{X_m^* \mid m = \overline{1, M}\}$ – еталонні вектори-реалізації образу, що задають центри контейнерів для кожного класу;

3) $\{d_m^*\}$ – оптимальні радіуси для побудови контейнерів;

4) $\left\{x^{(j)} \,\middle|\, j = \overline{1, n}\right\}$ – бінарні вектори-реалізації образу, що проходить процедуру розпізнавання;

5) $\left\{\delta_{k,i}^* \,\middle|\, i = \overline{1, N}\right\}$ – оптимізована під час навчання система контрольних допусків на ознаки розпізнавання.

Для побудови бінарної матриці на етапі екзамену взято рівень селекції ρ_m = 0.5.

Процедура екзамену здійснюється на основі аналізу результатів обчислення функції належності, що для класу X_m^0 розраховується за формулою (1.1):

$$\mu_m = 1 - \frac{d\left(x_m^* \oplus x^{(j)}\right)}{d_m^*} \qquad (1.1)$$

Для виконання поставленої задачі необхідно розробити алгоритмічне та програмне забезпечення для інформаційно-аналітичної системи «Educational Monitoring», реалізувати функціонування за алгоритмами навчання та моніторингу. Передбачити формування вхідних даних на основі результатів опитування щодо якості змісту навчання та розробити зручний інтерфейс системи.

2 ІНФОРМАЦІЙНО-ЕКСТРЕМАЛЬНА ІНТЕЛЕКТУАЛЬНА ТЕХНОЛОГІЯ

2.1 Основні положення інформаційно-екстремальної інтелектуальної технології

Сутність інформаційно-екстремальної інтелектуальної технології (ІЕІ-технологія) [10] визначається перетворенням визначеного наперед нечіткого розбиття простору ознак у множину класів за допомогою знаходження оптимальних значень просторово-часових параметрів функціонування системи шляхом застосування методу ітерацій. Алгоритм машинного навчання зводиться до пошуку глобального екстремуму функції інформаційного критерію в області її визначення (робочій області) і разом з тим, побудови замкнених роздільних гіперповерхонь-контейнерів.

Особливістю використання методів, що ґрунтуються на ІЕІ-технології, є те, що таке перетворення проводиться у ході оптимізації системи контрольних допусків, внаслідок чого змінюються значення ознак розпізнавання. За допомогою даного алгоритму побудови системи можна отримати безпомилкові вирішальні правила за багатовимірною навчальною матрицею. Отже, ІЕІ-технологія одночасно здійснює нормалізацію образу у відношенні до еталонного вектору та реалізує процес навчання системи завдяки побудові вирішальних правил.

На початковому етапі реалізації алгоритму здійснюється визначення алфавіту класів $\{X_m^0 \mid m = \overline{1,M}\}$. Враховуючи, що розбиття на класи розпізнавання у бінарному просторі ознак Ω є нечітким $\widetilde{\Re}^{|M|}$, виконуються вирази (2.1):

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

Оскільки, у бінарному просторі контейнер класу розпізнавання представлений у формі гіперпаралелепіпеду, то для його представлення в радіальному базисі можна використати гіперсферу, що включає усі вершини гіперпаралелепіпеду. При цьому, $x_m \in X_m^0$ – еталонний вектор, що визначає центр контейнера K_m^0, d_m – радіус сферичного контейнера, що розраховується за формулою для знаходження відстані Хеммінга (2.2):

$$d_m = d(x_m \oplus \lambda) = \sum_{i=0}^{N}\left(x_{m,i} \oplus \lambda_i\right), \quad (2.2)$$

де $x_{m,i}$ – i-та координата еталонного вектора x_m;

λ_i – i-та координата вектора λ, вершина якого належить контейнеру $K_m^0 \in X_m^0$.

В радіальному базисі встановлення оптимального контейнера K_m^0 відбувається за допомогою перетворення в гіперсферичний габарит, а його радіус змінюється на кожній ітерації за формулою (2.3):

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

де k – величина збільшення радіуса K_m^0;

h – крок збільшення радіуса;

G_m^d – область допустимих значень радіуса контейнера d_m.

Припустимо, що класи X_k^0 та X_l^0 – найближчі сусіди, тобто мають мінімальну міжцентрову відстань $d(x_k \oplus x_l)$, де x_k та x_l – еталонні вектори цих класів відповідно. Для виключення ситуації, коли центр одного класу потрапляє у межі контейнера іншого, умови (2.1) доповнюються виразом (2.4):

$$\left(\forall X_k^0 \in \widetilde{\Re}^{|M|}\right)\left(\forall X_l^0 \in \widetilde{\Re}^{|M|}\right)[X_k^0 \neq X_l^0 \rightarrow \left(d_k^* < d(x_k \oplus x_l)\right) \& \\ \&\left(d_l^* < d(x_k \oplus x_l)\right)], \quad (2.4)$$

де d_k^* та d_l^*– оптимальні радіуси контейнерів K_k^0 та K_l^0.

Як зазначалося раніше, етап навчання здійснюється ітераційно шляхом оптимізації параметрів функціонування інтелектуальної системи та пошуку глобального екстремуму усередненого значення критерію функціональної ефективності для класів розпізнавання $\{X_m^0\}$.

Вектор структурованих просторово-часових параметрів функціонування системи має наступний вигляд (2.5):

$$g = \langle g_1, \dots, g_{\xi_1}, \dots, g_{\Xi_1}, f_1, \dots f_{\xi_2}, \dots f_{\Xi_2} \rangle, \Xi_1 + \Xi_2 = \Xi, \quad (2.5)$$

де $\langle g_1, \dots, g_{\xi_1}, \dots, g_{\Xi_1} \rangle$ – генотипні параметри, що визначають параметри розподілу реалізацій (радіуси), $R_{\varepsilon_1}(g_1, \dots, g_{\xi_1}, \dots, g_{\Xi_1}) \leq 0$;

$\langle f_1, \dots f_\xi, \dots f_{\Xi_2} \rangle$ – фенотипні параметри, що визначають форму контейнерів класів (контрольні допуски на ознаки, параметри оптимізації словника ознак, рівні селекції координат еталонних векторів, параметри впливу оточення та ін.), $R_{\varepsilon_2}(f_1, \dots f_{\xi_2}, \dots f_{\Xi_2}) \leq 0$.

Описаний алгоритм навчання з оптимізацією параметрів функціонування інтелектуальної системи для $M > 2$ має наступний формалізований вигляд (2.6):

$$(\forall g_{\xi_1} \in g)(\forall f_{\xi_2} \in g)(\exists g_{\xi_1} \in G_{\xi_1}) \{if\ E^* = max_{G_E} \bar{E}\ then$$

$$g^*_{\xi_1} = arg \langle \left[max_{G_{\xi_1}} \dots [max_{G_1} \left[max_{F_{\xi_2}} [\dots [max_{F_1} \bar{E}] \dots] \right] \right] \dots] \rangle\ else$$

$$(if\ \xi_1 \leq \Xi_1\ then\ \xi_1 = \xi_1 + 1\ else\ STOP)\}, \xi_1 = \overline{1, \Xi_1}, \xi_2 = \overline{1, \Xi_2} \quad (2.6)$$

де $G_{\xi_1}, \dots, G_1$ – області допустимих значень генотипних параметрів навчання;

$\bar{E} = \frac{1}{M} \sum_{m=1}^{M} E_m$ – середнє значення критерію функціональної ефективності;

G_E – область значень функції інформаційного критерію функціональної ефективності;

$g^*_{\xi_1}$ – оптимальне значення параметра;

$F_{\xi_2}, \dots, F_1$ – області допустимих значень фенотипних параметрів навчання;

E_m – інформаційний критерій функціональної ефективності навчання для розпізнавання класу X^0_m.

Кількість ітерацій залежить від числа параметрів навчання у векторі параметрів функціонування (2.5). За допомогою послідовної оптимізації кожного параметру можна збільшити екстремальне значення критерію функціональної ефективності, що у свою чергу збільшує імовірність того, що рішення, прийняте на етапі моніторингу буде правильним. Оптимізація контрольних допусків є важливим кроком на етапі навчання, адже вони

визначають не лише відповідні ознаки розпізнавання, а і параметри розподілу реалізацій.

На практиці часто зустрічається задача, що полягає у порівнянні образу з еталоном $(\mathrm{M} = 2)$, наприклад, контроль топології друкованих плат, розпізнавання текстури тканини, ідентифікація об'єктів, пошук фрагментів зображень, дактилоскопія та інші. Для даного класу задач алгоритм навчання має наступний вигляд (2.7):

$$\left(\forall g_{\xi_1} \in g\right)\left(\forall f_{\xi_2} \in g\right)\left(\exists g_{\xi_1} \in G_{\xi_1}\right)\{if\ E_1^* = max_{G_E} E_1\ then$$

$$g_{\xi_1}^* = arg\,\langle\left[max_{G_{\xi_1}}[\ldots[max_{G_1}\left[max_{F_{\xi_2}}[\ldots[max_{F_1} E_1]\ldots]\right]\right]\ldots]]\rangle\ else$$

$$(if\ \xi_1 \le \Xi_1\ then\ \xi_1 = \xi_1 + 1\ else\ STOP)\},\xi_1 = \overline{1,\Xi_1},\xi_2 = \overline{1,\Xi_2} \qquad (2.7)$$

де E_1 – інформаційний критерій функціональної ефективності навчання для розпізнавання класу X_1^0.

Отже, у разі прийняття гіпотези чіткої (нечіткої) компактності реалізацій образу, алгоритм машинного навчання, згідно IEI-технології, зводиться до послідовної нормалізації вхідного математичного опису за допомогою перетворення параметрів розподілу реалізацій задля знаходження оптимальних геометричних параметрів контейнерів класів розпізнавання [10]. Такі параметри дозволяють задовольнити максимальну несхожість між класами, а мірою відображеної різноманітності є найбільше значення інформаційного критерію функціональної ефективності в межах робочої області.

Параметри контейнерів, що характеризують максимальну функціональну ефективність, застосовуються на етапі моніторингу шляхом прийняття рішень за чіткими вирішальними правилами. Згідно з детерміновано-статистичним підходом, побудований класифікатор забезпечує достовірність рішень, наближену до максимально асимптотичної, що розраховується на етапі навчання. При цьому важливо гарантувати для навчальної та екзаменаційної матриці однакові статистичну стійкість та однорідність матриць.

2.2 Категорійні моделі машинного навчання

Одним із основних етапів аналізу при проектуванні інформаційного та алгоритмічного забезпечення інтелектуальної системи, що навчається, є категорійне моделювання слабо формалізованих процесів [10]. Завдяки реалізації методів IEI-технології в рамках функціонального підходу процес машинного навчання можна візуалізувати за допомогою категорійної моделі, що має вигляд орієнтованого графу.

У процесі пошуку оптимальних значень структурованих просторово-часових параметрів функціонування системи за допомогою ітераційних циклів базовий алгоритм виконується у внутрішньому колі циклічної процедури навчання [10,12]. Категорійна модель базового алгоритму навчання має структуру, показану на рисунку 2.1:

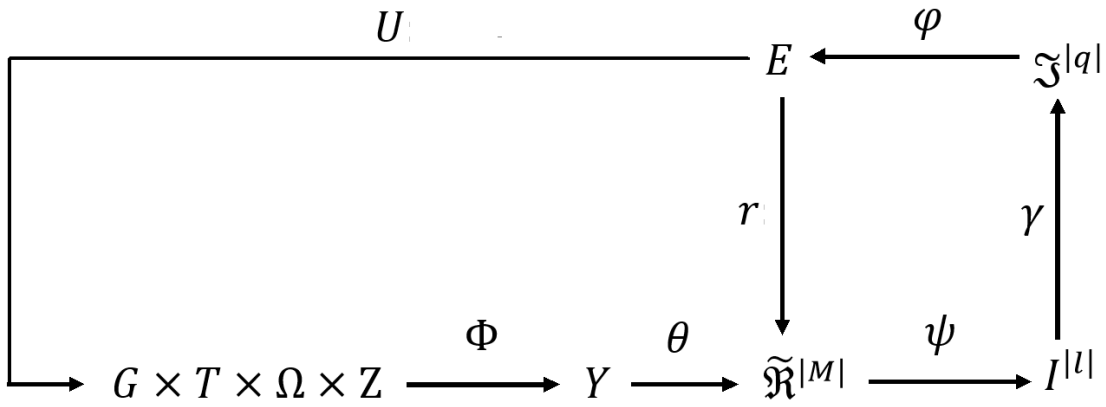


Рисунок 2.1 – Категорійна модель базового алгоритму навчання інформаційної системи [10]

Вхідний математичний опис інформаційної системи представлений у вигляді структури параметрів:

$$\Delta_{\mathrm{B}} = \langle G,\ T,\ \Omega,\ \mathrm{Z},\ \mathrm{Y};\ \Phi \rangle, \tag{2.8}$$

де G – множина вхідних сигналів;

T – значення точок часу отримання інформації;

Ω – множина ознак розпізнавання;

Z – функціональні стани інформаційної системи;

Y – вибірка значень, з яких формується вхідна навчальна матриця.

Враховуючи нечітке розбиття класів розпізнавання $\widetilde{\Re}^{|M|}$ у бінарному просторі ознак Ω з відповідними умовами, категорійна модель навчання за базовим алгоритмом подається як граф з відображенням множин за допомогою наступних операторів:

1) $\theta : Y \to \widetilde{\Re}^{|M|}$ – оператор нечіткої факторизації простору ознак;

2) $\psi : \widetilde{\Re}^{|M|} \to I^{|l|}$, де l – число статичних гіпотез, – оператор класифікації, що здійснює перевірку належності реалізацій $\left\{x_m^{(j)} \mid j = \overline{1,n}\right\}$ до класу розпізнавання X_m^0;

3) $\gamma : I^{|l|} \to \Im^{|q|}$, де $q = l^2$ – кількість точнісних характеристик, – оператор, що здійснює формування множини точнісних характеристик $\Im^{|q|}$ за допомогою аналізу статистичних гіпотез;

4) $\varphi : \Im^{|q|} \to E$ – оператор, що розраховує масив значень інформаційного критерію функціональної ефективності;

5) $r : E \to \widetilde{\Re}^{|M|}$ – оператор, що здійснює поєднання кінців контуру оптимізації геометричних параметрів $\widetilde{\Re}^{|M|}$ за допомогою пошуку глобального максимуму критерію функціональної ефективності в межах робочої області його визначення;

6) $U : E \to G \times T \times \Omega \times \mathrm{Z}$ – оператор, що регламентує процедуру машинного навчання і формує універсум випробувань, що створює діагностичні дані;

7) $\Phi : G \times T \times \Omega \times \mathrm{Z} \to Y$ – оператор формування вибіркової множини.

Вхідний математичний опис категорійної моделі алгоритму навчання з оптимізацією геометричних параметрів контейнерів класів і системи контрольних допусків [13] отримуємо шляхом доповнення структури (2.8):

$$\Delta_{\mathrm{B}} = \langle G,\ T,\ \Omega,\ \mathrm{Z},\ \mathrm{Y},\ X;\ \Phi_1, \Phi_2 \rangle, \qquad (2.9)$$

де X – бінарна навчальна матриця.

Для відображення алгоритму навчання з оптимізацією параметрів контейнерів і системи допусків категорійна модель 2.1 доповнюється наступними операторами:

8) $\Phi_2 : Y \to X$ – оператор перетворення матриці в бінарну форму;

9) $\delta_1 : E \to D$ та $\delta_2 : D \to Y$, де D – система контрольних допусків на ознаки розпізнавання, – здійснює оптимізацію контрольних допусків;

10) $U : V \to G \times T \times \Omega \times \mathrm{Z}$, де V – множина типів вирішальних правил, – оператор, що регламентує процес навчання.

Категорійна модель алгоритму інформаційно-екстремального навчання з оптимізацією параметрів контейнерів класів і системи контрольних допусків зображена на рисунку 2.2:

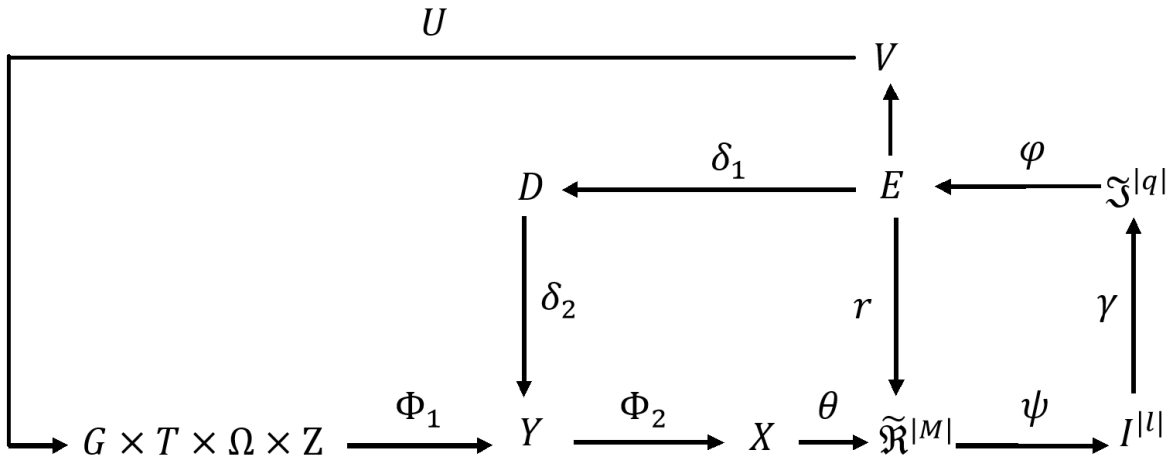


Рисунок 2.2 – Категорійна модель алгоритму навчання з оптимізацією параметрів контейнерів класів та системи контрольних допусків [13]

Отже, розглянуті категорійні моделі машинного навчання інформаційної системи виступають як узагальнена структура алгоритму інформаційно-екстремального синтезу системи підтримки прийняття рішень, що навчається. Категорійні моделі машинного навчання можуть розширюватись в результаті введення інших параметрів оптимізації інформаційної системи. Застосування категорійних моделей дозволяє реалізувати ефективну роботу сучасних інтелектуальних технологій, що орієнтовані на прогресивне функціональне програмування.

2.3 Інформаційні критерії оптимізації навчання

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 (2.10):

$$\mathrm{E}_m^{(k)} = \frac{I_m^{(k)}}{I_{max}^{(k)}} = \frac{H_m^{(k)} - H_m^{(k)}(\gamma)}{H_m^{(k)}}, \tag{2.10}$$

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 (2.10) та заміни умовної ентропії за формулою Байєса:

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

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

$$\left.\log_2\frac{D_{1,m}^{(k)}(d)}{\beta_m^{(k)}(d)+D_{1,m}^{(k)}(d)} + \frac{D_{2,m}^{(k)}(d)}{\alpha_m^{(k)}(d)+D_{2,m}^{(k)}(d)}\log_2\frac{D_{2,m}^{(k)}(d)}{\alpha_m^{(k)}(d)+D_{2,m}^{(k)}(d)}\right) \qquad (2.11)$$

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 (2.11) є нелінійним, взаємно-неоднозначним функціоналом від описаних точнісних характеристик, а його визначення вимагає пошуку робочої області в ході навчання.

Враховуючи, що інформаційний критерій (2.11) залежить від точнісних характеристик, то за репрезентативного обсягу навчальної вибірки слід застосовувати наступні оцінки (2.12):

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

$$\beta_m^{(k)}(d) = \frac{K_{3,m}^{(k)}}{n_{min}}; \qquad D_{2,m}^{(k)}(d) = \frac{K_{4,m}^{(k)}}{n_{min}}, \qquad (2.12)$$

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 (2.12) до виразу (2.11) отримано робочу формулу для розрахунку ентропійного інформаційного критерію (2.13):

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

$$\left. + \frac{K_{3,m}^{(k)}}{K_{1,m}^{(k)} + K_{3,m}^{(k)}} \log_2 \frac{K_{3,m}^{(k)}}{K_{1,m}^{(k)} + K_{3,m}^{(k)}} + \frac{K_{4,m}^{(k)}}{K_{2,m}^{(k)} + K_{4,m}^{(k)}} \log_2 \frac{K_{4,m}^{(k)}}{K_{2,m}^{(k)} + K_{4,m}^{(k)}}\right) \qquad (2.13)$$

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:

$$\Lambda = \log_2 \frac{P_{t,m}^{(k)}}{P_{f,m}^{(k)}} = \log_2 \frac{p(\mu_m)p\left(\frac{\gamma_{1,k}}{\mu_m}\right) + p(\mu_c)p\left(\frac{\gamma_{2,k}}{\mu_c}\right)}{p(\mu_m)p\left(\frac{\gamma_{2,k}}{\mu_m}\right) + p(\mu_c)p\left(\frac{\gamma_{1,k}}{\mu_c}\right)}, \qquad (2.14)$$

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:

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

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

$$+\beta_m^{(k)}(d))] = \log_2 \left(\frac{2-\left(\alpha_m^{(k)}(d)+\beta_m^{(k)}(d)\right)}{\alpha_m^{(k)}(d)+\beta_m^{(k)}(d)} \right) * [1 - (\alpha_m^{(k)}(d) + \beta_m^{(k)}(d))] \qquad (2.15)$$

Нормований варіант інформаційної міри Кульбака представлений наступним виразом (2.16):

$$\mathrm{E}_{K,m}^{(k)} = \frac{\mathrm{E}_{K\,m}^{(k)}}{\mathrm{E}_{K\,max}^{(k)}}, \qquad (2.16)$$

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 (2.15).

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 (2.12) підставлено до виразу (2.15):

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

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.

3 ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ОЦІНКИ ЯКОСТІ НАВЧАЛЬНОГО КОНТЕНТУ

3.1 Формування вхідного математичного опису

На початковому етапі проектування інформаційного забезпечення системи здійснюється процедура формування вхідного математичного опису, яка полягає у вирішенні ряду задач:

– визначення словника ознак та алфавіту класів розпізнавання;

– пошук мінімального обсягу репрезентативної навчальної матриці, забезпечення її статистичної стійкості та однорідності;

– формування системи нормованих допусків на ознаки розпізнавання, що визначають область допустимих значень системи контрольних допусків.

Метою побудови вхідного математичного опису [10] є отримання багатовимірної навчальної матриці $\left\| y_{m,i}^{(j)} \mid m = \overline{1,M}; i = \overline{1,N}; j = \overline{1,n} \right\|$ типу «об’єкт-властивість». У загальному вигляді, вхідний математичний опис подається як теоретико-множинні структури, що наведені в (2.8) та (2.9).

Первинні та вторинні ознаки досліджуваного процесу складають структурований словник ознак розпізнавання $\Sigma^{|N|}$, де $N = Card\,\Sigma^{|N|}$. Первинними ознаками виступають дані, отримані в результаті експерименту шляхом проведення опитування. Вторинні ознаки являють собою характеристики реалізацій $\left\{ x_{m,i}^{(j)} \mid i = \overline{1,N} \right\}$, навчальних вибірок $\left\{ x_{m,i}^{(j)} \mid j = \overline{1,n} \right\}$ або навчальної матриці.

Алфавіт класів $\{X_m^0 \mid m = \overline{1,M}\}$, формується відповідно до функціональних станів, які може приймати інформаційна система, та з урахуванням їх перетину в просторі ознак. Класи розпізнавання характеризують якість навчального контенту і відповідають оцінкам за національною шкалою: клас X_1^0 – відмінно (85-100), клас X_2^0 – добре (70-90), клас X_3^0 – задовільно (60-78) та клас X_4^0 – незадовільно (50-64).

Пошук мінімального обсягу навчальної вибірки n_{min} виконується з урахуванням наявності статистичної похибки $\mathcal{E}$ за допомогою методу динамічного довірчого інтервального оцінювання. Для цього здійснюється пошук довірчих інтервалів на кожному кроці випробувань, що дозволяє визначити ймовірність потрапляння i-ї ознаки в поле контрольних допусків з ймовірністю довіри рівній $1 - Q$, де Q – рівень значущості. Критерієм зупинки випробувань є стан, за якого довірчий інтервал накривається визначеним інтервалом $[0.5 \pm \Delta], |\Delta| < 0.5$. Таким чином, кінцевий перетин визначеного інтервалу з границею довірчого інтервалу задає значення n_{min}. Для даного дослідження задано обсяг навчальної вибірки, що дорівнює 40. Отже, в результаті проектування отримано навчальні матриці для кожного класу, де кількість ознак $\mathrm{N} = 40$, а кількість векторів-реалізацій $n = 40$.

Згідно з детерміновано-статистичним підходом, важливою задачею моделювання системи є формування системи нормованих (експлуатаційних) допусків, яка в свою чергу задає область значень контрольних допусків. Нормоване поле допусків $\delta_{H,i} \mid i = \overline{1,N}$ визначається як область, де i-та ознака знаходиться з ймовірністю $p_i = 0$ або $p_i = 1$. Область, в межах якої i-та ознака знаходиться з ймовірністю $0 < p_i < 1$, вважається контрольним полем допусків $\delta_{K,i} \mid i = \overline{1,N}$. При цьому стан інформаційної системи має відповідати базовому класу. Крім того, $|\delta_{K,i}| \leq |\delta_{H,i}|$ та базова система контрольних допусків не змінюється для усього алфавіту класів розпізнавання.

Введення контрольних допусків дозволяє провести рандомізацію процесу прийняття рішень шляхом використання одночасно детермінованих та статистичних характеристик, а також здійснити перетворення параметрів розподілу реалізацій в дискретному просторі у ході навчання системи. Границі системи контрольних допусків забезпечують не систематичність значень векторів-реалізацій, а також запобігають виникненню збігу еталонних векторів, які слугують центрами контейнерів класів.

Оптимізація системи нормованих допусків здійснюється за дистанційно-максимальним критерієм. Навчальна вибірка $\left\{x_1^{(j)} \mid j = \overline{1,n}\right\}$, що належить базовому класу $\{X_1^0\}$ поділяється на пари векторів-сусідів та проводиться пошук кодових відстаней за допомогою формули (3.1):

$$d_j = \sum_{i=1}^{N} \left[x_{1,i}^{(j)} \oplus x_{c,i}\right], j = \overline{1,n}, \quad (3.1)$$

де $x_{c,i}$ – i-та координата реалізації $\left\{x_{c,i} \mid i = \overline{1,N}\right\} \in X_c^0$, що є сусідом для реалізації $\left\{x_{1,i}^{(j)}\right\} \in X_1^0$.

Система допусків $\left\{\delta_{H,i}^*\right\}$ обирається за умови, що середня кодова відстань $\bar{d}_n = \frac{1}{n}\sum_{j=1}^{n} d_j$ до сусідніх реалізацій класу X_1^0 є найбільшою, тобто $\left\{\delta_{H,i}\right\} = \left\{\delta_{H,i}^*\right\}$, якщо $\bar{d}_n^* = max_{\{j\}}\bar{d}_j$. Отже, оптимізація поля допусків $\delta_{H,i}$ з центром у точці $y_{1,i} \in X_1^0$ полягає у виконанні послідовності кроків пошуку максимуму $\bar{d}_n^*$, на яких для поточної системи контрольних допусків здійснюється формування бінарних векторів $\left\{x_1^{(j)} \mid j = \overline{1,n}\right\}$ та пар векторів-сусідів за допомогою обчислення кодових відстаней між ними, і розраховується середня кодова відстань до векторів-сусідів. У результаті оптимізації отримано поля нормованих допусків $\left\{\delta_{H,i}^*\right\}$, що визначають допустиму область для контрольних допусків та гарантують випадковість ознак розпізнавання.

3.2 Визначення базового класу розпізнавання для заданого алфавіту

Ефективність функціонування системи в режимі навчання прямо залежить від заданої системи контрольних допусків, яка формується на основі базового касу. На практиці в якості базового прийнято вважати клас, що є найбільш бажаним для дослідника та особи, яка приймає рішення, тому існує необхідність дослідити функціональну ефективність системи в залежності від вибору базового класу.

Згідно принципів IEI-технології, визначення базового класу [15] відбувається на етапі оптимізації параметрів навчання в результаті пошуку максимального усереднення за алфавітом класів $\{X_m^0\}$ значень інформаційного критерію функціональної ефективності за виразом (3.2):

$$\bar{E}^* = \frac{1}{M}\sum_{m=1}^{M} max_{\{d\}}\, E_m, \qquad (3.2)$$

де E_m – інформаційний критерій функціональної ефективності системи розпізнавання реалізацій, що належать класу X_m^0;

$\{d\}$ – масив кроків процедури навчання.

Параметрами навчання системи виступають геометричні величини контейнерів класів та система контрольних допусків на ознаки розпізнавання. У цьому випадку алгоритм навчання інформаційної системи відповідає категорійній моделі, показаній на рисунку 2.2, та має наступний структурований вигляд:

$$\delta = arg\langle max_{G_\delta}\{max_{G_d}\bar{E}\}\rangle, \qquad (3.3)$$

де G_δ і G_d – області допустимих значень параметра, що задає величину поля контрольних допусків δ та радіусів контейнерів.

Крім того, відомо області допустимих значень параметрів:

1) $\delta \le \frac{\delta_H}{2}$, де δ_H – система нормованих допусків;

2) $d_m \in [0; d(x_m \oplus x_c) - 1]$, де $d(x_m \oplus x_c)$ – кодова відстань від еталонного вектора класу X_m^0 до вектора класу-сусіда X_c^0.

Значення ознак мають однакову розмірність, тому слід реалізувати алгоритм навчання із паралельною оптимізацією системи контрольних допусків та контейнерів класів за всіма ознаками одночасно. Вхідними даними є навчальна матриця, описана у розділі 3.1, областю значень кроку навчання δ обрано проміжок $\left[1; \frac{\delta_H}{2}\right]$. Тож, навчання системи виконується за алгоритмом:

1) ініціалізація лічильника, що задає крок δ зміни поля контрольних допусків: $l = l + 1$;

2) розрахунок значень нижнього та верхнього допусків: $\{\mathrm{A}_{HK,i}[l] = y_{1,i} - \delta[l]\}$ та $\{\mathrm{A}_{BK,i}[l] = y_{1,i} + \delta[l]\}$, $i = \overline{1,N}$, де $y_{1,i}$ – значення елементу навчальної вибірки для i-ї ознаки реалізації базового класу X_b^0;

3) знаходження бінарної матриці $\left\| x_{m,i}^{(j)} \right\|$ за правилом:

$$x_{m,i}^{(j)} = \begin{cases} 1, & if\ y_{m,i}^{(j)} \in \delta_{K,i} \\ 0, & if\ y_{m,i}^{(j)} \notin \delta_{K,i} \end{cases}; \tag{3.4}$$

4) знаходження еталонних бінарних векторів $\{x_{m,i} \mid m = \overline{1,M}, i = \overline{1,N}\}$ за правилом:

$$x_{m,i} = \begin{cases} 1, & if\ \frac{1}{n}\sum_{j=1}^{n} x_{m,i}^{(j)} > \rho_m \\ 0, & if\ else \end{cases}, \tag{3.5}$$

де ρ_m – рівень селекції.

5) формування масиву пар найближчих сусідніх класів: $\Re_m^{|2|} = \langle x_m, x_c \rangle$, де x_c – вектор класу-сусіда X_c^0;

6) пошук оптимальної кодової відстані d_m, враховуючи, що $\mathrm{E}_m(0) = 0$;

7) обчислення та пошук максимального значення критерію функціональної ефективності в межах робочої області: $E_m^* = max_{\{d\}} E_m$, де $\{d\} = \{0,1, \dots, d < d(x_m \oplus x_l)\}$ – масив радіусів контейнерів з вершиною $x_m \in X_m^0$. В якості критерія оптимізації виступають статистичні інформаційні міри, наприклад, ентропійна міра за Шенноном або інформаційна міра Кульбака, що представлені у (2.13) та (2.15) відповідно;

8) якщо $E_1^*[l] \geq E_1^*[l-1]$, то виконується перехід до пункту 9, інакше – до пункту 10;

9) якщо $\delta \leq \frac{\delta_H}{2}$, то виконується перехід до пункту 1, інакше – 10;

10) вибір оптимальних параметрів: $\{A_{HK,i}^* = A_{HK,i}[l-1]\}$; $\{A_{BK,i}^* = A_{BK,i}[l-1]\}, i = \overline{1,N}$; $E_1^* = E_1^*[l-1]$.

Алгоритм вибору базового класу здійснюється за допомогою наступної послідовності кроків:

1) ініціалізація лічильника класів розпізнавання: $m = m + 1$;

2) виконується оптимізація системи контрольних допусків за описаним вище алгоритмом;

3) пошук найбільшого усередненого значення критерію функціональної ефективності в межах робочої області;

4) якщо $m + 1 \leq M$, то виконується перехід до пункту 1, інакше – до пункту 5;

5) вибір базового класу шляхом пошуку найбільшого усередненого значення критерію функціональної ефективності серед класів розпізнавання:

$$m^* = arg\left(max_{m \in M}(\bar{E}^*)\right). \quad (3.6)$$

Наведений алгоритм виконується в процесі оптимізації кожного параметру функціонування системи до тих пір, доки критерій не набуде максимально граничного значення, забезпечуючи при цьому отримання безпомилкових вирішальних правил.

У ході виконання алгоритму здійснено оптимізацію системи контрольних допусків за паралельним алгоритмом, використовуючи міру Кульбака у якості критерію функціональної ефективності. Аналіз отриманих результатів показав, що класи X_1^0 та X_4^0 не можуть використовуватись як базовий, адже не мають робочої області визначення для усередненого значення критерію. Тому розглянемо результат оптимізації параметрів при виборі класів X_2^0 та X_3^0 за базовий.

У разі класу X_2^0 у якості базового, знайдено оптимальне значення параметра, що визначає величину поля контрольних допусків $\delta^* = 20$. Також визначено оптимальні радіуси контейнерів класів розпізнавання X_1^0, X_2^0, X_3^0, X_4^0, вони мають наступні значення: $d_1^* = 5,\ d_2^* = 1,\ d_3^* = 11,\ d_4^* = 10$ відповідно. Значення максимального усередненого критерію функціональної ефективності $\bar{E}^* = 2.61436$, при цьому максимальні значення критеріїв для класів розпізнавання: $E_1^* = 1.177,\ E_2^* = 3.17,\ E_3^* = 2.941,\ E_4^* = 3.17$.

Результат оптимізації параметрів навчання зображено у вигляді графіку на рисунку 3.1:

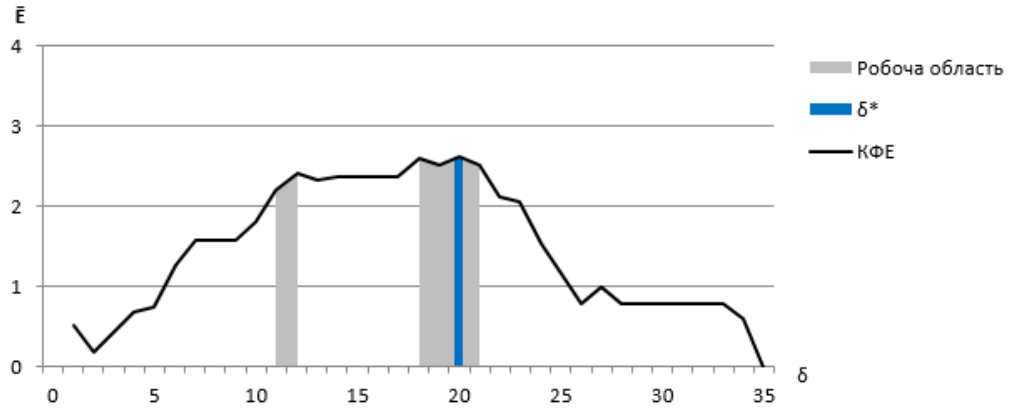


Рисунок 3.1 – Графік залежності КФЕ від параметра поля допусків δ при базовому класі X_2^0

При виборі базового класу X_3^0, оптимальне значення параметра системи контрольних допусків $\delta^* = 14$. Визначено оптимальні радіуси контейнерів класів, що мають значення $d_1^* = 4$, $d_2^* = 15$, $d_3^* = 4$, $d_4^* = 13$. Значення максимального усередненого критерію $\bar{E}^* = 3.06013$, при цьому максимальні значення критеріїв для класів розпізнавання: $E_1^* = E_2^* = E_3^* = 3.17$, $E_4^* = 2.731$. Результат оптимізації відображено на графіку 3.2:

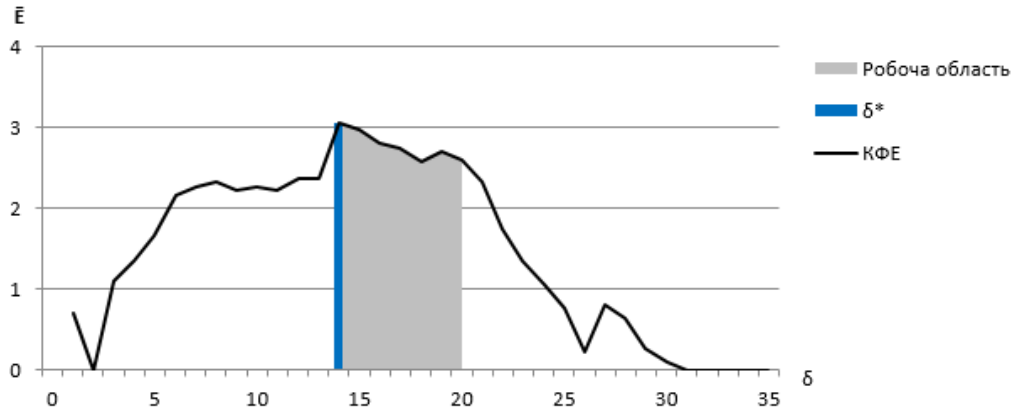


Рисунок 3.2 – Графік залежності КФЕ від параметра поля допусків δ при базовому класі X_3^0

Отже, у результаті порівняльного аналізу отриманих значень можна зробити висновок, що максимальне значення критерію функціональної ефективності отримано за умови, що обрано базовий клас X_3^0.

3.3 Опис алгоритму функціонування інформаційно-аналітичної системи на етапі екзамену

Нехай задано вхідні параметри для алгоритму екзамену:

- M – кількість класів розпізнавання;
- $\{X_m^* \mid m = \overline{1,M}\}$ – еталонні вектори-реалізації образу, що задають центри контейнерів для кожного класу;
- $\{d_m^*\}$ – оптимальні радіуси для побудови контейнерів;
- $\{x^{(j)} \mid j = \overline{1,n}\}$ – бінарні вектори-реалізації образу, які розпізнаються;
- $\{\delta_{k,i}^* \mid i = \overline{1,N}\}$ – оптимізована система контрольних допусків.

Тоді, алгоритм функціонування системи в режимі моніторингу за IEI-технологією має наступну послідовність кроків:

1) ініціалізація лічильника класів розпізнавання: $m = m + 1$;
2) ініціалізація лічильника реалізацій, які треба розпізнати: $j = j + 1$;
3) обчислення кодової відстані (відстані Хеммінга) між екзаменаційним вектором та еталонним вектором поточного класу $d(x_m^* \oplus x^{(j)})$;
4) розрахунок функції належності до класу розпізнавання за формулою:

$$\mu_m = 1 - \frac{d(x_m^* \oplus x^{(j)})}{d_m^*} \qquad (3.7)$$

5) перевірка: якщо $j \leq n$, то виконується перехід до пункту 2, інакше – до пункту 6;
6) перевірка: якщо $m \leq M$, то виконується перехід до 1, інакше – 7;
7) знаходження класу, до якого належить реалізація, визначається за допомогою умови : $\bar{\mu}_m^* = max_{\{m\}}\bar{\mu}_m$, де $\bar{\mu}_m = \frac{1}{n}\sum_{j=1}^{n}\mu_{m,j}$ – середнє значення функцій належності реалізацій класу X_m^0. Якщо $\bar{\mu}_m^* \leq \mathrm{c}$, де c – порогове значення, то відповідь – клас не визначено.

Отже, згідно положень IEI-технології, алгоритм екзамену носить детермінований характер і не має громіздких обчислень, що дозволяє приймати рішення за простим обумовленим вирішальним правилом та здійснювати моніторинг у реальному часі.

3.4 Короткий опис програмного забезпечення

У результаті розробки інформаційно-аналітичної системи «Educational Monitoring» створено механізми функціонування системи в режимах навчання та екзамену. Програмна реалізація виконана за допомогою об’єктно-орієнтованої мови програмування C#. Інформаційно-аналітична система «Educational Monitoring» працює у двох інтерфейсах.

Інтерфейс користувача дозволяє пройти опутування випускникам кафедри щодо якості навчального матеріалу та відповідності його до вимог ринку праці, крім того, отримати оцінку навчального контенту за накопиченими результатами опитування. Головна сторінка системи відображена на рисунку 3.1:

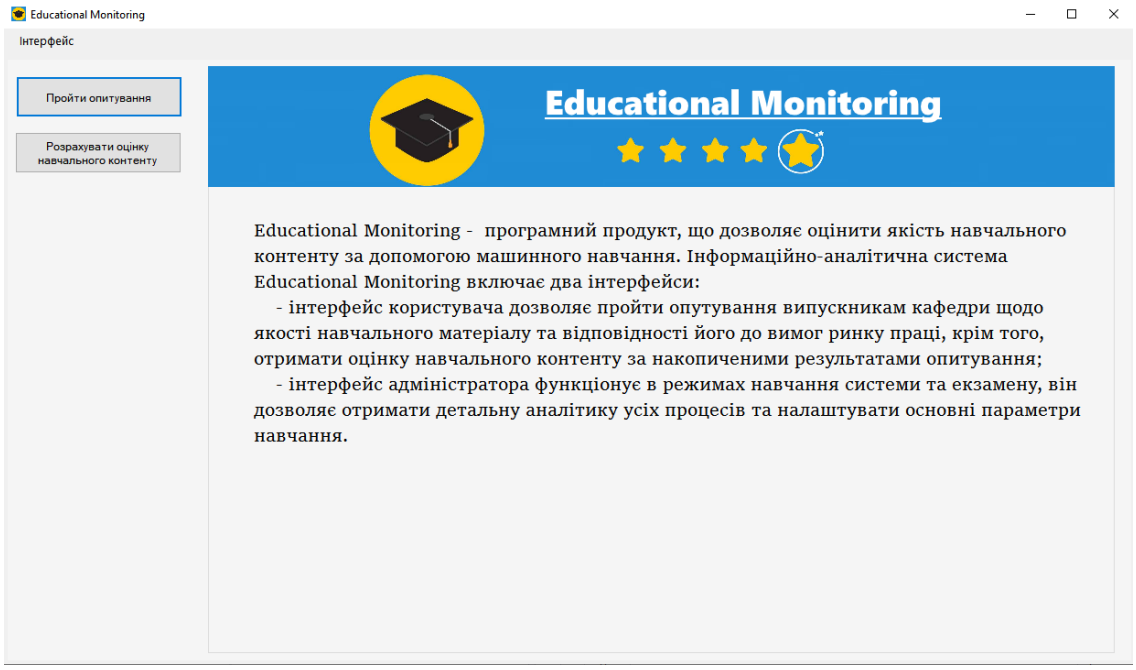


Рисунок 3.3 – Головна сторінка інформаційно-аналітичної системи «Educational Monitoring»

Команда «Пройти опитування» відкриває форму опитування, що містить блоки із заголовками дисциплін та пункти, що оцінюються (не більше 10-ти на кожний блок). Дані для опитування зберігаються в файлі програми в форматі JSON, тому він легко може доповнюватись новими блоками та пунктами. Кожен пункт можна оцінити, пересунувши відповідний повзунок до

необхідного положення (від 0 до 100). На рисунку 3.4 показано приклад однієї з сторінок опитування:

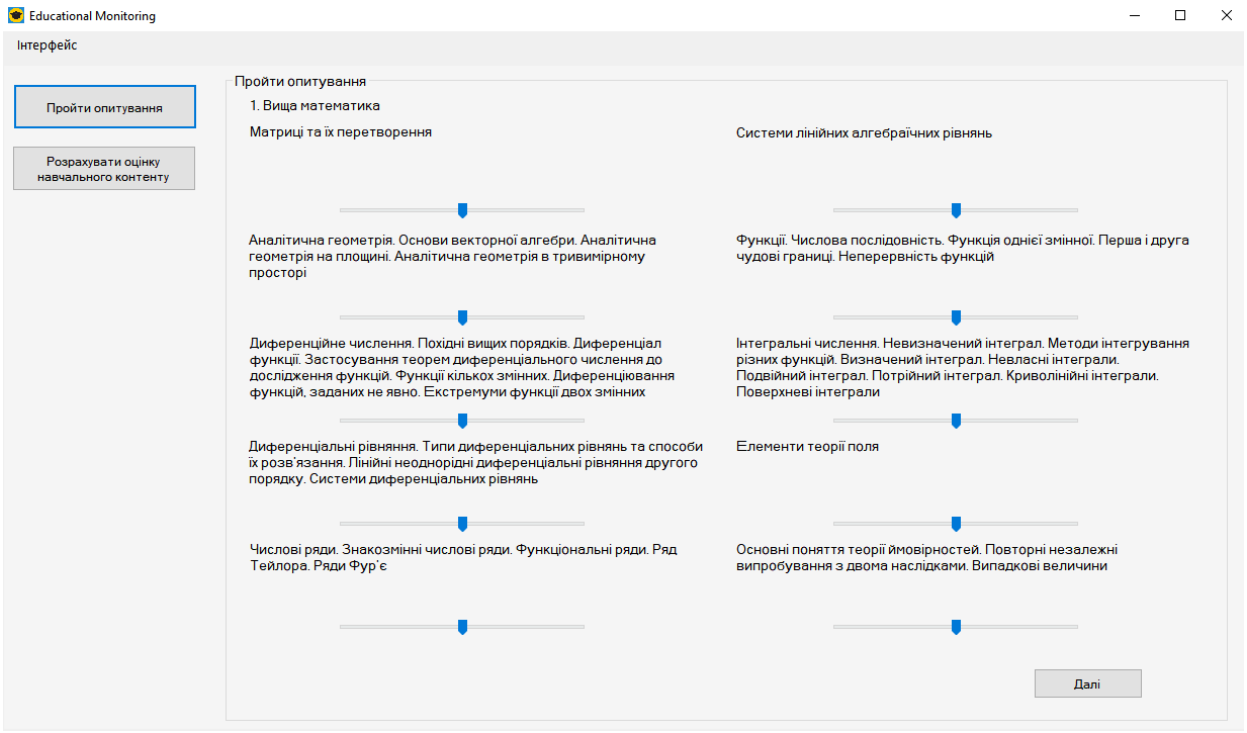


Рисунок 3.4 – Форма для опитування студентів

Перехід на наступну сторінку здійснюється за допомогою команди «Далі». На останньому блоці за допомогою команди «Завершити» здійснюється збереження результатів опитування, які будуть використані для побудови екзаменаційної матриці. Команда «Розрахувати оцінку навчального контенту» дозволяє отримати оцінку навчального контенту за накопиченими результатами опитування шляхом виконання алгоритмів навчання та екзамену у фоновому режимі. У результаті користувач отримає лише результат оцінювання за національною шкалою.

Інтерфейс адміністратора функціонує в режимах навчання системи та екзамену, він дозволяє налаштувати основні параметри навчання, отримати детальну аналітику усіх процесів та візуалізувати результати оптимізації параметрів навчання. Параметри функціонування системи, такі як, базовий клас, інформаційний критерій та початкове значення кроку δ, що задає

величину поля контрольних допусків, задані за замовчуванням та доступні для редагування.

Команда «Почати навчання» здійснює виконання алгоритму машинного навчання системи за IEI-технологією. Результати навчання для заданого та оптимального значення δ по закінченню відображаються у відповідному блоці. Команда «Екзамен» запускає виконання алгоритму екзамену за результатами навчання і відображає результат у блоці «Результат екзамену». Сторінку для роботи в режимі адміністратора показано на рисунку 3.5:

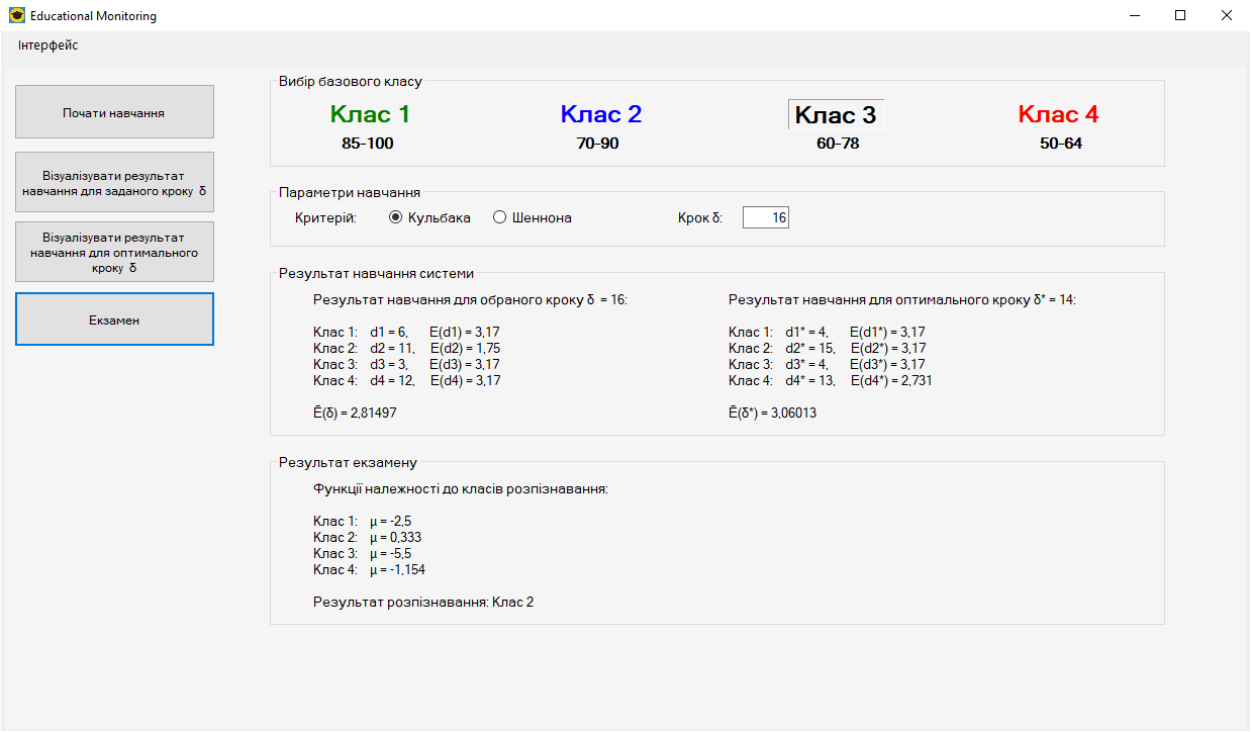


Рисунок 3.5 – Робота в режимі адміністратора

Команди «Візуалізувати результат навчання для заданого кроку δ» та «Візуалізувати результат навчання для оптимального кроку δ» дозволяють побудувати графіки залежності критерію функціональної ефективності від радіуса для кожного класу розпізнавання, а також відобразити залежність усередненого значення критерію від параметру δ. Побудова графіків здійснюється за результатами розрахунків на етапі навчання в залежності від обраного критерію. Команда «Зберегти в Excel» дозволяє експортувати дані для побудови графіків в файл типу Excel XLS з метою їх подальшого

використання та форматування. Приклад сторінки, що відображає результати навчання системи у вигляді графіків наведено на рисунку 3.6:

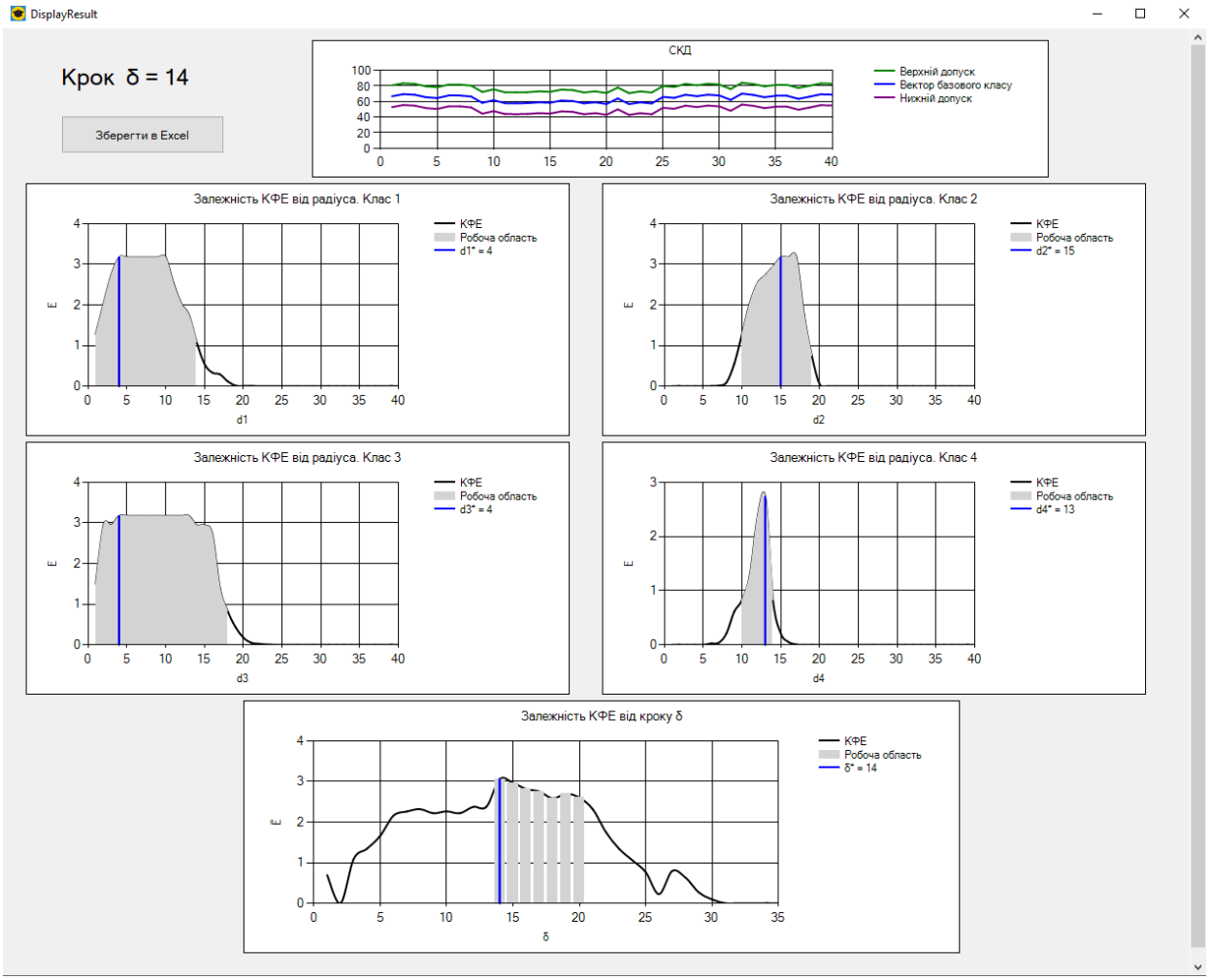


Рисунок 3.6 – Візуалізація результатів машинного навчання системи

Для функціонування основних модулів системи розроблено наступні класи для роботи з даними:

- EducationalMonitorig.cs – основний клас, що реалізує роботу на формі та запуск основних процесів системи;
- UsingFiles.cs – клас, що відповідає за роботу з файлами;
- PollData.cs – клас, що описує структуру опитування;
- InputDescription.cs – клас, який здійснює формування вхідного математичного опису та екзаменаційної матриці;
- BasicAlgorithm.cs – клас, що реалізує алгоритм навчання системи;
- ExamAlgorithm.cs – клас, що реалізує алгоритм екзамену;

- DisplayResults.cs – клас, який здійснює відображення та збереження результатів навчання системи у вигляді графіків.

У таблиці 3.1 наведені основні змінні, що задіяні у реалізації етапу екзамену системи:

Таблиця 3.1 – Основні змінні етапу екзамену

Назва	Тип	Опис
inputDescription	InputDescription	Містить вхідний математичний опис, що включає навчальні матриці для класів розпізнавання та екзаменаційну матрицю.
basicAlgorithm	BasicAlgorithm	Містить результати отримані на етапі навчання системи.
binMartix	double[,]	Екзаменаційна матриця, що формується на основі результатів опитування.
binVector	int[]	Двійковий екзаменаційний вектор.
codeDistances	int[]	Кодові відстані між екзаменаційним вектором та еталонними векторами класів розпізнавання.
function Membership Kulbak	double[]	Значення функції належності, розраховані для кожного класу за інформаційним критерієм Кульбака.
function Membership Shannon	double[]	Значення функції належності, розраховані для кожного класу за інформаційним критерієм Шеннона.
Recognized ClassKulbak	int	Клас, що був розпізнаний у результаті екзамену за інформаційним критерієм Кульбака.
Recognized ClassShannon	int	Клас, що був розпізнаний у результаті екзамену за інформаційним критерієм Шеннона.

У таблиці 3.2 наведені методи, що виконують основні процедури на етапі екзамену системи:

Таблиця 3.2 – Основні методи етапу екзамену

Назва	Опис
startExam	Головний метод класу ExamAlgorithm, запускає виконання алгоритму екзамену.
createBinMatrix	Здійснює формування екзаменаційної бінарної матриці.
createBinVector	Будує еталонний вектор на основі екзаменаційної матриці.
vectorsXOR	Розраховує кодові відстані (відстань Хеммінга) від екзаменаційного вектора до векторів класів розпізнавання.
findFunction Accessories	Розраховує функцію належності до класів розпізнавання.
findRecognized Class	Порівнює результати та знаходить клас, до якого належить образ.

В проект інформаційної системи також підключено додаткові бібліотеки та пакети для роботи з різними типами даних:

- Math.Net.Numerics – бібліотека, що включає методи та алгоритми для виконання обчислень, які виникають у науці, техніці та щоденному використанні. Вона дозволяє працювати з спеціальними математичними функціями, використовувати лінійну алгебру, імовірнісні моделі, випадкові числа, інтерполяцію, регресію, оптимізацію, інтегральні перетворення та інші.
- Newtonsoft.Json – пакет, що забезпечує механізм парсингу об’єктів JSON з динамічною структурою. Такі об’єкти приводяться до типу простору імен Newtonsoft.Json.Ling, а потім у структуру даних мови C#.
- Microsoft.Office.Interop.Excel – бібіліотека, що надає можливість працювати з файлами типу Excel XLS, здійснює читання та запис даних у електронні таблиці за допомогою об’єктів Application, Workbook, Worksheet.

Програмний код основних класів та методів наведено у додатку А.

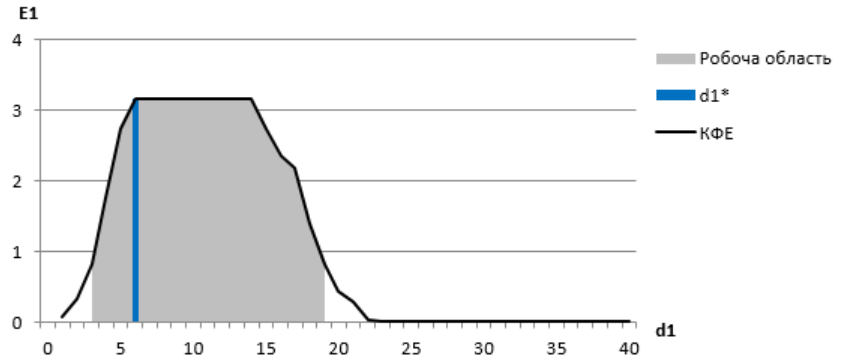
3.5 Результати фізичного моделювання

В процесі розробки інформаційного забезпечення системи «Educational Monitoring» виконано побудову вхідного математичного опису, що передбачає розпізнавання чотирьох класів, які характеризують оцінку якості навчального контенту: клас X_1^0 – відмінно (85-100), клас X_2^0 – добре (70-90), клас X_3^0 – задовільно (60-78) та клас X_4^0 – незадовільно (50-64). Для рівня селекції координат еталонних векторів за замовчуванням взято значення 0.5 за усіма класами та ознаками. Поля контрольних допусків визначаються як половина відповідних полів нормованих допусків, тому областю значень параметру δ є проміжок $\left[1; \frac{\delta_H}{2}\right]$.

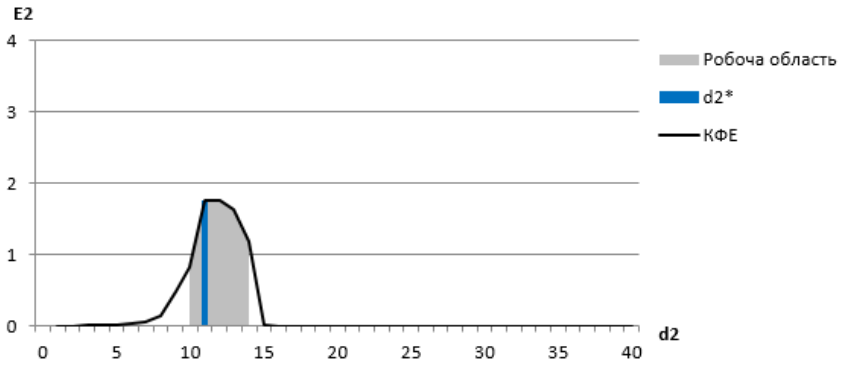
З метою максимізації функціональної ефективності системи було визначено, що базовий клас X_3^0 забезпечує найбільше значення інформаційного критерію (див. розділ 3.2). Тому для отримання оптимальних параметрів функціонування інформаційно-аналітичної системи реалізовано алгоритм машинного навчання з паралельною оптимізацією системи контрольних допусків та радіусів контейнерів класів розпізнавання. Аналіз ефективності інформаційної системи здійснено за обома критеріями: ентропійною мірою Шеннона та інформаційною мірою Кульбака. Початковим значенням параметру, що визначає величину поля контрольних допусків обрано $\delta = \pm 16$.

Аналіз отриманих даних показує, що для кожного класу існує робоча область та знайдено оптимальні значення радіусів для заданого кроку: $d_1 = 6$, $d_2 = 11$, $d_3 = 3$, $d_4 = 12$. При цьому, за критерієм Кульбака їм відповідають наступні максимальні значення КФЕ: $E_1 = 3.17$, $E_2 = 1.75$, $E_3 = E_4 = 3.17$, а значення найбільшого усередненого критерію $\bar{E} = 2.81497$. На рисунку 3.7 показано графіки залежності КФЕ від радіусів геометричних контейнерів для кожного з класів за критерієм Кульбака:

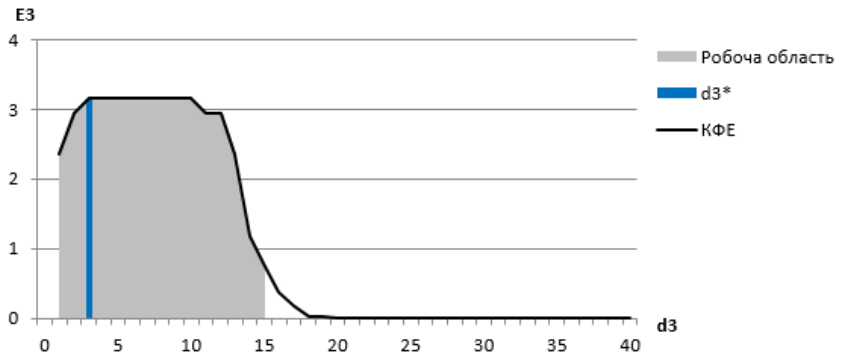
а)



б)



в)



г)

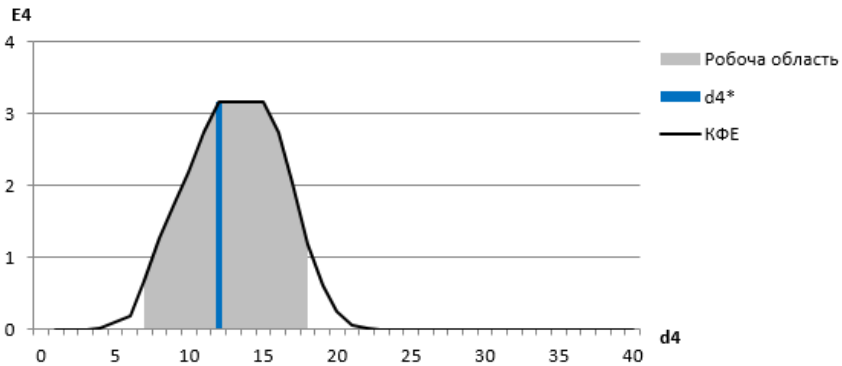
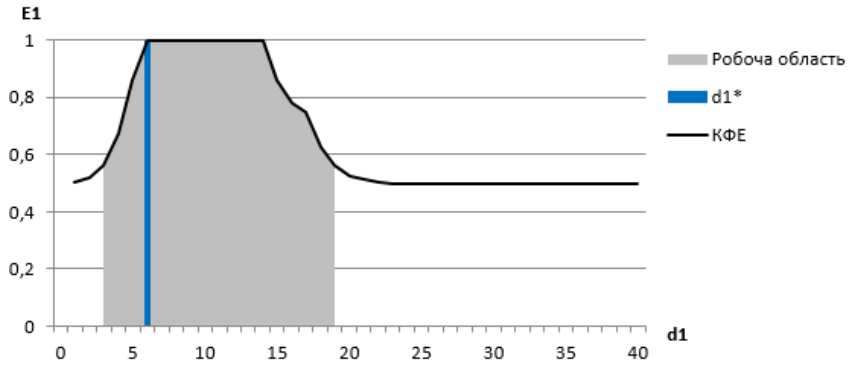


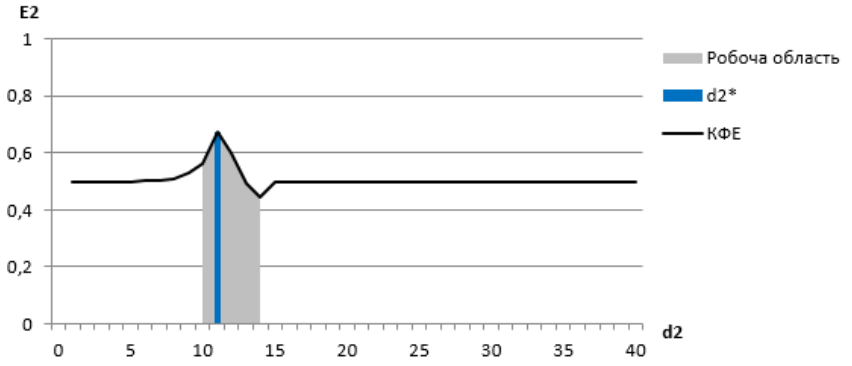
Рисунок 3.7 – Графіки залежності КФЕ від радіусів класів за критерієм Кульбака: а – клас X_1^0; б – клас X_2^0; в – клас X_3^0; г – клас X_4^0

За критерієм Шеннона радіуси класів мають ті самі значення, при цьому оптимальні значення КФЕ дорівнюють: $E_1 = 1,\ E_2 = 0.675,\ E_3 = E_4 = 1,$ $\bar{E} = 0.91875$. На рисунку 3.8 показано графіки, побудовані за цим критерієм:

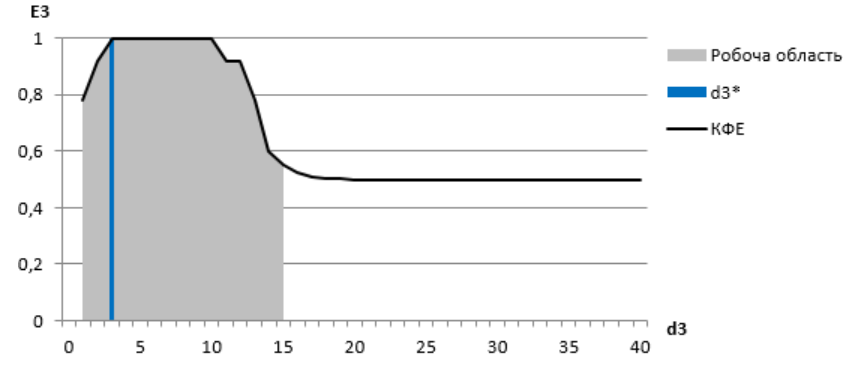
а)



б)



в)



г)

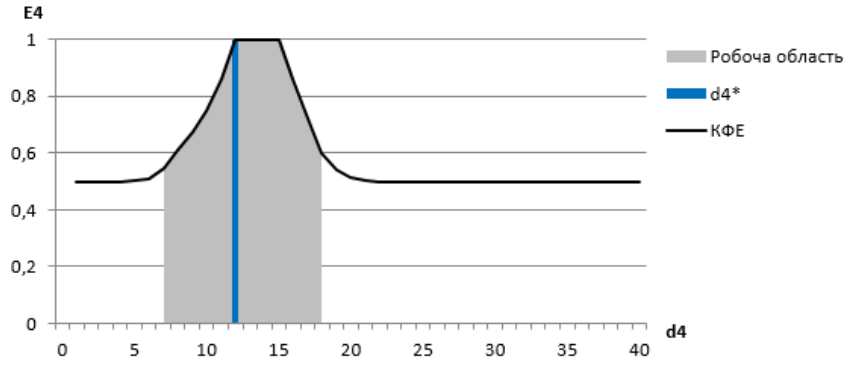


Рисунок 3.8 – Графіки залежності КФЕ від радіусів класів за критерієм Шеннона: а – клас X_1^0; б – клас X_2^0; в – клас X_3^0; г – клас X_4^0

Наведені графіки показують, що за обома критеріями спостерігаються не високі максимальні значення для класу X_2^0. Отже, система контрольних допусків на ознаки розпізнавання потребує оптимізації.

Пошук оптимальних значень параметрів полягає у проведенні ітераційної процедури навчання системи з метою пошуку максимального усередненого значення критерію. Тож, у результаті виконання алгоритму оптимізації отримано масив значень КФЕ, які залежать від параметра поля контрольних допусків δ, та за критерієм Кульбака має вигляд графіку, показаного на рисунку 3.9:

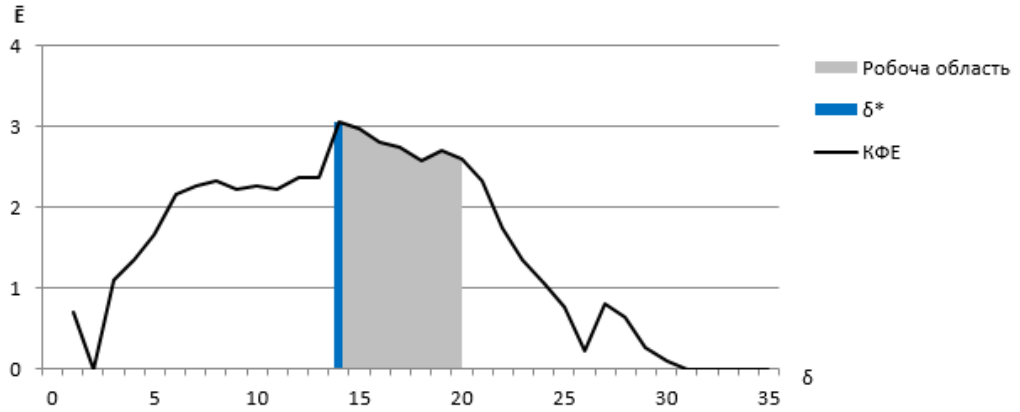


Рисунок 3.9 – Графік залежності КФЕ від параметра поля допусків δ за критерієм Кульбака

Аналогічним чином проведено розрахунок критерію за мірою Шеннона, графік його залежності від параметру δ показано на рисунку 3.10:

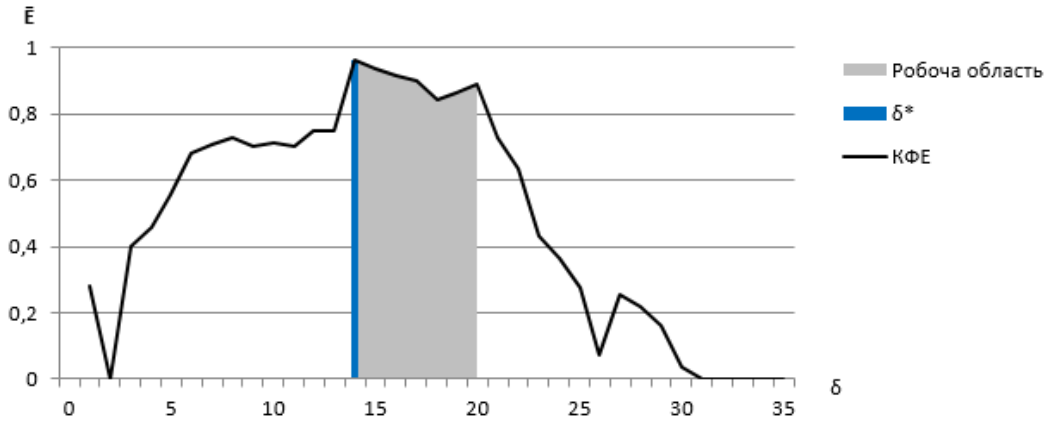
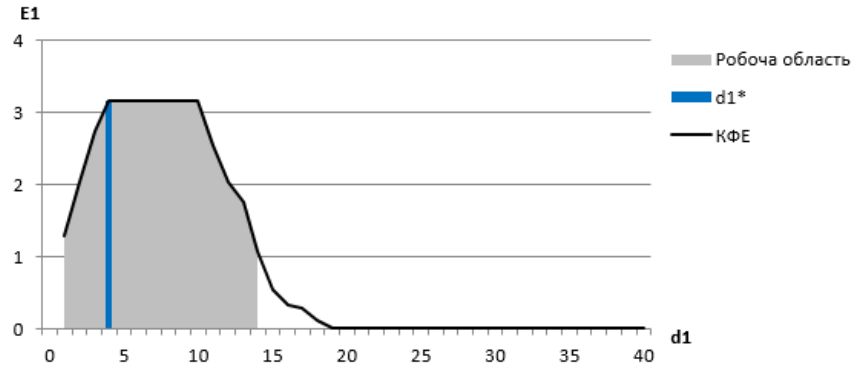


Рисунок 3.10 – Графік залежності КФЕ від параметра поля допусків δ за критерієм Шеннона

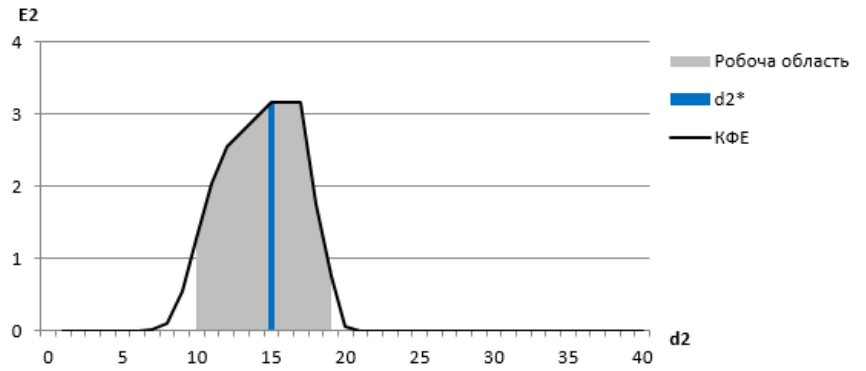
В процесі аналізу визначено оптимальне значення параметру поля контрольних допусків, що за обома критеріями має значення $\delta^* = \pm 14$, та забезпечує найбільше значення усередненого критерію $\bar{E}^* = 3.06013$ – за

критерієм Кульбака та $\bar{E}^* = 0.96548$ – за критерієм Шеннона. Для отриманого значення кроку δ побудовано графіки, що показують залежність КФЕ від радіусів класів за критерієм Кульбака (див. рисунок 3.11):

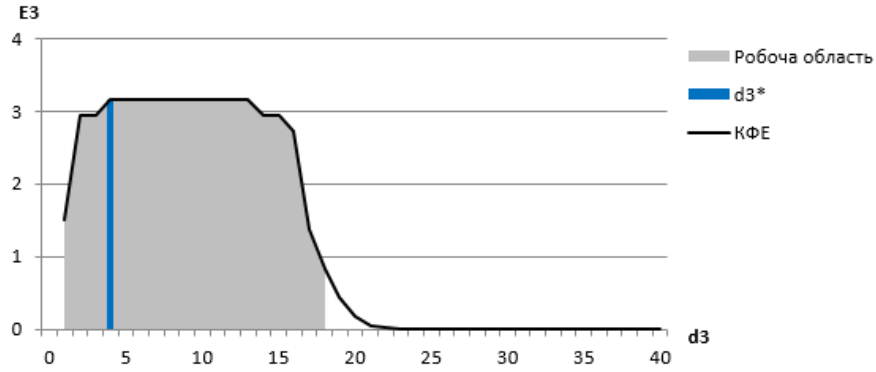
а)



б)



в)



г)

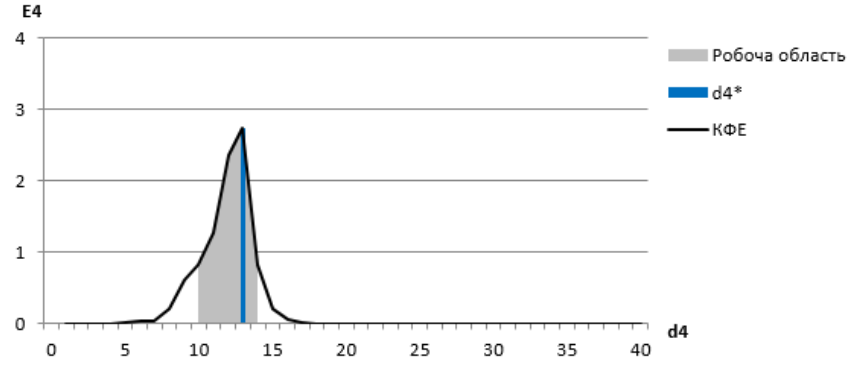
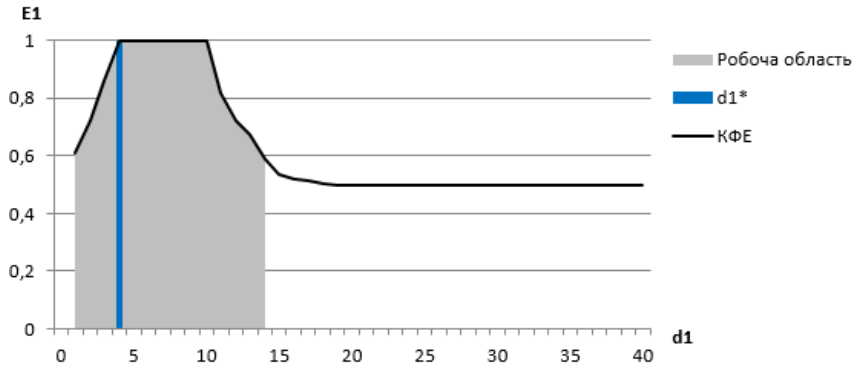


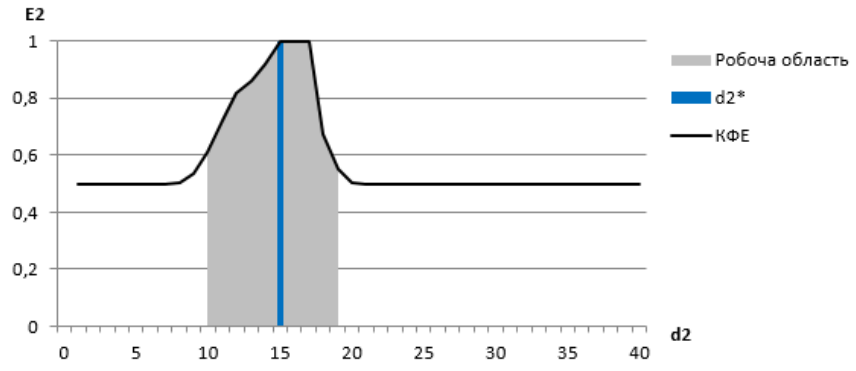
Рисунок 3.11 – Графіки залежності КФЕ від радіусів класів для оптимального δ за критерієм Кульбака: а – клас X_1^0; б – клас X_2^0; в – клас X_3^0; г – клас X_4^0

Подібного вигляду набувають і графіки залежності КФЕ від радіусів геометричних контейнерів для кожного з класів за критерієм Шеннона, що показані на рисунку 3.12:

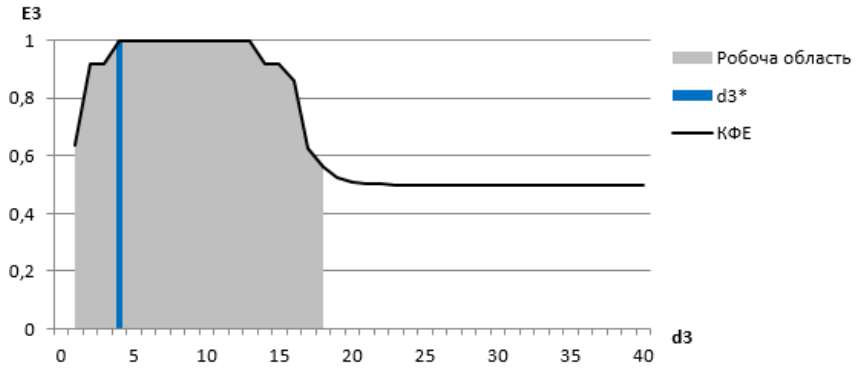
а)



б)



в)



г)

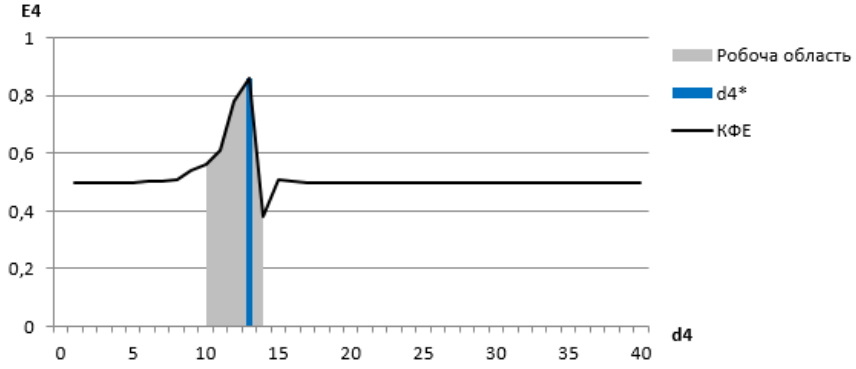


Рисунок 3.12 – Графіки залежності КФЕ від радіусів для оптимального δ за критерієм Шеннона: а – клас X_1^0; б – клас X_2^0; в – клас X_3^0; г – клас X_4^0

Для побудови безпомилкових вирішальних правил треба також визначити оптимальні значення геометричних параметрів контейнерів для кожного класу. У результаті аналізу даних, одержаних при оптимальному параметрі контрольних допусків, визначено, що радіуси контейнерів класів мають значення $d_1^* = 4,\ d_2^* = 15,\ d_3^* = 4,\ d_4^* = 13$ за обома критеріями. При цьому максимальні значення для класів розпізнавання за критерієм Кульбака дорівнюють $E_1^* = E_2^* = E_3^* = 3.17,\ E_4^* = 2.731$ та за критерієм Шеннона – $E_1^* = E_2^* = E_3^* = 1,\ E_4^* = 0.862.$

На етапі екзамену виконано два незалежних експерименти, для цього сформовано екзаменаційні матриці на основі даних, що були накопичені під час опитування. У таблиці 3.3 наведено кількісне співвідношення входження значень екзаменаційної матриці до діапазонів класів розпізнавання.

Таблиця 3.3 – Кількість входжень значень матриць до діапазонів класів

Номер експерименту	Клас 1 (85-100)	Клас 2 (70-90)	Клас 3 (60-78)	Клас 4 (0-64)
1	855	280	631	346
2	307	384	874	674

У ході виконання алгоритму екзамену розраховано функції належності та здійснена класифікація, результати експериментів занесено у таблицю 3.4:

Таблиця 3.4 – Результати розпізнавання класу

Номер експерименту	Значення функції належності μ				Результат оцінювання
	Клас 1	Клас 2	Клас 3	Клас 4	
1	1	-0.333	-9	-0.077	Відмінно
2	-8.5	-0.2	0.5	-1.154	Задовільно

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

ВИСНОВКИ

У ході виконання роботи здійснено аналіз сучасного стану дослідження в області оцінювання навчальної діяльності, розглянуто основні засоби оцінки і актуалізації навчальних планів та програм, виявлено наукову проблему, яка полягає у розробці алгоритмів функціонування інформаційно-аналітичної системи адаптації навчального контенту до вимог ринку праці.

В процесі пошуку методів інтелектуального аналізу даних та машинного навчання, які функціонують за умови апріорної невизначеності, здійснено вибір та обґрунтування методології дослідження в межах інформаційно-екстремальної технології, яка базується на максимізації функціональної ефективності системи підтримки прийняття рішень, здатної навчатися. На етапі інформаційного проектування системи розглянуто категорійні моделі машинного навчання, які дозволяють створити безпомилкові вирішальні правила у результаті навчання; здійснено вибір інформаційних критеріїв оптимізації навчання; проведено формування вхідного математичного опису та визначення базового класу розпізнавання.

В результаті розробки інформаційно-аналітичної системи «Educational Monitoring» реалізовано програмне забезпечення за допомогою об’єктно-орієнтованої мови C#. Програма дозволяє провести опитування випускників кафедри та оцінити якість навчального контенту за алгоритмами машинного навчання з оптимізацією контрольних допусків та параметрів контейнерів класів і екзамену. Для роботи з системою розроблено інтерфейс, який дає змогу користувачеві налаштувати основні параметри навчання, отримати детальну аналітику усіх процесів та візуалізувати результати оптимізації параметрів навчання.

На етапі навчання було визначено оптимальні параметри функціонування системи та проведено тестування її роботи за допомогою двох експериментів, що полягають у проведенні етапу моніторингу за сформованими наперед екзаменаційними матрицями.

СПИСОК ЛІТЕРАТУРИ

1. Свіжевська С. А. Розвиток системи акредитації у вищій освіті України (кінець XX - початок XXI ст.) [Текст] : автореф. дис. на здоб. наук. ступ. канд. пед. наук : 13.00.01 - загальна педагогіка та історія педагогіка / Свіжевська Світлана Андріївна ; Ін-т вищої освіти НАПН України. – Київ, 2015. – 20 с.

2. Мосьпан Н.В. Вища освіта України в умовах розвитку ринкової економіки [Електронний ресурс] / Н.В. Мосьпан // Педагогічний процес: теорія і практика. – 2013. – Вип. 4. – С. 145-153. – Режим доступу : http://nbuv.gov.ua/j-pdf/pptp_2013_4_17.pdf.

3. Забезпечення якості вищої освіти в Європі: досвід Ірландії [Текст] : навч.-метод. рек. для студентів пед. спец. / Київ. нац. ун-т ім. Тараса Шевченка; [уклад.] Ю. Г. Запорожченко. - Київ : Принт-центр, 2010. - 74 с.

4. Єсіна О. Г. Критерії оцінки якості підготовки сучасних фахівців / О. Г. Єсіна // Теорія та методика навчання фундаментальних дисциплін у вищій школі: збірник наукових праць. – Кривий Ріг: НМетАУ, 2012. – Вип. VII. – С. 84-90. – Режим доступу: http://dspace.oneu.edu.ua/jspui/handle/123456789/1401.

5. Quality procedures in European Higher Education: ENQA survey (ENQA Occasional Papers 5). – Helsinki: European Network for Quality Assurance in Higher Education, 2003. – 41 p.

6. Oliveira, P.C.d., C.J.C.d.A. Cunha and M.K. Nakayama, 2016. Learning management systems (lms) and e-learning management: An integrative review and research agenda. JISTEM-Journal of Information Systems and Technology Management, 13(2): 157-180.Available at: https://doi.org/10.4301/s1807-177520160002000001.

7. Musca, G., Mihalache, A., & Musca, E. (2016, November). E-learning implementation in the superior technical educational system. In IOP Conference Series: Materials Science and Engineering (Vol. 161, No. 1, p. 012110). IOP Publishing. https://doi.org/10.1088/1757-899X/161/1/012110.

8. Bharati M., Ramageri.: Data Mining Techniques and Applications. In Indian Journal of Computer Science and Engineering, Vol. 1 No. 4 301-305, Dec 4, 2010.

9. Інтелектуальний аналіз даних / О. М. Колодчак // Вісник Національного університету "Львівська політехніка". Комп'ютерні системи та мережі. - 2013. - № 773. - С. 49-58. - Режим доступу: http://nbuv.gov.ua/UJRN/ VNULPKSM_2013_773_11

10. Довбиш, А.С. Основи проектування інтелектуальних систем [Текст] : навч. посіб. / А.С. Довбиш. - Суми : СумДУ, 2009. - 170 с.

11. Краснопоясовський А.С., Черниш А.В., Сластушевський О.Ю. Про вибір критерію функціональної ефективності системи розпізнавання // Радиоэлектроника и информатика.– 2001.- №4.-С. 121-124.

12. Довбиш А.С. Оптимізація ієрархічної структури даних інтелектуальної системи функціонального діагностування технічного стану складної машини/ А.С. Довбиш, В. І. Зимовець, М. В. Бібик // Вісник НТУ «ХПІ». Серія: Системний аналіз, управління та інформаційні технології. – Харків : НТУ «ХПІ», 2018. – № 48 . – С. 45-57.

13. Москаленко, В. В. Вступ до інформаційного аналізу і синтезу інфокомунікаційних систем [Текст] : навч. посіб. / В. В. Москаленко, А. С. Довбиш. – Суми : СумДУ, 2016. – 226 с. – 80-85.

14. Довбиш, А.С. Основи теорії розпізнавання образів [Текст]: навч. посіб.: у 2-х ч. Ч.1 / А.С. Довбиш, І.В. Шелехов. - Суми: СумДУ, 2015. - 109 с.

15. 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.

ДОДАТОК А

Опис програмного класу EducationalMonitorig.cs:

```
public partial class EducationalMonitorig : System.Windows.Forms.Form
{
    public System.Windows.Forms.ToolStripMenuItem currInterfase;
    public System.Windows.Forms.RadioButton currCriterion;
    public int currPage;
    public int currQuestion;

    public int volTrainingSample;
    public int baseClass;
    public int maxDelta;

    public double[] pollResult;
    double[] EAverageKulbak;
    double[] EAverageShannon;
    bool[] workArea;

    BasicAlgorithm[] baseAlgorithm;
    InputDescription inputDescription;
    PollData pollData;
    UsingFiles usingFiles;
    DisplayResults displayResults;
    ExamAlgorithm examAlgorithm;

    public EducationalMonitorig()
    {
        InitializeComponent();

        currInterfase     = interfaceUser;
        currCriterion     = KulbaksCriterion;
        currPage          = 0;
        currQuestion      = 0;
        volTrainingSample = 40;
        baseClass         = 3;
        maxDelta          = 35;

        usingFiles        = new UsingFiles();
        inputDescription  = new InputDescription();
    }

    private void interface_Click(object sender, EventArgs e)
    {
        currInterfase = (ToolStripMenuItem)sender;

        interfaceUser.Checked     = (currInterfase == interfaceUser) ? true : false;
        interfaceAdmin.Checked    = (currInterfase == interfaceAdmin) ? true : false;
        tabInterfaces.SelectedTab = (sender == interfaceUser) ? pageUser : pageAdmin;
    }

    private void buttonPoll_Click(object sender, EventArgs e)
    {
        textBoxResult.Visible = false;

        buttonNext.Text = "Далі";
        currPage        = 0;
        currQuestion    = 0;

        pollData = usingFiles.readFileJSON("pollData", ".json");

        if ((pollData.disciplines == null) || (pollData.disciplines.Length == 0))
```

```
                return;

            pollResult = new Double[pollData.getCountOfQuestions()];

            setDataForCurrentPage();

            groupBoxPoll.Visible = true;
        }

        public void setDataForCurrentPage()
        {
            if (currPage < pollData.disciplines.Length)
            {
                Discipline discipline = pollData.disciplines[currPage];

                textBoxDiscipline.Text = discipline.title;

                int i;
                for (i = 0; i < discipline.questions.Length; i++)
                {
                    TextBox textBox = this.Controls.Find("textBoxQuestion" + (i + 1),
true).FirstOrDefault() as TextBox;
                    textBox.Text = discipline.questions[i];
                    textBox.Visible = true;

                    TrackBar trackBar = this.Controls.Find("trackBarAnswer" + (i + 1),
true).FirstOrDefault() as TrackBar;
                    trackBar.Visible = true;
                    trackBar.Value = 50;
                }
                for (; i < 10; i++)
                {
                    TextBox textBox = this.Controls.Find("textBoxQuestion" + (i + 1),
true).FirstOrDefault() as TextBox;
                    textBox.Text = "";
                    textBox.Visible = false;

                    TrackBar trackBar = this.Controls.Find("trackBarAnswer" + (i + 1),
true).FirstOrDefault() as TrackBar;
                    trackBar.Visible = false;
                    trackBar.Value = 50;
                }
            }
        }

        private void buttonNext_Click(object sender, EventArgs e)
        {
            for (int i = 0; i < 10; i++)
            {
                TrackBar trackBar = this.Controls.Find("trackBarAnswer" + (i + 1),
true).FirstOrDefault() as TrackBar;
                if (trackBar.Visible)
                    if (currQuestion < pollResult.Length)
                    {
                        pollResult[currQuestion] = trackBar.Value;
                        currQuestion = currQuestion + 1;
                    }
            }

            currPage = currPage + 1;

            if (currPage + 1 == pollData.disciplines.Length)
                buttonNext.Text = "Завершити";
```

```
            if (currPage == pollData.disciplines.Length)
            {
                if (usingFiles.writeFileArray("pollResult", ".array", pollResult))
                    MessageBox.Show("Результати опитування збережено!", "Завершення
опитування", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
                else
                    MessageBox.Show("Результати опитування не вдалося зберегти!",
"Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                groupBoxPoll.Visible = false;
            }
            else
                setDataForCurrentPage();
        }

        private void imageClass_Click(object sender, EventArgs e)
        {
            for (int i = 0; i < 4; i++)
            {
                Label label = this.Controls.Find("imageClass" + (i + 1),
true).FirstOrDefault() as Label;
                if (label == sender)
                {
                    label.BorderStyle = BorderStyle.Fixed3D;
                    baseClass = i + 1;
                }
                else
                    label.BorderStyle = BorderStyle.None;
            }
        }

        private void buttonStartTraining_Click(object sender, EventArgs e)
        {
            if (currInterfase == interfaceAdmin)
            {
                buttonVisualizeSelectedDelta.Visible = false;
                buttonVisualizeOptimalDelta.Visible = false;
                buttonExam.Visible = false;
                groupBoxLearningResult.Visible = false;
                groupBoxExamResult.Visible = false;
            }

            if (! inputDescription.createLearningMatrix(volTrainingSample))
                return;

            baseAlgorithm = new BasicAlgorithm[maxDelta];

            for (int d = 0; d < maxDelta; d++)
                baseAlgorithm[d] = new BasicAlgorithm(inputDescription, baseClass,
volTrainingSample, d + 1);

            EAverageKulbak  = findEAverage(KulbaksCriterion);
            EAverageShannon = findEAverage(ShannonCriterion);

            workArea = findWorkArea();

            if (currInterfase == interfaceAdmin)
            {
                MessageBox.Show("Процес навчання завершено!", "Завершення навчання",
MessageBoxButtons.OK, MessageBoxIcon.Information);

                showLearningResult();
            }

        }
```

```
        public void showLearningResult()
        {
            int delta;
            try
            {
                delta = int.Parse(deltaValue.Text);
                if (delta < 1 || delta > maxDelta)
                {
                    MessageBox.Show("Параметр кроку δ може приймати значення від 1 до " +
maxDelta + "!", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return;
                }
            }
            catch
            {
                MessageBox.Show("Параметр кроку δ може приймати значення від 1 до " +
maxDelta + "!", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            buttonVisualizeSelectedDelta.Visible    = true;
            buttonVisualizeOptimalDelta.Visible     = true;
            buttonExam.Visible                      = true;
            groupBoxLearningResult.Visible          = true;

            int initialDelta = int.Parse(deltaValue.Text);

            learningResultSelectedDelta.Text = "Результат навчання для обраного кроку δ
= " + initialDelta + ":" + Environment.NewLine + Environment.NewLine;

            for (int i = 0; i < 4; i++)
            {
                int optimalRadius = (currCriterion == ShannonCriterion) ?
baseAlgorithm[initialDelta - 1].optimalRadiusShannon[i] : baseAlgorithm[initialDelta -
1].optimalRadiusKulbak[i];
                double Emax = (currCriterion == ShannonCriterion) ? ((optimalRadius ==
0) ? 0 : baseAlgorithm[initialDelta - 1].EShannon[i][optimalRadius - 1]) :
((optimalRadius == 0) ? 0 : baseAlgorithm[initialDelta - 1].EKulbak[i][optimalRadius -
1]);
                learningResultSelectedDelta.Text = learningResultSelectedDelta.Text +
"Клас " + (i + 1) + ":    d" + (i + 1) + " = " + optimalRadius + ", " + ((optimalRadius <
10) ? "      ":"    ") + "E(d" + (i + 1) + ") = " + Math.Round(Emax, 3) +
Environment.NewLine;
            }

            learningResultSelectedDelta.Text = learningResultSelectedDelta.Text +
Environment.NewLine + "Ē(δ) = " + ((currCriterion == ShannonCriterion) ?
Math.Round(EAverageShannon[initialDelta - 1], 5) : Math.Round(EAverageKulbak[initialDelta
- 1], 5));

            int optimalDelta = findOptimalDelta();

            learningResultOptimalDelta.Text = "Результат навчання для оптимального кроку
δ* = " + optimalDelta + ":" + Environment.NewLine + Environment.NewLine;

            for (int i = 0; i < 4; i++)
            {
                int optimalRadius = (currCriterion == ShannonCriterion) ?
baseAlgorithm[optimalDelta - 1].optimalRadiusShannon[i] : baseAlgorithm[optimalDelta -
1].optimalRadiusKulbak[i];
                double Emax = (currCriterion == ShannonCriterion) ? ((optimalRadius ==
0) ? 0 : baseAlgorithm[optimalDelta - 1].EShannon[i][optimalRadius - 1]) :
((optimalRadius == 0) ? 0 : baseAlgorithm[optimalDelta - 1].EKulbak[i][optimalRadius -
1]);
```

```
                learningResultOptimalDelta.Text = learningResultOptimalDelta.Text + "Клас
" + (i + 1) + ":    d" + (i + 1) + "* = " + optimalRadius + ", " + ((optimalRadius <
10) ? "      " : "    ") + "E(d" + (i + 1) + "*) = " + Math.Round(Emax, 3) +
Environment.NewLine;
            }

            learningResultOptimalDelta.Text = learningResultOptimalDelta.Text +
Environment.NewLine + "Ē(δ*) = " + ((currCriterion == ShannonCriterion) ?
Math.Round(EAverageShannon[optimalDelta - 1],5) : Math.Round(EAverageKulbak[optimalDelta
- 1], 5));
        }

        private void criterion_CheckedChanged(object sender, EventArgs e)
        {
            currCriterion = (RadioButton)sender;

            if (groupBoxLearningResult.Visible)
                showLearningResult();
            if (groupBoxExamResult.Visible)
                showExamResult();
        }

        private void buttonExam_Click(object sender, EventArgs e)
        {
            if (currInterfase == interfaceAdmin)
                groupBoxExamResult.Visible = false;

            if (!inputDescription.createExamMatrix(volTrainingSample))
                return;

            examAlgorithm = new ExamAlgorithm(inputDescription,
baseAlgorithm[findOptimalDelta()-1]);

            if (currInterfase == interfaceAdmin)
                MessageBox.Show("Процес екзамену завершено!", "Завершення екзамену",
MessageBoxButtons.OK, MessageBoxIcon.Information);

                showExamResult();
        }

        public void showExamResult()
        {
            if (currInterfase == interfaceAdmin)
            {
                groupBoxExamResult.Visible = true;

                textBoxExamResult.Text = "Функції належності до класів розпізнавання: " +
Environment.NewLine + Environment.NewLine;

                for (int i = 0; i < 4; i++)
                {
                    textBoxExamResult.Text = textBoxExamResult.Text + "Клас " + (i + 1) +
":    " + "μ = " + ((currCriterion == ShannonCriterion) ?
Math.Round(examAlgorithm.functionMembershipShannon[i], 3) :
Math.Round(examAlgorithm.functionMembershipKulbak[i], 3)) + Environment.NewLine;
                }

                int result = (currCriterion == ShannonCriterion) ?
examAlgorithm.recognizedClassShannon : examAlgorithm.recognizedClassKulbak;
                textBoxExamResult.Text = textBoxExamResult.Text + Environment.NewLine +
"Результат розпізнавання: Клас " + ((result == 0) ? "не розпізнано" : result.ToString());
            }
            else
            {
```

```
                int result = (currCriterion == ShannonCriterion) ?
examAlgorithm.recognizedClassShannon : examAlgorithm.recognizedClassKulbak;

                textBoxResult.Text = "Результат оцінювання:  " + ((result == 1) ?
"Відмінно" : ((result == 2) ? "Добре" : ((result == 3) ? "Задовільно" : ((result == 4) ?
"Незадовільно" : "Не визначено"))));

                textBoxResult.Visible = true;
            }
        }

        private void buttonVisualizeSelectedDelta_Click(object sender, EventArgs e)
        {
            int delta;
            try
            {
                delta = int.Parse(deltaValue.Text);
                if (delta < 1 || delta > maxDelta)
                {
                    MessageBox.Show("Параметр кроку δ може приймати значення від 1 до " +
maxDelta + "!", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return;
                }
            }
            catch
            {
                MessageBox.Show("Параметр кроку δ може приймати значення від 1 до " +
maxDelta + "!", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            double[] EAverage = new double[maxDelta];
            EAverage = (currCriterion == ShannonCriterion) ? EAverageShannon :
EAverageKulbak;
            int optimalDelta = findOptimalDelta();
            displayResults = new DisplayResults(delta, optimalDelta, currCriterion.Name,
baseAlgorithm[delta - 1], EAverage, workArea);

        }

        public double[] findEAverage(RadioButton criterion)
        {
            double[] EAverage = new double[maxDelta];
            for (int i = 0; i < maxDelta; i++)
            {
                List<double[]> E;
                int[] optimalRadius;
                if (criterion == ShannonCriterion)
                {
                    E = baseAlgorithm[i].EShannon;
                    optimalRadius = baseAlgorithm[i].optimalRadiusShannon;
                }
                else
                {
                    E = baseAlgorithm[i].EKulbak;
                    optimalRadius = baseAlgorithm[i].optimalRadiusKulbak;
                }

                double sumE = 0;
                for (int j = 0; j < E.Capacity; j++)
                    sumE = sumE + ((optimalRadius[j] == 0) ? 0 : E[j][optimalRadius[j] -
1]);
                EAverage[i] = sumE / E.Capacity;
            }
```

```
            return EAverage;
        }

        public int findOptimalDelta()
        {
            double maxEAverage = 0;
            int optimalDelta = 1;
            double[] EAverage = (currCriterion == ShannonCriterion) ? EAverageShannon :
EAverageKulbak;
            for (int i = 0; i < EAverage.Length; i++)
            {
                if (EAverage[i] > maxEAverage)
                {
                    maxEAverage = EAverage[i];
                    optimalDelta = i + 1;
                }
            }

            return optimalDelta;
        }

        private void deltaValue_TextChanged(object sender, EventArgs e)
        {
            int delta;
            try
            {
                delta = int.Parse(deltaValue.Text);
                if (delta < 1 || delta > maxDelta)
                {
                    MessageBox.Show("Параметр кроку δ може приймати значення від 1 до " +
maxDelta + "!", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return;
                }
            }
            catch
            {
                MessageBox.Show("Параметр кроку δ може приймати значення від 1 до " +
maxDelta + "!", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            if (groupBoxLearningResult.Visible)
                showLearningResult();
        }

        private void buttonVisualizeOptimalDelta_Click(object sender, EventArgs e)
        {
            int optimalDelta = findOptimalDelta();

            double[] EAverage = new double[maxDelta];
            EAverage = (currCriterion == ShannonCriterion) ? EAverageShannon :
EAverageKulbak;
            displayResults = new DisplayResults(optimalDelta, optimalDelta,
currCriterion.Name, baseAlgorithm[optimalDelta - 1], EAverage, workArea);
        }

        public bool[] findWorkArea()
        {
            bool[] workArea = new bool[maxDelta];
            for (int delta = 0; delta < baseAlgorithm.Length; delta++)
            {
                bool[] mark = new bool[4];

                for (int j = 0; j < 4; j++)
```

```
                {
                    mark[j] = false;

                    for (int i = 0; i < volTrainingSample; i++)
                    {
                        if (baseAlgorithm[delta].D1[j][i] > 0.5 &&
baseAlgorithm[delta].D2[j][i] > 0.5)
                            mark[j] = true;
                    }
                }

                workArea[delta] = true;
                for (int j = 0; j < mark.Length; j++)
                {
                    if (!mark[j])
                    {
                        workArea[delta] = false;
                        break;
                    }
                }
            }
            return workArea;
        }

        private void buttonResult_Click(object sender, EventArgs e)
        {
            groupBoxPoll.Visible = false;
            textBoxResult.Visible = false;

            buttonStartTraining_Click(null, null);

            buttonExam_Click(null, null);
        }
    }
```

Опис програмного класу PollData.cs:

```
    class PollData
    {
        public Discipline[] disciplines {get; set;}

        public int getCountOfQuestions()
        {
            int countOfQuestions = 0;
            for (int i = 0; i < disciplines.Length; i++)
                countOfQuestions = countOfQuestions + disciplines[i].questions.Length;

            return countOfQuestions;
        }
    }

    class Discipline
    {
        public string title { get; set; }
        public string[] questions { get; set; }
    }
```

Опис програмного класу InputDescription.cs:

```
    class InputDescription
    {
        public List<double[,]> classes;
```

```
        public double[,] examMatrix;

        UsingFiles usingFiles;

        public struct classRanges
        {
            public double center;
            public double deviation;
            public string fileName;

            public classRanges(int center, int deviation, string filleName)
            {
                this.center = center;
                this.deviation = deviation;
                this.fileName = filleName;
            }
        }

        public InputDescription()
        {
            usingFiles = new UsingFiles();
        }

        public bool createLearningMatrix(int martixSize)
        {
            classes = new List<double[,]>(4);
            try
            {
                for (int i = 0; i < classes.Capacity; i++)
                {
                    classRanges ranges = getClassRanges(i + 1);

                    classes.Add(createMatrixFromFile(martixSize, ranges));
                }
            }
            catch
            {
                MessageBox.Show("Не вдалось прочитати навчальні матриці з файлів!",
"Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return false;
            }
            return true;
        }

        public bool createExamMatrix(int matrixSize)
        {
            double[] matrixArray;
            classRanges ranges = getClassRanges(0);

            matrixArray = usingFiles.readFileArray(ranges.fileName, ".array");

            if (matrixArray.Length < matrixSize * matrixSize)
            {
                DialogResult dialogResult = MessageBox.Show("Результатів опитування
недостатньо для формування екзаменаційної матриці. " +
                    "Сформувати випадкові результати?", "Формування вхідного
математичного опису", MessageBoxButtons.OKCancel);

                if (dialogResult == DialogResult.OK)
                    examMatrix = createRandomMatrix(matrixSize, ranges);
                else
                    return false;
            }
            else
```

```
            {
                examMatrix = new double[matrixSize, matrixSize];
                int index = 0;

                for (int i = 0; i < matrixSize; i++)
                    for (int j = 0; j < matrixSize; j++)
                        if (index < matrixArray.Length)
                        {
                            examMatrix[i, j] = matrixArray[index];
                            index++;
                        }
            }

            return true;
        }

        public double[,] createRandomMatrix(int martixSize, classRanges ranges)
        {
            double[,] matrix = new double[martixSize, martixSize];

            MathNet.Numerics.Distributions.Normal normalRandom = new
Normal(ranges.center, ranges.deviation);
            double min = ranges.center - ranges.deviation;
            double max = ranges.center + ranges.deviation;

            for (int i = 0; i < martixSize; i++)
            {
                for (int j = 0; j < martixSize; j++)
                {
                    int randomGaussianValue = Convert.ToInt32(normalRandom.Sample());
                    while (randomGaussianValue < min || randomGaussianValue > max)
                    {
                        randomGaussianValue = Convert.ToInt32(normalRandom.Sample());
                    }

                    matrix[i, j] = randomGaussianValue;
                }
            }
            return matrix;
        }

        public double[,] createMatrixFromFile(int matrixSize, classRanges ranges)
        {
            double[] matrixArray;
            double[,] learningMatrix;

            matrixArray = usingFiles.readFileArray(ranges.fileName, ".array");

            learningMatrix = new double[matrixSize, matrixSize];

            int index = 0;

            for (int i = 0; i < matrixSize; i++)
                for (int j = 0; j < matrixSize; j++)
                    if (index < matrixArray.Length)
                    {
                        learningMatrix[i, j] = matrixArray[index];
                        index++;
                    }

            return learningMatrix;
        }
    }
```

Опис програмного класу BasicAlgorithm:

```
    class BasicAlgorithm
    {
        InputDescription inputDescription;
        int baseClass;
        int matrixSize;
        int delta;
        public double[] midVector;
        public double[] lowerTolerance;
        public double[] upperTolerance;
        public List<int[]> binVectors;
        public List<double[]> D1;
        public List<double[]> D2;
        public int[] optimalRadiusShannon;
        public int[] optimalRadiusKulbak;
        public List<double[]> EShannon;
        public List<double[]> EKulbak;

        public BasicAlgorithm(InputDescription inputDescription, int baseClass, int
matrixSize, int delta)
        {
            this.inputDescription = inputDescription;
            this.baseClass = baseClass;
            this.matrixSize = matrixSize;
            this.delta = delta;

            startTraining();
        }

        public void startTraining()
        {
            midVector = createMidVector();

            lowerTolerance = createTolerance(midVector, delta, 0);

            upperTolerance = createTolerance(midVector, delta, 1);

            List<double[,]> binMatrix = new List<double[,]>(4);
            for (int i = 0; i < binMatrix.Capacity; i++)
            {
                binMatrix.Add(createBinMatrix(inputDescription.classes[i]));
            }

            binVectors = new List<int[]>(4);
            for (int i = 0; i < binVectors.Capacity; i++)
            {
                binVectors.Add(createBinVector(binMatrix[i]));
            }

            int[] PARA = new int[4];
            for (int i = 0; i < PARA.Length; i++)
            {
                PARA[i] = findMinCenterDistanceClass(i, binVectors);
            }

            List<int[,]> codeDistances = new List<int[,]>(4);
            for (int i = 0; i < codeDistances.Capacity; i++)
            {
                codeDistances.Add(createCodeDistances(binVectors[i], binMatrix[i],
binMatrix[PARA[i]]));
            }

            List<int[]> K1 = new List<int[]>(4);
```

```
for (int i = 0; i < K1.Capacity; i++)
{
    K1.Add(findNumberEvents(codeDistances[i], 0));
}
List<int[]> K3 = new List<int[]>(4);
for (int i = 0; i < K3.Capacity; i++)
{
    K3.Add(findNumberEvents(codeDistances[i], 1));
}
List<int[]> K2 = new List<int[]>(4);
for (int i = 0; i < K2.Capacity; i++)
{
    K2.Add(findNumberEventsK2K4(K1[i]));
}
List<int[]> K4 = new List<int[]>(4);
for (int i = 0; i < K4.Capacity; i++)
{
    K4.Add(findNumberEventsK2K4(K3[i]));
}

D1 = new List<double[]>(4);
for (int i = 0; i < D1.Capacity; i++)
{
    D1.Add(findCharacteristic(K1[i]));
}
List<double[]> Betta = new List<double[]>(4);
for (int i = 0; i < Betta.Capacity; i++)
{
    Betta.Add(findCharacteristic(K3[i]));
}
D2 = new List<double[]>(4);
for (int i = 0; i < D2.Capacity; i++)
{
    D2.Add(findCharacteristicD2Alfa(Betta[i]));
}
List<double[]> Alfa = new List<double[]>(4);
for (int i = 0; i < Alfa.Capacity; i++)
{
    Alfa.Add(findCharacteristicD2Alfa(D1[i]));
}

EKulbak = new List<double[]>(4);
for (int i = 0; i < EKulbak.Capacity; i++)
{
    EKulbak.Add(findKFEKulbakWorkingFormula(K2[i], K3[i]));
}

EShannon = new List<double[]>(4);
for (int i = 0; i < EShannon.Capacity; i++)
{
    EShannon.Add(findKFEShannonWorkingFormula(K1[i], K2[i], K3[i], K4[i]));
}

optimalRadiusKulbak = new int[4];
for (int i = 0; i < optimalRadiusKulbak.Length; i++)
{
    optimalRadiusKulbak[i] = findOptimalRadius(EKulbak[i], D1[i], D2[i]);
}

optimalRadiusShannon = new int[4];
for (int i = 0; i < optimalRadiusShannon.Length; i++)
{
    optimalRadiusShannon[i] = findOptimalRadius(EShannon[i], D1[i], D2[i]);
}
```

```
        }
        public double[] createMidVector()
        {
            double[] midVector = new double[matrixSize];

            for (int j = 0; j < matrixSize; j++)
            {
                double sum = 0;
                for (int i = 0; i < matrixSize; i++)
                {
                    sum = sum + inputDescription.classes[baseClass - 1][i, j];
                }
                midVector[j] = sum / matrixSize;
            }

            return midVector;
        }

        public double[] createTolerance(double[] midVector, double delta, int type)
        {
            double[] tolerance = new double[matrixSize];

            for (int j = 0; j < matrixSize; j++)
            {
                if (type == 0)
                    tolerance[j] = midVector[j] - delta;
                else
                    tolerance[j] = midVector[j] + delta;
            }

            return tolerance;
        }

        public double[,] createBinMatrix(double[,] matrix)
        {
            double[,] binMatrix = new double[matrixSize, matrixSize];

            for (int x = 0; x < matrixSize; x++)
            {
                for (int y = 0; y < matrixSize; y++)
                {
                    if (matrix[x, y] > lowerTolerance[y] && matrix[x, y] <
upperTolerance[y])
                        binMatrix[x, y] = 1;
                    else
                        binMatrix[x, y] = 0;
                }
            }

            return binMatrix;
        }

        public int[] createBinVector(double[,] binMatrix)
        {
            int[] binVector = new int[matrixSize];
            for (int j = 0; j < matrixSize; j++)
            {
                double sum = 0;
                for (int i = 0; i < matrixSize; i++)
                {
                    sum = sum + binMatrix[i, j];
                }

                if (sum / matrixSize > 0.5)
```

```
                binVector[j] = 1;
            else
                binVector[j] = 0;
        }

        return binVector;
    }

    public int findMinCenterDistanceClass(int index, List<int[]> binVectors)
    {
        int[] distances = new int[binVectors.Capacity];
        int minDistance;
        int numbOfClass;

        for (int i = 0; i < binVectors.Capacity; i++)
        {
            distances[i] = (vectorsXOR(binVectors[index], binVectors[i]));
        }

        if (index == binVectors.Capacity - 1)
            numbOfClass = 0;
        else
            numbOfClass = index + 1;

        minDistance = distances[numbOfClass];

        for (int i = 0; i < distances.Length; i++)
        {
            if (i != index && distances[i] < minDistance)
            {
                numbOfClass = i;
                minDistance = distances[numbOfClass];
            }
        }

        return numbOfClass;
    }

    public int vectorsXOR(int[] binVectorMainClass, int[] binVectorNeighborClass)
    {
        int distance = 0;
        for (int i = 0; i < binVectorMainClass.Length; i++)
        {
            if (binVectorMainClass[i] != binVectorNeighborClass[i])
                distance++;
        }

        return distance;
    }

    public int[,] createCodeDistances(int[] binVector, double[,] binMatrixMainClass,
double[,] binMatrixNeighborClass)
    {
        int[,] codeDistances = new int[matrixSize, 2];
        for (int i = 0; i < matrixSize; i++)
        {
            for (int j = 0; j < matrixSize; j++)
            {
                codeDistances[i, 0] = codeDistances[i, 0] +
(Math.Abs((int)binMatrixMainClass[i, j] - binVector[j]));
            }
        }
        for (int i = 0; i < matrixSize; i++)
        {
```

```
                for (int j = 0; j < matrixSize; j++)
                {
                    codeDistances[i, 1] = codeDistances[i, 1] +
(Math.Abs((int)binMatrixNeighborClass[i, j] - binVector[j]));
                }
            }

            return codeDistances;
        }

        public int[] findNumberEvents(int[,] codeDistance, int type)
        {
            int[] K = new int[matrixSize];
            for (int d = 0; d < matrixSize; d++)
            {
                for (int i = 0; i < matrixSize; i++)
                {
                    if (codeDistance[i, type] <= d + 1)
                        K[d] = K[d] + 1;
                }
            }

            return K;
        }

        public int[] findNumberEventsK2K4(int[] K)
        {
            int[] K_ = new int[K.Length];
            for (int i = 0; i < K.Length; i++)
            {
                K_[i] = K.Length - K[i];
            }

            return K_;
        }

        public double[] findCharacteristic(int[] K)
        {
            double[] characteristic = new double[K.Length];
            for (int i = 0; i < K.Length; i++)
                characteristic[i] = (double)K[i] / K.Length;

            return characteristic;
        }

        public double[] findCharacteristicD2Alfa(double[] D)
        {
            double[] characteristic = new double[D.Length];
            for (int i = 0; i < D.Length; i++)
                characteristic[i] = 1 - D[i];

            return characteristic;
        }

        public double[] findKFEShannonWorkingFormula(int[] K1, int[] K2, int[] K3, int[]
K4)
        {
            double[] E = new double[matrixSize];

            for (int i = 0; i < E.Length; i++)
            {
```

```
                double part1 = (double.IsNaN((double)K1[i] / (K1[i] + K3[i])) ? 0 :
((double)K1[i] / (K1[i] + K3[i]))) * Math.Log(double.IsNaN((double)K1[i] / (K1[i] +
K3[i])) ? 0 : ((double)K1[i] / (K1[i] + K3[i])), 2);
                if (double.IsNaN(part1))
                    part1 = 0;
                double part2 = (double.IsNaN((double)K2[i] / (K2[i] + K4[i])) ? 0 :
((double)K2[i] / (K2[i] + K4[i]))) * Math.Log(double.IsNaN((double)K2[i] / (K2[i] +
K4[i])) ? 0 : ((double)K2[i] / (K2[i] + K4[i])), 2);
                if (double.IsNaN(part2))
                    part2 = 0;
                double part3 = (double.IsNaN((double)K4[i] / (K4[i] + K2[i])) ? 0 :
((double)K4[i] / (K4[i] + K2[i]))) * Math.Log(double.IsNaN((double)K4[i] / (K4[i] +
K2[i])) ? 0 : ((double)K4[i] / (K4[i] + K2[i])), 2);
                if (double.IsNaN(part3))
                    part3 = 0;
                double part4 = (double.IsNaN((double)K3[i] / (K3[i] + K1[i])) ? 0 :
((double)K3[i] / (K3[i] + K1[i]))) * Math.Log(double.IsNaN((double)K3[i] / (K3[i] +
K1[i])) ? 0 : ((double)K3[i] / (K3[i] + K1[i])), 2);
                if (double.IsNaN(part4))
                    part4 = 0;
                E[i] = 1 + 0.5 * (part1 + part2 + part3 + part4);

                if (double.IsNaN(E[i]))
                    E[i] = 0;
            }

            return E;
        }

        public double[] findKFEKulbakWorkingFormula(int[] K2, int[] K3)
        {
            double[] E = new double[matrixSize];
            double maxE = 0;
            for (int i = 0; i < E.Length; i++)
            {
                E[i] = (1.0 / matrixSize) * Math.Log((2 * matrixSize + Math.Pow(10, 1) -
(K2[i] + K3[i])) / ((K2[i] + K3[i]) + Math.Pow(10, 1)), 2) * (matrixSize - (K2[i] +
K3[i]));

                if (double.IsNaN(E[i]))
                    E[i] = 0;
                if (!double.IsInfinity(E[i]) && E[i] > maxE)
                    maxE = E[i];
            }
            for (int i = 0; i < E.Length; i++)
                if (double.IsInfinity(E[i]))
                    E[i] = maxE;

            return E;
        }

        public int findOptimalRadius(double[] E, double[] D1, double[] D2)
        {
            double max = 0;
            int optimalRadius = 0;

            for (int i = 0; i < E.Length; i++)
            {
                if (E[i] > max && D1[i] > 0.5 && D2[i] > 0.5)
                {
                    optimalRadius = i + 1;
                    max = E[i];
                }
            }
```

```
            return optimalRadius;
        }
    }
```

Опис програмного класу ExamAlgorithm.cs:

```
    class ExamAlgorithm
    {
        public double[] functionMembershipKulbak;
        public double[] functionMembershipShannon;
        public int recognizedClassKulbak;
        public int recognizedClassShannon;

        InputDescription inputDescription;
        BasicAlgorithm basicAlgorithm;

        public ExamAlgorithm(InputDescription inputDescription, BasicAlgorithm
basicAlgorithm)
        {
            this.inputDescription = inputDescription;
            this.basicAlgorithm = basicAlgorithm;

            startExam();
        }

        public void startExam()
        {
            double[,] binMartix =
basicAlgorithm.createBinMatrix(inputDescription.examMatrix);

            int[] binVector = basicAlgorithm.createBinVector(binMartix);

            int[] codeDistances = new int[4];
            for (int i = 0; i < codeDistances.Length; i++)
            {
                codeDistances[i] = basicAlgorithm.vectorsXOR(binVector,
basicAlgorithm.binVectors[i]);
            }

            functionMembershipKulbak = new double[codeDistances.Length];
            for (int i = 0; i < codeDistances.Length; i++)
            {
                functionMembershipKulbak[i] = findFunctionAccessories(codeDistances[i],
basicAlgorithm.optimalRadiusKulbak[i]);
            }

            functionMembershipShannon = new double[codeDistances.Length];
            for (int i = 0; i < codeDistances.Length; i++)
            {
                functionMembershipShannon[i] = findFunctionAccessories(codeDistances[i],
basicAlgorithm.optimalRadiusShannon[i]);
            }

            recognizedClassKulbak = findRecognizedClass(functionMembershipKulbak);

            recognizedClassShannon = findRecognizedClass(functionMembershipShannon);
        }

        public double findFunctionAccessories(int codeDistance, int optimalRadius)
        {
            return 1 - ((double)codeDistance / optimalRadius);
        }

        public int findRecognizedClass(double[] functionMembership)
```

```
        {
            int numberOfClass = 0;
            double maxFunctionMembership = functionMembership[0];

            for(int i = 0; i < functionMembership.Length; i++)
            {
                if (functionMembership[i] > maxFunctionMembership)
                {
                    numberOfClass = i;
                    maxFunctionMembership = functionMembership[numberOfClass];
                }
            }
            if (maxFunctionMembership < 0)
                return 0;

            return numberOfClass + 1;
        }
    }
```

Опис програмного класу DisplayResults.cs:

```
    partial class DisplayResults : System.Windows.Forms.Form
    {
        BasicAlgorithm basicAlgorithm;
        int delta;
        int optimalDelta;
        string criterion;
        double[] EAverage;
        bool[] workArea;

        public DisplayResults(int delta, int optimalDelta, string criterion,
BasicAlgorithm basicAlgorithm, double[] EAverage, bool[] workArea)
        {
            InitializeComponent();

            this.basicAlgorithm = basicAlgorithm;
            this.delta = delta;
            this.optimalDelta = optimalDelta;
            this.criterion = criterion;
            this.EAverage = EAverage;
            this.workArea = workArea;

            labelDelta.Text = labelDelta.Text + delta;

            showResults();
        }

        public void showResults()
        {
            buildControlToleranceSystem(chartControlToleranceSystem);

            List<double[]> E = (criterion == "ShannonCriterion") ?
basicAlgorithm.EShannon : basicAlgorithm.EKulbak;
            int[] optimalRadius = (criterion == "ShannonCriterion") ?
basicAlgorithm.optimalRadiusShannon : basicAlgorithm.optimalRadiusKulbak;

            for (int i = 0; i < 4; i++)
            {
                Chart chartClass = this.Controls.Find("chartClass" + (i + 1),
true).FirstOrDefault() as Chart;
                buildChartClass(chartClass, (i + 1), E[i], optimalRadius[i],
basicAlgorithm.D1[i], basicAlgorithm.D2[i]);
            }
```

```
            buildEAverage(chartEAverage);

            this.Show();
        }

        public void buildControlToleranceSystem(Chart chart)
        {
            chart.BorderlineColor = Color.Black;
            chart.BorderlineDashStyle = ChartDashStyle.Solid;
            chart.BorderlineWidth = 1;

            chart.ChartAreas[0].BackColor = Color.White;
            chart.ChartAreas[0].AxisX.Minimum = 0;
            chart.ChartAreas[0].AxisX.Maximum = basicAlgorithm.midVector.Length;
            double minVal = Math.Round(basicAlgorithm.lowerTolerance.Min(), 0);
            double maxVal = Math.Round(basicAlgorithm.upperTolerance.Max(), 0);
            minVal = (minVal < 0) ? minVal : 0;
            maxVal = (maxVal > 100) ? maxVal : 100;
            chart.ChartAreas[0].AxisY.Minimum = minVal;
            chart.ChartAreas[0].AxisY.Maximum = (maxVal == 0) ? 1 : maxVal;
            chart.ChartAreas[0].AxisX.Interval = 5;

            for (int i = 0; i < basicAlgorithm.upperTolerance.Length; i++)
                chart.Series[0].Points.Add(basicAlgorithm.upperTolerance[i]);
            for (int i = 0; i < basicAlgorithm.midVector.Length; i++)
                chart.Series[1].Points.Add(basicAlgorithm.midVector[i]);
            for (int i = 0; i < basicAlgorithm.lowerTolerance.Length; i++)
                chart.Series[2].Points.Add(basicAlgorithm.lowerTolerance[i]);
        }

        public void buildChartClass(Chart chart, int numbOfClass, double[] E, int
optimalPadius, double[] D1, double[] D2)
        {
            chart.BorderlineColor = Color.Black;
            chart.BorderlineDashStyle = ChartDashStyle.Solid;
            chart.BorderlineWidth = 1;

            chart.ChartAreas[0].BackColor = Color.White;
            chart.ChartAreas[0].AxisX.Minimum = 0;
            chart.ChartAreas[0].AxisX.Maximum = E.Count();
            double minVal = Math.Floor(E.Min());
            double maxVal = Math.Ceiling(E.Max());
            chart.ChartAreas[0].AxisY.Minimum = minVal;
            chart.ChartAreas[0].AxisY.Maximum = (maxVal == 0) ? 1 : maxVal;
            chart.ChartAreas[0].AxisX.Interval = 5;
            chart.ChartAreas[0].AxisY.Interval = (criterion == "ShannonCriterion") ?
0.2 : 1;
            chart.ChartAreas[0].AxisX.Title = "d" + numbOfClass;
            chart.ChartAreas[0].AxisY.Title = "E";

            chart.Series.Clear();

            Series seriesKFE = new Series("КФЕ");
            seriesKFE.ChartType = SeriesChartType.Spline;
            seriesKFE.Color = Color.Black;
            seriesKFE.BorderWidth = 2;

            Series seriesWorkArea = new Series("Робоча область");
            seriesWorkArea.ChartType = SeriesChartType.SplineArea;
            seriesWorkArea.Color = Color.LightGray;

            Series seriesOptimalRadius = new Series("d" + numbOfClass + ((delta ==
optimalDelta) ? "*" : "") + " = " + optimalPadius);
            seriesOptimalRadius.ChartType = SeriesChartType.Line;
```

```
            seriesOptimalRadius.Color = Color.Blue;
            seriesOptimalRadius.BorderWidth = 2;

            for (int i = 0; i < E.Length; i++)
                seriesKFE.Points.AddXY(i + 1, E[i]);

            for (int i = 0; i < E.Length; i++)
            {
                if (D1[i] > 0.5 && D2[i] > 0.5)
                    seriesWorkArea.Points.AddXY(i + 1, E[i]);
            }

            if (optimalPadius != 0)
            {
                seriesOptimalRadius.Points.AddXY(optimalPadius, 0);
                seriesOptimalRadius.Points.AddXY(optimalPadius, E[optimalPadius - 1]);
                seriesOptimalRadius.Points.AddXY(optimalPadius, 0);
            }

            chart.Series.Add(seriesKFE);
            chart.Series.Add(seriesWorkArea);
            chart.Series.Add(seriesOptimalRadius);
        }


        public void buildEAverage(Chart chart)
        {
            chart.BorderlineColor = Color.Black;
            chart.BorderlineDashStyle = ChartDashStyle.Solid;
            chart.BorderlineWidth = 1;

            chart.ChartAreas[0].BackColor = Color.White;
            chart.ChartAreas[0].AxisX.Minimum = 0;
            chart.ChartAreas[0].AxisX.Maximum = EAverage.Length;
            double minVal = Math.Floor(EAverage.Min());
            double maxVal = Math.Ceiling(EAverage.Max());
            chart.ChartAreas[0].AxisY.Minimum = minVal;
            chart.ChartAreas[0].AxisY.Maximum = (maxVal == 0) ? 1 : maxVal;
            chart.ChartAreas[0].AxisX.Interval = 5;
            chart.ChartAreas[0].AxisY.Interval = (criterion == "ShannonCriterion") ?
0.2 : 1;
            chart.ChartAreas[0].AxisX.Title = "δ";
            chart.ChartAreas[0].AxisY.Title = "Ē";

            chart.Series.Clear();

            Series seriesKFE = new Series("КФЕ");
            seriesKFE.ChartType = SeriesChartType.Spline;
            seriesKFE.Color = Color.Black;
            seriesKFE.BorderWidth = 2;

            Series seriesWorkArea = new Series("Робоча область");
            seriesWorkArea.ChartType = SeriesChartType.Column;
            seriesWorkArea.Color = Color.LightGray;

            Series seriesOptimalDelta = new Series("δ* = " + optimalDelta);
            seriesOptimalDelta.ChartType = SeriesChartType.Line;
            seriesOptimalDelta.Color = Color.Blue;
            seriesOptimalDelta.BorderWidth = 2;

            seriesKFE.Points.Clear();

            for (int i = 0; i < EAverage.Length; i++)
                seriesKFE.Points.AddXY(i + 1, EAverage[i]);
```

```
            for (int i = 0; i < workArea.Length; i++)
            {
                if (workArea[i])
                    seriesWorkArea.Points.AddXY(i + 1, EAverage[i]);
            }

            seriesOptimalDelta.Points.AddXY(optimalDelta, 0);
            seriesOptimalDelta.Points.AddXY(optimalDelta, EAverage[optimalDelta - 1]);
            seriesOptimalDelta.Points.AddXY(optimalDelta, 0);

            chart.Series.Add(seriesKFE);
            chart.Series.Add(seriesWorkArea);
            chart.Series.Add(seriesOptimalDelta);
        }

        private void buttonSaveToExcel_Click(object sender, EventArgs e)
        {
            FolderBrowserDialog folderBrowserDialog = new FolderBrowserDialog();
            folderBrowserDialog.Description = "Вибір каталогу для збереження";
            folderBrowserDialog.SelectedPath = @"C:\";
            folderBrowserDialog.ShowDialog();

            string pathToFile = folderBrowserDialog.SelectedPath + "\\" + "Delta" + delta
+ "_" + criterion + ".xls";

            Excel.Application xlApp;
            Excel.Workbook xlWorkBook;
            Excel.Worksheet xlWorkSheet1, xlWorkSheet2;
            object misValue = System.Reflection.Missing.Value;

            xlApp = new Excel.Application();
            xlWorkBook = xlApp.Workbooks.Add(misValue);
            xlWorkSheet1 = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(1);
            xlWorkSheet1.Name = "Оптимізація";

            saveChartOptimizationData(xlWorkSheet1, chartEAverage);

            xlWorkSheet2 = (Excel.Worksheet)xlWorkBook.Worksheets.Add();
            xlWorkSheet2.Name = "Дельта " + delta;

            for (int i = 0; i < 4; i++)
            {
                Chart chartClass = this.Controls.Find("chartClass" + (i + 1),
true).FirstOrDefault() as Chart;
                saveChartClassData(xlWorkSheet2, chartClass, i);
            }

            try
            {
                xlWorkBook.SaveAs(pathToFile, Excel.XlFileFormat.xlWorkbookNormal,
misValue, misValue, misValue, misValue, Excel.XlSaveAsAccessMode.xlExclusive, misValue,
misValue, misValue, misValue, misValue);
                xlWorkBook.Close(true, misValue, misValue);
                xlApp.Quit();
            }
            catch
            {
                MessageBox.Show("Не вдалось зберегти дані! Перевірте доступ до файлу " +
pathToFile, "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            releaseObject(xlWorkSheet1);
```

```
            releaseObject(xlWorkSheet2);
            releaseObject(xlWorkBook);
            releaseObject(xlApp);

            MessageBox.Show("Дані збережено! Шлях до файлу: " + pathToFile, "Збереження
даних", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
        }

        private void releaseObject(object obj)
        {
            try
            {
                System.Runtime.InteropServices.Marshal.ReleaseComObject(obj);
                obj = null;
            }
            catch (Exception ex)
            {
                obj = null;
                MessageBox.Show("Exception Occured while releasing object " +
ex.ToString());
            }
            finally
            {
                GC.Collect();
            }
        }

        public void saveChartClassData(Excel.Worksheet xlWorkSheet, Chart chart, int
indexOfClass)
        {
            int interval = 6 * indexOfClass;

            xlWorkSheet.Cells[1, 1 + interval] = "d" + (indexOfClass + 1);
            xlWorkSheet.Cells[1, 2 + interval] = "КФЕ";
            xlWorkSheet.Cells[1, 3 + interval] = "Робоча область";
            xlWorkSheet.Cells[1, 4 + interval] = "d" + (indexOfClass + 1) + "*";
            xlWorkSheet.Cells[2, 1 + interval] = 0;

            for (int j = 0; j < chart.Series[0].Points.Count; j++)
            {
                xlWorkSheet.Cells[j + 3, 1 + interval] =
chart.Series[0].Points[j].XValue;
                xlWorkSheet.Cells[j + 3, 2 + interval] =
chart.Series[0].Points[j].YValues[0];
            }
            for (int j = 0; j < chart.Series[1].Points.Count; j++)
            {
                double X = chart.Series[1].Points[j].XValue;
                xlWorkSheet.Cells[X + 2, 3 + interval] =
chart.Series[1].Points[j].YValues[0];
            }
            for (int j = 0; j < chart.Series[2].Points.Count; j++)
            {
                double X = chart.Series[2].Points[j].XValue;
                double Y = chart.Series[2].Points[j].YValues[0];
                if (Y > 0)
                    xlWorkSheet.Cells[X + 2, 4 + interval] =
chart.Series[2].Points[j].YValues[0];
            }
        }

        public void saveChartOptimizationData(Excel.Worksheet xlWorkSheet, Chart chart)
        {
            xlWorkSheet.Cells[1, 1] = "δ";
```

```
            xlWorkSheet.Cells[1, 2] = "КФЕ";
            xlWorkSheet.Cells[1, 3] = "Робоча область";
            xlWorkSheet.Cells[1, 4] = "δ*";
            xlWorkSheet.Cells[2, 1] = 0;
            for (int j = 0; j < chart.Series[0].Points.Count; j++)
            {
                xlWorkSheet.Cells[j + 3, 1] = chart.Series[0].Points[j].XValue;
                xlWorkSheet.Cells[j + 3, 2] = chart.Series[0].Points[j].YValues[0];
            }
            for (int j = 0; j < chart.Series[1].Points.Count; j++)
            {
                double X = chart.Series[1].Points[j].XValue;
                xlWorkSheet.Cells[X + 2, 3] = chart.Series[1].Points[j].YValues[0];
            }
            for (int j = 0; j < chart.Series[2].Points.Count; j++)
            {
                double X = chart.Series[2].Points[j].XValue;
                double Y = chart.Series[2].Points[j].YValues[0];
                if (Y > 0)
                    xlWorkSheet.Cells[X + 2, 4] = chart.Series[2].Points[j].YValues[0];
            }
        }
    }
```

Опис програмного класу UsingFiles.cs:

```
    class UsingFiles
    {
        public string url;

        public UsingFiles()
        {
            this.url = @"..\..\resources\";
        }

        public PollData readFileJSON(string fileReadName, string format)
        {
            PollData pollData;

            try
            {
                using (FileStream sr = File.OpenRead(url + fileReadName + format))
                {
                    byte[] byteText = new byte[sr.Length];

                    sr.Read(byteText, 0, byteText.Length);

                    string textFromFileJSON = new UTF8Encoding(true).GetString(byteText);

                    pollData = JsonConvert.DeserializeObject<PollData>(textFromFileJSON);
                }
            }
            catch
            {
                MessageBox.Show("Не вдалось прочитати дані!", "Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Error);

                pollData = new PollData();
            }

            return pollData;
        }
```

```
    public bool writeFileArray(string fileWriteName, string format, double[] array)
    {
        bool result = true;
        string textToWrite = "";

        try
        {
            using (StreamWriter sw = File.AppendText(url + fileWriteName + format))
            {
                for (int i = 0; i < array.Length; i++)

                    textToWrite = textToWrite + Convert.ToString(array[i] + " ");

                sw.Write(textToWrite);
            }
        }
        catch
        {
            result = false;
        }

        return result;
    }

    public double[] readFileArray(string fileReadName, string format)
    {
        double[] array;

        using (FileStream sr = File.OpenRead(url + fileReadName + format))
        {
            byte[] byteText = new byte[sr.Length];

            sr.Read(byteText, 0, byteText.Length);

            string textFromFile = new UTF8Encoding(true).GetString(byteText);

            string[] textValues = textFromFile.Split(' ');

            int countOfElements = textValues.Length;

            if (textValues.Last() == "")
                countOfElements = countOfElements - 1;

            array = new double[countOfElements];

            int index = 0;

            for (int i = 0; i < textValues.Length; i++)
                if (textValues[i] != "")
                {
                    array[index] = Double.Parse(textValues[i]);
                    index++;
                }
        }

        return array;
    }
}
```