НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«Київський політехнічний інститут імені Ігоря Сікорського»

Затверджую

Голова Приймальної комісії
Ректор

Михайло ЗГУРОВСЬКИЙ

28 квітня 2023

ПРОГРАМА
комплексного фахового випробування
для вступу на освітньо-наукову програму підготовки магістра
«Комп'ютерні науки»

за спеціальністю 122 Комп'ютерні науки

Програму ухвалено:
Науково-методичною комісією за спеціальністю
122 Комп'ютерні науки
Протокол № від «___» «_____» 2023 р.
Голова НМК

Наталія АУШЕВА

Київ – 2023

ВСТУП

Програму комплексного фахового випробування для вступу на освітньо-наукову програму підготовки магістра за спеціальністю 122 Комп'ютерні науки (далі — Програма) призначено для отримання досвіду самостійної роботи абітурієнта з підготовки до екзамену. Програму вступного випробування розроблено атестаційною комісією, ухвалено Науково- методичною комісією за спеціальністю 122 «Комп'ютерні науки» та затверджено Головою Приймальної комісії.

Метою програми є формування у абітурієнтів здатності ознайомитися із предметними питаннями курсів навчальних дисциплін, що включені в екзаменаційні білети; опрацювати підручники, навчальні посібники та інші інформаційно-літературні джерела предметної області знання; осмислено впорядкувати і систематизувати засвоєні теоретичні знання і практичні навички; мотивовано виконати роботу на екзамені, продемонструвавши певний рівень засвоєння навчальних дисциплін в результаті навчання.

Перелік навчальних дисциплін цієї Програми складають такі, що належать до циклу дисциплін базової підготовки навчального плану підготовки бакалавра напряму 122 Комп'ютерні науки:

1) Дослідження операцій;
2) Методи та системи штучного інтелекту;
3) Об'єктно-орієнтоване програмування;
4) Інтелектуальний аналіз даних.

Вступний екзамен проводиться чотири академічних години без перерви (180 хвилин) у спосіб одержання екзаменаційного білету — повернення письмової роботи. Метою на екзамені є розв'язання завдань екзаменаційного білету. Екзаменаційний білет містить чотири теоретичних або практичних завдання. Диференціації робочого часу, відведеного на виконання кожного завдання, немає. Фіксується час початку і закінчення роботи.

ПЕРЕЛІК НАВЧАЛЬНОГО МАТЕРІАЛУ, ЩО ВИНОСИТЬСЯ НА ФАХОВЕ ВИПРОБУВАННЯ

1. Навчальна дисципліна «Дослідження операцій»

Основні принципи дослідження операцій. Етапи операційного дослідження. Перевірка адекватності моделі при виконанні операційного дослідження. Основні класи задач дослідження операцій та їхня характеристика.

Постановка та математична модель задач лінійного програмування (ЛП). Форми запису задачі лінійного програмування. Канонічна та розгорнута форма. Множина допустимих розв'язків.

Постановка та математична модель задач лінійного програмування (ЛП).Графічний метод розв'язання задач лінійного програмування.

Постановка та математична модель багатокритеріальної задачі прийняття

рішень і її властивості. Властивості ефективних альтернатив і способи їх знаходження.

Методи розв'язання задач лінійного програмування Теоретичні основи симплекс-метода, алгоритм симплекс-метода. Зв'язок між елементами симплекс-таблиць, ознака оптимальності та нерозв'язності задачі.

Двоїста задача лінійного програмування. Зв'язок між формами запису прямої та двоїстої задач ЛП, основні теореми двоїстості, зведена теорема про зв'язок оптимальних рішень пари двоїстих задач, та зв'язок обмежень прямої задачі з оптимальним розв'язком двоїстої задачі і навпаки.

Двоїстий симплекс-метод. Теоретичні основи двоїстого симплекс-метода. Псевдоплан та його властивості. Алгоритм двоїстого симплекс-метода. Ознаки оптимальності плану та нерозв'язності задачі ЛП. Порівняльний аналіз двоїстого симплекс-метода з симплекс-методом.

Дослідження моделей задач лінійного програмування на чутливість. Економічна інтерпретація оптимальних значень двоїстих змінних. Дослідження чутливості оптимального розв'язку задач ЛП при варіюванні обмежених ресурсів.

Дослідження чутливості при варіюванні матриці обмежень задачі. Дослідження чутливості при введенні нового способу виробництва.

Змістовна постановка та математична модель транспортної задачі. Умови розв'язності транспортної задачі. Опорні плани транспортної задачі та методи їх знаходження: метод північно-західного кута та мінімального елемента.

Методи знаходження оптимального розв'язку транспортної задачі. Алгоритм методу потенціалів. Ознака оптимальності плану транспортної задачі.

Постановка та математична модель задачі лінійного цілочисельного програмування (ЛЦП). Особливості задач ЛЦП. Алгоритм методу відсікаючих площин Гоморі. Ознаки оптимальності розв'язку та нерозв'язності задачі лінійного цілочисельного програмування при використанні методу Гоморі.

Постановка та математична модель задачі лінійного цілочисельного програмування (ЛЦП).Загальна схема методу гілок та меж. Основні процедури методу. Ознака оптимальності. Алгоритм методу гілок та меж для задачі ЛЦП. Процедура розбиття множини на підмножини варіантів та знаходження оцінок в задачі ЛЦП. Ознака оптимальності. Приклад застосування.

Постановка та математична модель задачі комівояжера. Опис алгоритму методу гілок та меж в задачі комівояжера. Ознака оптимальності.

Постановка та математична модель задач нелінійного програмування (НП) і дослідження структури. Визначення опуклої та увігнутої функції.

Теорема про розташування оптимальних розв'язків задачі НП. Визначення опуклої та увігнутої функції. Ознаки опуклості та вгнутості. Алгоритм класичного методу умовної оптимізації.

Постановка та математична модель задач нелінійного програмування Теорема про множники Лагранжа Метод множників Лагранжа.

Теорема Куна-Такера та її роль в нелінійному програмуванні (НП). Застосування теореми Куна-Таккера для задачі опуклого програмування.

Методи знаходження оптимальних рішень задач нелінійного

програмування. Класичний метод пошуку умовного екстремуму. Метод змінної метрики. Прямі методи пошуку.

Задача квадратичного програмування та метод її розв'язання. Форма запису задачі. Умови оптимальності Куна-Таккера для задач квадратичного програмування.

Класифікація методів пошуку екстремуму нелінійних функцій. Градієнтні методи. Метод спряжених напрямків.

Загальна постановка задачі для методів одновимірної оптимізації. Математична модель задачі. Алгоритм дихотомічного пошуку.

Прямі методи одновимірного пошуку. Постановка та математична модель. Метод золотого перетину.

Прямі методи одновимірного пошуку. Постановка та математична модель. Метод Фібоначчі.

Загальна характеристика методів пошуку екстремуму нелінійних функцій при обмеженнях. Порівняння методів лінійного пошуку.

Загальна характеристика методів можливих напрямків. Можливий напрямок та можливий напрямок спуску. Застосування методів можливих напрямків у випадку лінійних обмежень.

Побудова можливих напрямків спуску. Застосування методів можливих напрямків у випадку нелінійних обмежень.

Загальна характеристика методів штрафних функцій. Алгоритм методу бар'єрних поверхонь. Алгоритм методу штрафних функцій.

Список літератури [1 — 9]

2. Навчальна дисципліна «Методи та системи штучного інтелекту»

Інтелектуальні алгоритми пошуку

Формалізація постановки задачі в просторі станів. Стратегії сліпого пошуку. Ітераційне поглиблення. Особливості, переваги і недоліки цих стратегій. Функції, які спрямовують пошук. Класифікація методів пошуку за стратегіями обходу графа простору станів. Стратегії евристичного пошуку hill-climbing, best-first search, A*. Характеристики оцінювальної функції: монотонність, допустимість, інформативність. A*-алгоритм евристичного пошуку. Теорема допустимості.

Генетичні алгоритми (ГА)

Відмінність ГА від традиційних оптимізаційних методів, ідея ГА, оператори ГА, способи підбору батьківської пари, принцип елітизму та їх вплив на ефективність роботи алгоритму.

Інтелектуальні інформаційні технології обробки сигналів

Методи інтелектуального видобування корисної інформації зі спотворених даних. Інтелектуальні методи оцінювання хаотичності часових рядів.

Нечітка математика

Загальні поняття (нечітка множина та функція приналежності. Способи

представлення та характеристики нечітких множин). Операції над нечіткими множинами. Поняття нечіткої змінної. Поняття лінгвістичної змінної.

Розпізнавання образів

Типи завдань розпізнавання образів. Пред'явлення навчальної множини образів. Правило розпізнавання: паралельна та послідовна процедура формування правила. Варіанти опису об'єктів розпізнавання. Розпізнавання обличь – особливості в порівнянні з іншими задачами розпізнавання.

Нейромережі

Біологічний нейрон та нейромережа. Модель та компоненти штучного нейрону Маккалока та Пітса. Задачі, які вирішуються за допомогою штучних нейромереж. Типові архітектури штучних нейромереж.

Експертні системи

Архітектура типової експертної системи. Прямий механізм висновку при побудові ЕС. Зворотній механізм висновку при побудові ЕС. Підходи при формуванні правил: навчання на базі досвіду, та на базі моделі досліджуваного об'єкту.

Список літератури [10 — 19]

3. Навчальна дисципліна «Об'єктно-орієнтоване програмування»

Основні принципи ОО-програмування та схеми реалізації програм на базі простої моделі ОО-мови. ОО-програма представляє собою сценарій взаємодії сукупності об'єктів, що поєднують дані та операції над ними в єдину сутність і є представниками своїх класів, причому останні можуть бути пов'язані в ієрархії наслідування та композиції. Іспит передбачає перевірку знань основних принципів ООП: інкапсуляція, наслідування, віртуальні методи та поліморфізм, графічна мова опису класів та їх ієрархії UML (оцінюється коректність напрямку та символіки позначення базових відношень між класами: композиції/агрегації та наслідування).

Також перевіряються знання основ ОО-мови C++: базовий синтаксис, методи, їх перевантаження, абстрактні класи, відкрите наслідування, робота з текстовими рядками, консольне введення-виведення, вказівники та посилання.

Список літератури [20 — 24]

4. Навчальна дисципліна «Інтелектуальний аналіз даних»

1. Етапи інтелектуального аналізу даних.
2. Характеристика типів аналізу, загальне та відмінності у постановках завдань кластерного, регресійного, дискримінатного та факторного аналізів, проблеми кожного з типів аналізу.

РЕГРЕСІЙНИЙ АНАЛІЗ

3. Завдання та проблеми регресійного аналізу (параметричний синтез).
4. Постановка задачі конструювання регресійних моделей.
5. Виведення матричної формули МНК для параметрів лінійної регресії з умови мінімуму квадратів відхилень моделі та об’єкту.
6. Виведення матричної формули розрахунку параметрів регресійної моделі із властивості проективності методу найменших квадратів.
7. Оператор проектування у методі найменших квадратів. Властивості оператору (симетричність та ідемпотентність), доведення властивостей.
8. Геометрична інтерпретація методу найменших квадратів у просторі змінних та у просторі об'єктів.
9. Алгоритми крокової регресії.
10. Бінарна логістична регресія.
11. Критерії оцінки якості регресійної моделі. Кількісна та якісна адекватність моделі.

КЛАСТЕРНИЙ АНАЛІЗ

12. Особливості навчання без вчителя. Задача кластеризації.
13. Завдання та проблеми кластерного аналізу.
14. Суть та характеристика етапів кластерного аналізу.
15. Класифікація та характеристика методів кластерного аналізу даних.
16. Загальне та відмінності кластерного аналізу від задач класифікації з вчителем.
17. Міри відстаней, що застосовуються в кластерному аналізі.
18. Критерії оцінки якості кластеризації.
19. Ієрархічний кластерний аналіз. Властивості кластерів, що утворюються. Переваги та недоліки підходу. Специфіка застосування.
20. Правила групування кластерів. Наведіть аналіз одного з методів.
21. Неієрархічні методи кластерного аналізу. Метод k-means.
22. Порівняльний аналіз ієрархічних і неієрархічних методів кластеризації
23. Нечіткі алгоритми кластеризації.

ФАКТОРНИЙ АНАЛІЗ

24. Характеристика, цілі, передумови, завдання, етапи факторного аналізу.
25. Метод головних компонент, етапи методу.
26. Градієнтний підхід до вирішення задачі компонентного аналізу.
27. Розв'язання задачі компонентного аналізу за допомогою спектральної теореми.
28. Етап МГК - "Виділення головних компонентів". Критерій Кайзера та Кеттелла.
29. Оцінка ефективності процедури методу головних компонент.
30. Геометрична інтерпретація початкового факторного рішення.
31. Другий етап факторного аналізу, цілі та завдання етапу.
32. Процедури обертання у факторному аналізі.
33. Критерії доцільності процедури факторного аналізу, критерій Кайзера-Мейєра-Олкіна, міра вибіркової адекватності (коефіцієнт MSA).
34. Доцільність існування різних способів обертання початкового факторного

рішення у факторному аналізі. Пояснення методу Варімакс.
35. Доцільність існування різних способів обертання початкового факторного рішення у факторному аналізі. Пояснення методу Квартимакс.
36. Висновки та результати факторного аналізу

ДИСКРИМІНАНТНИЙ АНАЛІЗ

37. Сутність і завдання дискримінантного аналізу.
38. Обґрунтування використання мір відстані, що використовуються у нормальному дискримінантному аналізі (зважена Евклідова та Махалонобіса).
39. Нормальний дискримінантний аналіз. Прості дискримінантні функції Фішера. Розрахунок параметрів. Геометрична інтерпретація.
40. Нормальний дискримінантний аналіз. Алгоритм Stepwise структурного синтезу дискримінантних функцій.
41. Нормальний дискримінантний аналіз. Обґрунтування критеріїв крокового алгоритму.
42. Висновки та результати нормального дискримінантного аналізу

КАНОНІЧНИЙ ДИСКРИМІНАНТНИЙ АНАЛІЗ

43. Обґрунтування канонічного дискримінантного аналізу.
44. Принципи побудови канонічних дискримінантних функцій. Пояснення розмірності простору канонічних дискримінаційних функцій.
45. Загальне та відмінності у постановці та у принципі вирішення завдань канонічного дискримінантного та факторного аналізу.
46. Висновки та результати канонічного дискримінантного аналізу.

Список літератури [25 — 30]

ПРИКІНЦЕВІ ПОЛОЖЕННЯ

Користування допоміжним матеріалом на екзамені
— Дозволяється використання інженерних калькуляторів.

Критерії оцінювання (за системою ECTS, стобальна шкала)

Розв’язання кожної задачі оцінюється за такими критеріями:

95—100	—	задачу розв’язано повністю, вірно
85—94	—	задачу розв’язано вірно, відповідь правильна, але наявними є один-два недоліки (наявними є деякі методичні помилки, порушено послідовність викладок тощо)
75—84	—	задачу розв’язано вірно, але відповідь неправильна (наявними є арифметичні помилки)
65—74	—	задачу розв’язано неповністю, але намічено правильний хід розв’язування
60—64	—	задачу не розв’язано, але наведено формули або твердження, що можуть бути використані при розв’язуванні задачі
менше 60	—	задачу не розв’язано

Результат роботи обчислюється як середнє арифметичне оцінок, що їх отримано за кожну задачу, і заокруглюється до цілих.

Оскільки «Правила прийому до КПІ ім. Ігоря Сікорського в 2023 році» вимагають при обчисленні конкурсного балу застосування шкали оцінювання 100…200 балів, виконується перерахунок оцінки рейтингової системи оцінювання в шкалу єдиного вступного іспиту з іноземної мови відповідно до Таблиці відповідності оцінок.

Таблиця відповідності оцінок РСО (60…100 балів)
оцінкам 200-бальної шкали (100…200 балів)

шкала РСО	шкала 100…200		шкала РСО	шкала 100…200		шкала РСО	шкала 100…200		шкала РСО	шкала 100…200
60	100		70	140		80	160		90	180
61	105		71	142		81	162		91	182
62	110		72	144		82	164		92	184
63	115		73	146		83	166		93	186
64	120		74	148		84	168		94	188
65	125		75	150		85	170		95	190
66	128		76	152		86	172		96	192
67	131		77	154		87	174		97	194
68	134		78	156		88	176		98	196
69	137		79	158		89	178		99	198
									100	200

Приклади типових завдань комплексного фахового випробування

1. Навчальна дисципліна «Дослідження операцій»

Приклад 1. Постановка та математична модель багатокритеріальної задачі прийняття рішень і її властивості. Властивості ефективних альтернатив і способи їх знаходження.

Розв'язок:

Математична модель багатокритеріальної задачі:

$\max f_i(x),\ \forall i \in I_1,\ I_1 = \{1,2,\ldots,m\}$ (1)

$\min f_i(x), \forall i \in I_2\ \ I_2 = \{m+1, m+2,\ldots,M\}$ (2)

Введемо такі відношення на множині альтернатив при наявності множини цільових функцій (критеріїв):

а) слабкої переваги (не гірше) $\succeq$: казатимемо, що $x_1 \succeq x_2$, коли

$$\begin{cases} f_i(x_1) \geq f_i(x_2), & \forall i \in I_1, \\ f_i(x_1) \leq f_i(x_2), & \forall i \in I_2 \end{cases} \qquad (3)$$

б) строгої переваги (краще): $x_1 \succ x_2$, тоді і тільки тоді, коли система нерівностей (3) виконується і хоча б одна з них – строго;

в) сильної переваги: $x_1 \succ\succ x_2$, якщо всі нерівності (3) – строгі;

г) еквівалентності: $x_1 \sim x_2$ тоді і тільки тоді, коли $f_i(x_1) = f_i(x_2) \quad \forall i \in I$

Альтернатива x_0 *зветься ефективною, якщо на множині допустимих альтернатив A не існує такої альтернативи* $\hat{x}$, *для якої б виконувалися нерівності:*

$$f_i(\hat{x}) \geq f_i(x_0), \quad \forall i \in I_1, \qquad (4)$$

$$f_i(\hat{x}) \leq f_i(x_0), \quad \forall i \in I_2 \qquad (5)$$

і хоча б одна з них була строгою. Це означає, що ніяка інша альтернатива не може поліпшити значення деякої цільової функції порівняно з ефективною альтернативою, не погіршуючи при цьому хоча б одну з інших цільових функцій.

Тому іноді ефективну альтернативу називають *не поліпшуваною на множині ц. ф.* або *оптимальною за Парето*.

Найкращим рішенням слід вважати таку альтернативу x, при якій відхилення від оптимальних значень для кожної цільової функції $f_i(x)$ досягає свого мінімального значення.

$$\Delta f_i(x) = \begin{cases} f_i^0 - f_i(x), \forall i \in I_1; \\ f_i(x) - f_i^0, \forall i \in I_2 \end{cases} \qquad (6)$$

Тут через f_i^0 позначено оптимальне значення *i*-ї ц. ф. f_i на множині допустимих альтернатив.

Оскільки ц. ф. мають різну розмірність, то потрібно ввести деяке

перетворення $w_i(f_i(x))$, що приведе $f_i(x)$ до безрозмірного вигляду. Це перетворення має задовольняти такі вимоги:

а) врахувати необхідність мінімізації відхилень від оптимальних значень для кожної цільової функції;

б) мати спільний початок і один порядок змінювання значень на множині допустимих альтернатив;

в) зберігати відношення переваги на множині альтернатив, що порівнюються для сукупності цільових функцій $f_i(x)$, і завдяки цьому не змінювати множини ефективних альтернатив.

$$w_i(f_i(x)) = \begin{cases} \dfrac{f_i^0 - f_i(x)}{f_i^0 - f_{i\min}}, \forall i \in I_1; \\ \dfrac{f_i(x) - f_i^0}{f_{i\max} - f_i^0}, \forall i \in I_2; \end{cases} \qquad (7)$$

де $f_{i\min}$, $f_{i\max}$ – відповідно найменші значення функцій, які максимізуються і найбільші значення функцій, що мінімізуються, на множині допустимих альтернатив, в перетвореннях (7) величини $w_i(f_i(x))$ завжди знаходяться в межах інтервалу $[0,1]$.

Метод обмежень для пошуку компромісних рішень ґрунтується на такій теоремі.

ТЕОРЕМА. Для того, щоб альтернатива була ефективною при заданому векторі переваг $\rho \geq 0$, достатньо, щоб x^* був єдиним розв'язком системи нерівностей

$$\rho_i w_i(x) \leq k_0, \forall i \in I \qquad (8)$$

для мінімального значення параметра k_0^*, при якому ця система сумісна.

Для знаходження компромісної альтернативи будується ітераційний процес з параметром $k_0 \in (0; \frac{1}{M})$, на кожному кроці якого перевіряється сумісність системи нерівностей (8). Зменшуючи параметр k_0 і тим самим зменшуючи зважені втрати для всіх ц. ф., наближаємося до альтернативи, що забезпечує мінімальні втрати по всіх ц. ф., тобто до компромісної альтернативи.

$$\min k_0 \qquad (9)$$

$$\rho_i w_i(x) \leq k_0, \forall i \in I \qquad (10)$$

де $w_i(x)$ – задається перетвореннями (7).

$$f_i(x) \geq f_i^* = f_i^0 - \frac{k_0}{\rho_i}(f_i^0 - f_{i\min}), \forall i \in I_1, \qquad (11)$$

$$f_i(x) \le f_i^* = f_i^0 + \frac{k_0}{\rho_i}(f_{i\max} - f_i^0), \forall i \in I_2 . (12)$$

Розв‘язок задачі (9) – (10) при мінімально можливому значенні параметра $k_0 \in (0; \frac{1}{M})$ визначить шукану компромісну альтернативу.

Приклад 2. Двоїста задача лінійного програмування. Зв’язок між формами запису прямої та двоїстої задач ЛП, основні теореми двоїстості, зведена теорема про зв’язок оптимальних рішень пари двоїстих задач, та зв’язок обмежень прямої задачі з оптимальним розв’язком двоїстої задачі і навпаки.

Розв’язок:

Запишемо пряму і двоїсту задачі в загальному вигляді.

Максимізувати $\sum_{j=1}^{n} c_j x_j$ (1)

при обмеженнях

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i, i = 1,2,\ldots,m ; \quad (2)$$

$x_j \ge 0$, j = 1, 2, …, n. (3)

Тоді двоїста задача записується так:

мінімізувати $\sum_{i=1}^{m} b_i y_i$ (4)

при обмеженнях

$$\sum_{i=1}^{m} a_{ij} y_i \ge c_j, j = 1,2,\ldots,n \quad (5)$$

$y_i \ge 0$; i=1, 2, …, m. (6)

Порівнюючи форми запису прямої та двоїстої задач, можна встановити між ними такі взаємозв’язки.

1. Якщо пряма задача є задачею максимізації, то двоїста буде задачею мінімізації, і навпаки.

2. Коефіцієнти цільової функції прямої задачі $c_1,\ldots,c_n$ стають вільними членами обмежень двоїстої задачі.

3. Вільні члени обмежень прямої задачі $b_1,\ldots,b_n$ стають коефіцієнтами цільової функції двоїстої задачі.

4. Матриця обмежень двоїстої задачі утворюється шляхом транспортування матриці обмежень прямої задачі.

5. Число обмежень прямої задачі дорівнює числу змінних двоїстої задачі і навпаки.

Теорема 1. Якщо x та y – допустимі розв‘язки прямої та двоїстої задач, тобто $Ax \le b$, та $A^T y \ge c$ то $C^T X \le B^T У$ (7).

Теорема 2 (основна теорема двоїстості). Якщо $\mathbf{x}_0$ та $\mathbf{y}_0$ – допустимі розв’язки прямої і двоїстої задач і, крім того, якщо $\mathbf{c}^T\mathbf{x}_0 = \mathbf{b}^T\mathbf{y}_0$ (8), то $\mathbf{x}_0$ та $\mathbf{y}_0$ – оптимальні

розв'язки пари двоїстих задач.

Теорема 3. Якщо в оптимальному розв'язку прямої задачі і-те обмеження виконується як строга нерівність, то оптимальне значення відповідної двоїстої змінної дорівнює нулю, тобто

$$\sum_{j=1}^{n} a_{ij} x_{j\,onm} = \mathrm{A}^{i}\mathrm{x}_{onm} < b_i, \; mo \; y_{i\,onm} = 0,$$

Якщо в оптимальному розв'язку двоїстої задачі обмеження j виконується як строга нерівність, то оптимальне значення відповідної змінної прямої задачі має дорівнювати нулю, тобто

якщо $A^T_j y_{опт} - c_j > 0$, то $x_{j\,опт} = 0$.

2. Навчальна дисципліна «Методи та системи штучного інтелекту»

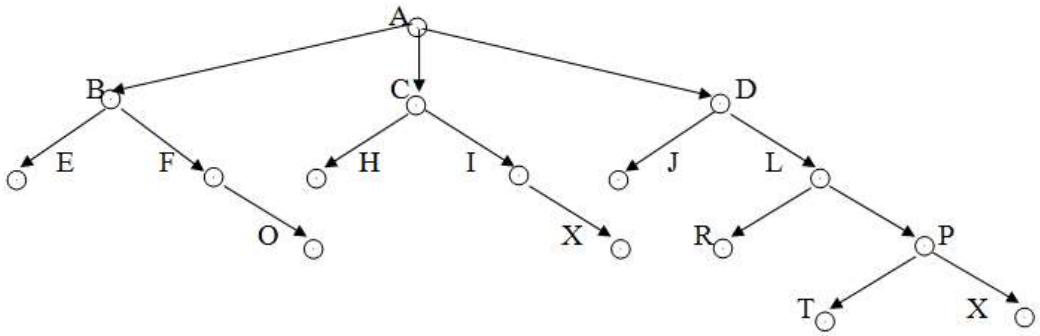
Приклад 1. Визначити для наступного графу:

- послідовність вершин, які будуть перевірятися при визначення маршруту між вершинами A і X за методом „Підіймання на пагорб" (Hill climbing);
- маршрут, який першим буде винайдено за цим методом.

Оцінювальна функція вершини *x*:

$$h(x) = 1/N,$$

де N – номер в латинському алфавіті букви, якою позначена вершина графа.



Розв'язок:

1) Поточна вершина A, вершини-спадкоємці: B, C, D.

$$1/N_D < 1/N_C < 1/N_B,$$

обрана вершина переходу - D.

2) Поточна вершина D, вершини-спадкоємці: J, L.

$$1/N_L < 1/N_J,$$

обрана вершина переходу - L.

3) Поточна вершина L, вершини-спадкоємці: P,R.

$$1/N_R < 1/N_P,$$

обрана вершина переходу - R.

4) Поточна вершина R, вершин-спадкоємців немає.

Повернення до попередньої вершини L, вершина-спадкоємець: P.

Обрана вершина переходу - P.

5) Поточна вершина P, вершини-спадкоємці: T,X.

$$1/N_X < 1/N_T,$$

обрана вершина переходу - X.

6) Поточна вершина X - цільова вершина.

Відповідь:

- послідовність вершин, які перевірялись:

A, D, L, R, P, X,

- маршрут, який першим винайдено:

A, D, L, P, X.

Приклад 2. За допомогою нечітких множин: 1) описати дві нечіткі змінні: «Середній прибуток» та «Малий прибуток»; 2) за допомогою операцій над нечіткими множинами утворити нову нечітку змінну: «прибуток, який можна вважати малим та середнім одночасно».

Розв'язок:

Нехай прибуток компанії в умовних одиницях: 1,2,3,4….

1) Утворимо довільні нечіткі множини, що описують поняття «малий прибуток» та «середній прибуток». В кожному елементі множини перше число – прибуток, друге – ступінь приналежності даного елемента нечіткій множині.

A (малий прибуток) = {<1,1> <2,1> <3,0.8> <4,0.6> <5,0.4> <6,0.3> <7,0.2> <8,0.1><9,0><10,0>};

B (середній прибуток) = {<1,0><2,0><3,0.1><4,0.3> <5,0.5><6,0.7><7,0.8><8.0,9><9,1><10,1>}

2) Для створення нової нечіткої змінної C «малий ТА середній прибуток» використовується операція перетину нечітких множин, формула приналежності якої розраховується по формулі: $\mu C = \min(\mu A, \mu B)$. Отримуємо множину:

C (малий та середній) прибуток = {<1,0><2,0><3,0.1><4,0.3><5,0.4><6,0.3><7.0,2><8,0.1><9,0><10,0>}

3. Навчальна дисципліна «Об'єктно-орієнтоване програмування»

Приклад 1. Нижче наведено код, в якому описуються класи для реалізації умовної "програми-дієтолога" (автоматичний підрахунок калорій залежно від кількості вжитих продуктів та способів їх приготування). В коді бракує кількох методів. Ієрархія наслідування класів страв (англ. dish) включає "просту страву" (BasicDish) з одного продукту та "складну (комплексну) страву" (ComplexDish), до складу якої входять інші страви. До інтерфейсу страви входять операції зі зміни ваги та питомої калорійності на 100 г. продукту, а також операції отримання ваги та загальної калорійності страви. Реалізовано один спосіб приготування страв: "сковорідка" (англ. pan) для смаження страв (нехай вона збільшує калорійність страви в півтора рази та зменшує вагу на 20%) та "аналізатор вмісту" для виведення на екран звіту з калорійності простої страви та комплексної страви з її складовими.

Завдання:

1) Намалювати **UML діаграму**, що включає усі класи в коді, за виключенням стандартних (таких, як std::string - їх можна вважати просто “типами даних”). Блок класу має містити лише назву класу, перелік його членів можна не наводити.

2) Написати **код двох пропущених методів** для зчитування та зміни ваги для класу ComplexDish (вага має змінюватися пропорційно для всіх складових, тобто, якщо вага усієї страви збільшилась удвічі, то це означає, що вага кожної складової має теж збільшитись удвічі).

3) Написати, що буде **виведено на екран** в процесі виконання програми.

```
#include <iostream>
#include <string>
#include <list>

class IDish {
    public:
        virtual std::string getDescription() = 0;
        virtual float getWeight() = 0;
        virtual float getCalories() = 0;
        virtual void setWeight(float w) = 0;
        virtual void setCaloriesPerWeight(float cpw) = 0;
};

class AbstractNamedDish: public IDish {
    public:
        AbstractNamedDish(std::string desc):
            description(desc) {
        }
        virtual std::string getDescription() override {
            return description;
        }
    private:
        std::string description;
};

class BasicDish: public AbstractNamedDish {
    public:
        BasicDish(std::string desc, float w, float cpw):
            AbstractNamedDish(desc), weight(w), caloriesPerWeight(cpw){
        }
        virtual float getWeight() override {
            return weight;
        }
        virtual float getCalories() override {
            return caloriesPerWeight * weight / 100;
        }
        virtual void setWeight(float w) override {
            weight = w;
        }
        virtual void setCaloriesPerWeight(float cpw) override {
            caloriesPerWeight = cpw;
        }
    private:
        float weight;
        float caloriesPerWeight;
};
```

```
class ComplexDish: public AbstractNamedDish {
    public:
        ComplexDish(std::string desc):
            AbstractNamedDish(desc) {
        }
        void addDish(IDish* sub_dish) {
            dishes.push_back(sub_dish);
        }
        std::list<IDish*>& dishList() {
            return dishes;
        }
        virtual float getCalories() override {
            float cal = 0;
            for (auto sub_dish: dishes) {
                cal += sub_dish->getCalories();
            }
            return cal;
        }
        virtual void setCaloriesPerWeight(float cpw) override {
            float cpwKoef = cpw / (getCalories() / getWeight());
            for (auto sub_dish: dishes) {
                sub_dish->setCaloriesPerWeight(sub_dish->getCalories()
                     / sub_dish->getWeight() * cpwKoef);
            }
        }
    private:
        std::list<IDish*> dishes;
};

class Pan {
    public:
        IDish* fry(IDish* dish) {
            dish->setCaloriesPerWeight(1.5 * dish->getCalories()
                 / (dish->getWeight() / 100));
            dish->setWeight(0.8 * dish->getWeight());
            return dish;
        }
};

class DishAnalyzer {
    public:
        void inspect(BasicDish* dish) {
            std::cout << dish->getDescription()
                << " has total " << dish->getCalories() << " kcal"
                << " in " << dish->getWeight() << " g" << std::endl;
        }
        void inspect(ComplexDish* dish) {
            inspect((BasicDish*) dish);
            std::cout << "including:" << std::endl;
            for (auto sub_dish: dish->dishList()) {
                inspect((BasicDish*) sub_dish);
            }
        }
};

int main() {
    BasicDish* meat = new BasicDish("chicken", 200, 200);
    BasicDish* veg = new BasicDish("potato", 100, 50);
    ComplexDish* dinner = new ComplexDish("fried chicken & potatoes");
    dinner->addDish(meat);
    dinner->addDish(veg);
```

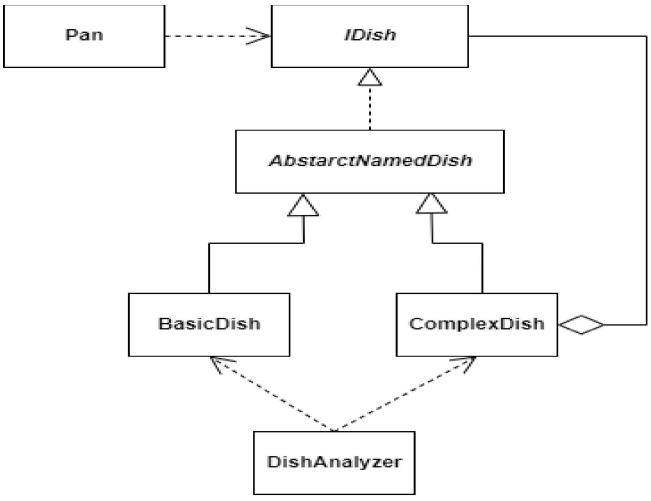
```
    Pan pan;
    DishAnalyzer analyzer;

    analyzer.inspect(meat);
    analyzer.inspect(veg);
    pan.fry(veg);
    pan.fry(dinner);
    analyzer.inspect(dinner);
}
```

Розв’язок.

[Успішне виконання завдання передбачає володіння базовими навичками написання класів та створення об’єктів, знання принципів інкапсуляції, наслідування (успадкування), поліморфізму (реалізованого через віртуальні методи), а також нотації UML (а саме – правил побудови діаграми класів). Завдання не спрямоване на оцінку рівня володіння конкретною мовою програмування, і хоча код наведено на мові C++, але з уникненням специфічних лише для цієї мови конструкцій, що дає змогу зрозуміти код тим, хто надає перевагу ОО-мовам Java чи C#. Виключення складають: списки ініціалізації в конструкторі (йдуть між двокрапкою та тілом конструктора, стандартний для C++ спосіб ініціалізації полів в конструкторі, особливо - успадкованих), оголошення чистих віртуальних методів (без реалізації, закінчуються на “= 0”) як частини “інтерфейсних” абстрактних класів, конструкції потокового виведення на екран (std::cout). В коді не використовуються “розумні вказівники” та майже не використовуються посилання, але використовуються звичайні вказівники (позначені через символ “зірочка” та наближені до посилань в інших мовах), при цьому замість оператора “точка” для доступу до членів класу використовується оператор “стрілочка”. Також наявна (стандарт C++11) спрощена конструкція циклу по елементам контейнеру (англ. “range-based FOR loop”, з використанням ключового слова auto, аналогічна конструкціям типу “for each” у інших мовах). Код спрощено для збереження лише важливих для виконання завдання елементів, і він не є оптимальним ні з точки зору продуктивності, ані надійності чи стилю, але є робочим. У екзаменаційних завданнях не передбачається використання інших конструкцій мови C++, які не зустрічалися б у даному прикладі.]

1. Почнемо з побудови діаграми наявних класів. Головне тут правильно показати відношення між класами (агрегація, композиція, наслідування - всі вони “однонаправлені” з різними типами стрілок: від наслідника до базового класу, від компонента до класу-контейнера). В даному прикладі композиція між з використанням нестандартних класів відсутня, є лише агрегація (час життя складових комплексної страви не залежить від часу існування самої комплексної страви). Маємо:



Зауважимо, що на діаграмі показані також відношення типу “знає /

використовує” (пунктирна стрілка), що не є обов’язковими для зображення в рамках завдання.

2. З коду інтерфейсу IDish видно, що клас ComplexDish має реалізувати методи getWeight() та setWeight(). Для зчитування загальної ваги страви достатньо просумувати вагу кожної страви-складової, а для задання нової ваги слід вирахувати коефіцієнт зміни ваги та використати його при заданні нової ваги кожної складової. Код може бути, наприклад, таким (не забороняється використовувати іншу, більш довгу форму циклу з оголошенням змінної-ітератора, головне - реалізувати алгоритм пропорційної зміни ваги згідно завдання):

```
virtual float getWeight() override {
    float w = 0;
    for (auto sub_dish: dishes) {
        w += sub_dish->getWeight();
    }
    return w;
}

virtual void setWeight(float w) override {
    float weightKoef = w / getWeight();
    for (auto sub_dish: dishes) {
        sub_dish->setWeight(sub_dish->getWeight() * weightKoef);
    }
}
```

3. Для визначення, що буде виведено на екрані, доцільно почати аналіз з точки входу в програму – функції main(). Хід міркування може бути наступним. Важливо акуратно виконувати обчислення у вірній послідовності.

3.1. У перших двох рядках створено об’єкти базових страв для позначення “курячого м’яса” (meat, вага порції 200 г, питома калорійність на 100 г: 200 ккал) та “картоплі” (veg, 100 г та 50 ккал на 100 г відповідно):

meat.weight = 200 [г]
meat.caloriesPerWeight = 200 [ккал на 100 г]
veg.weight = 100 [г]
veg.caloriesPerWeight = 50 [ккал на 100 г]

3.2. Далі створюється комплексна страва - “вечеря” (dinner), до якої додаються два попередньо створені об’єкти-інгредієнти meat та veg (точніше - вказівники на них). Далі створюється об’єкт-сковорідка (pan) для смаження та аналізатор складу страви (analyzer).

3.3. Спочатку викликається метод inspect() “аналізатора”, куди передаються об’єкти базових страв. Аналізатор виводить на екран назву страви, її калорії та вагу за допомогою викликів відповідних методів класу BasicDish. З урахуванням того, що загальна калорійність є добутком питомої калорійності на 100 г на вагу у грамах, розділену на 100:

*meat.getCalories() = meat.caloriesPerWeight * meat.weight / 100 =*
*= 200 * 200 / 100 = 400 [ккал]*
*veg.getCalories() = veg.caloriesPerWeight * veg.weight / 100 =*
*= 50 * 100 / 100 = 50 [ккал]*

На екрані буде виведено:

```
chicken has total 400 kcal in 200 g
potato has total 50 kcal in 100 g
```

3.4. Далі повертаючись до функції main(), об’єкт veg (овоч - “картопля”) передається методу “смаження” fry() об’єкта-”сковорідки” pan. Реалізація цього методу збільшує питому калорійність об’єкта-картоплі у 1.5 рази, а вагу множить на 0.8:

*veg.caloriesPerWeight = 1.5 * veg.getCalories() / (veg.weight / 100) =*
*= 1.5 * 50 / (100 / 100) = 75 [ккал на 100 г],*
*veg.weight = 0.8 * veg.weight = 0.8 * 100 = 80 [г].*

3.5. Далі знов відбувається виклик методу “смаження” pan.fry(), але на цей раз туди передається об’єкт-”вечеря” dinner (комплексна страва), в якій зберігаються вказівники на щодно змінений об’єкт-”картоплю” (veg) та досі не змінений об’єкт-”м’ясо” (meat). При розрахунках слід зважати, що будуть поліморфно викликані віртуальні методи об’єкта типу ComplexDish*. Розберемо покроково:

а) розрахуємо аргумент (назвемо його arg) для
dish->setCaloriesPerWeight():
arg[dish->setCaloriesPerWeight()] =
*= 1.5 * dinner.getCalories() / (dinner.getWeight() / 100)*
тут:
dinner.getCalories() = meat.getCalories() + veg.getCalories()
*veg.getCalories() = veg.caloriesPerWeight * veg.weight / 100 =*
*= 75 * 80 / 100 = 60 [ккал]*
dinner.getCalories() = 400 + 60 = 460 [ккал]
dinner.getWeight() = meat.weight + veg.weight = 200 + 80 = 280 [г]
отже:
*arg[dish->setCaloriesPerWeight()] = 1.5 * 460 / (280 / 100) =*
= 246.429 [ккал на 100 г]
б) при виконанні dish->setCaloriesPerWeight(246.429) питома калорійність складових вечері будуе змінена наступним чином (має зрости у 1.5 рази):
cpwKoef = 246.429 / (dinner.getCalories() / dinner.getWeight()) =
= 246.429 / (460 / 280) = 150
*meat.caloriesPerWeight = meat.getCalories() / meat.weight **
** cpwKoef = 400 / 200 * 150 = 300 [ккал на 100 г]*
*veg.caloriesPerWeight = veg.getCalories() / veg.weight * cpwKoef =*
*= 60 / 80 * 150 = 112.5 [ккал на 100 г]*
в) розрахуємо аргумент для dish->setWeight():
*arg[dish->setWeight()] = 0.8 * dinner.getWeight() = 0.8 * 280 =*
= 224 [г]
г) при виконанні dish->setWeight(224) вага складових вечері буде змінена наступним чином (має змінитися у 0.8 раз):
weightKoef = 224 / dinner.getWeight() = 224 / 280 = 0.8
*meat.weight = meat.weight * weightKoef = 200 * 0.8 = 160 [г]*
*veg.weight = veg.weight * weightKoef = 80 * 0.8 = 64 [г]*

3.6. І, нарешті, в останньому рядку функції main() відбувається виклик методу inspect() об’єкта analyzer з аргументом dinner. Тобто, буде викликаний метод inspect() класу DishAnalyzer з арументом типу ComplexDish*, що вказує на об’єкт dinner. При цьому спочатку буде викликаний код методу inspect() класу DishAnalyzer з арументом типу BasicDish* (через оператор приведення типу).

dinner.getCalories() = meat.getCalories() + veg.getCalories()
*meat.getCalories() = meat.caloriesPerWeight * meat.weight / 100 =*
*= 300 * 160 / 100 = 480 [ккал]*
*veg.getCalories() = veg.caloriesPerWeight * veg.weight / 100 =*
*= 112.5 * 64 / 100 = 72 [ккал]*
dinner.getCalories() = 480 + 72 = 552 [ккал]
dinner.getWeight() = meat.weight + veg.weight = 160 + 64 = 224 [г]

На екрані буде (аналогічно п.3.3):

```
fried chicken and potatoes has total 552 kcal in 224 g
```

Далі додається рядок “including:” та надається аналогічна інформація по складовим:

```
chicken has total 480 kcal in 160 g
potato has total 72 kcal in 64 g
```

Відповідь:

Загалом буде виведено:

```
chicken has total 400 kcal in 200 g
potato has total 50 kcal in 100 g
fried chicken & potatoes has total 552 kcal in 224 g
including:
chicken has total 480 kcal in 160 g
potato has total 72 kcal in 64 g
```

4. Навчальна дисципліна «Інтелектуальний аналіз даних»

Приклад 1. Міри відстаней, що застосовуються в кластерному аналізі.

Розв’язок.

Для того, щоб визначити, до якого з кластерів доцільно віднести об’єкт, потрібно спочатку:

- скласти вектор характеристик для кожного об'єкта - як правило, це набір числових значень;

- як правило слід провести нормалізацію вектора, щоб всі компоненти давали однаковий внесок при розрахунку «відстані»;

- для кожної пари об'єктів вимірюється «відстань» між ними - ступінь схожості.

Потім використати одну з мір відстаней. Серед них:

Евклідова відстань

$$\rho(x, x') = \sqrt{\sum_{i}^{n} (x_i - x_i')^2}$$

Відстань Манхетеновська

$$\rho(x,x') = \sum_{i}^{n} | x_i - x_i' |$$

Ця відстань є середньою різницею по координатах. Здебільшого цей спосіб знаходження відстані призводить до таких же результатів, як і для звичайної відстані Евкліда. Однак для цього способу вплив окремих великих різниць (викидів) зменшується (бо вони не підносяться у квадрат).

Відстань Чебишева

$$\rho(x,x') = \max(| x_i - x_i' |)$$

Ця відстань може бути корисною, коли потрібно визначити два об'єкти як «різні», якщо вони відрізняються за якоюсь однією координатою.

Та інші.

СПИСОК ЛІТЕРАТУРИ

1. Зайченко Ю. П. Дослідження операцій./ Ю.П.Зайченко. -К.ЗАТ „Віпол”.-688.
2. Зайченко О.Ю. Дослідження операцій /Зайченко О.Ю., Зайченко Ю.П. — К: Видавничій дім «Слово», 2014. — 472 с.
3. Ладогубець В.В. Алгоритми параметричної оптимізації складних систем / Ладогубець В.В., Ладогубець Т.С., Ладогубець О.В. — К.: АВЕРС, 2006. — 139 с.
4. Безкровний О.І. Дослідження операцій і методи прийняття технічних рішень. /Безкровний О.І., Павленко В.І., Тимошенко. Університет "Україна",2019.-420с.
5. Меньшикова О.В. Дослідження операцій / Меньшикова О.В., Чмир О.Ю., Карабин О.О. –Львів : ЛДУ БЖД, 2019. — 196 с.
6. Бартіш М. Я. Дослідження операцій. Нелінійне програмування : підручник / М. Я. Бартіш, І. М. Дудзяний. – Львів : ЛНУ ім. Івана Франка, 2011. - 208 с.
7. Лисенко О. І. Математичні методи моделювання та оптимізації. Математичне програмування та дослідження операцій: підручник / О. І. Лисенко, О. М. Тачиніна. –К.НАУ,2017.-213с.
8. Hamdy A. Taha.Operations Research: An Introduction, 10th Edition. – 2017. – 848 p.
9. Wayne L. Winston. Operations Research: Applications and Algorithms, 4th Edition. ISBN: 9780534423605. – 2021.
10. Зайченко Ю.П. Основи проектування інтелектуальних систем. Навч. посібник. Київ. Слово. 2004. 352с.
11. Russell S., Norvig P. Artificial Intelligence: A Modern Approach, 4th US ed. Pearson, 2020. 1136 p. ISBN 978-0134610993.
12. Giarratano J., Riley G. Expert Systems: Principles and Programming, 4th ed. Course Technology, 2004. 288 p. ISBN-100534384471:
13. Luger G. Artificial Intelligence: Structures and Strategies for Complex Problem Solving, 6th ed. Addison-Wesley, 2008. ISBN-10: 0-321-54589-3

14.Рибальченко М.О., Єгоров О.П., Зворикін В.Б. Цифрова обробка сигналів. Навчальний посібник. – Дніпро: НМетАУ, 2018. – 79 с.
15. Дробик О. В. Цифрова обробка аудіо- та відеоінформації у мультимедійних системах: навчальний посібник / О. В. Дробик, В. В. Кідалов, В. В. Коваль, Б. Я. Костік, В. С. Лазебний, Г. М. Розорінов, Г. О. Сукач. – К.: Наукова думка, 2008. – 144 с.
16. Желдак Т.А., Коряшкіна Л.С., Ус С.А. Нечіткі множини в системах управління та прийняття рішень. Навчальний посібник. —М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». – Дніпро : НТУ «ДП», 2020. – 387 с. ISBN 978-960-350-726-2
17. Погребенник В.Д. Системи розпізнавання образів / Навч. посібник. – Львів: СПОЛОМ, 2007. – 170 с.
18. Кацадзе Т. Л. Експертні системи прийняття рішень в енергетиці: навч. посіб. / Т. Л. Кацадзе. – К.: ЛОГОС, 2014. – 173 с. – Бібліогр.: с. 167-173.
19. Субботін С. О. Нейронні мережі : теорія та практика: навч. посіб. / С. О. Субботін. – Житомир : Вид. О. О. Євенок, 2020. – 184 с
20. Васильєв О. Програмування на С++ в прикладах і задачах. – К.: “Ліра – К”. – 2017. – 382 с. – ISBN 9786177507412.
21. Stroustrup B. A Tour of C++ (C++ In Depth Series, 3rd ed.). – Addison-Wesley Professional. – 2022. – 320 p. – ISBN 9780136816485.
22. Фрімен Е., Робсон Е., Бейтс Б., Сієрра К. Head First. Патерни проєктування. – Х.:“Фабула”. – 2020. – 672 с. – ISBN 9786170961594
23. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. – Addison-Wesley. – 1995. – 395 p. – ISBN 0201633612
24. Booch G. Object-Oriented Analysis and Design with Applications. – Addison-Wesley Professional. – 2007. – 720 p. – ISBN 9780201895513
25. Основи теорії і практики інтелектуального аналізу даних у сфері кібербезпеки [Електронний ресурс] : навчальний посібник / Д. В. Ланде, І. Ю. Субач, Ю. Є. Бояринова ; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 4,54 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2018. – 300 с. Доступ: https://ela.kpi.ua/bitstream/123456789/45721/1/NP_Osnovy_teorii_intelekt_analizu.pdf
26. Басюк Т.М, Литвин В.В., Захарія Л.М., Кунанець Н.Е. Машинне навчання: навч. посіб. Львів: Новий Світ-2000, 2019. 315 с.
27. Інтелектуальний аналіз даних: Комп’ютерний практикум [Електронний ресурс] : навч. посіб. для студ. спеціальності 122 «Комп’ютерні науки та інформаційні технології», спеціалізацій «Інформаційні системи та технології проектування», «Системне проектування сервісів» / О. О. Сергеєв- Горчинський, Г. В. Іщенко ; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 1,72 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2018. – 73 с.: Іл. Доступ: https://ela.kpi.ua/bitstream/123456789/24971/1/Komp_prakt.pdf
28. Vannatta, R. A., & LaVenia, K. N., (2020). Linear Discriminant Analysis, In P. Atkinson, S. Delamont, A. Cernat, J.W. Sakshaug, & R.A. Williams (Eds.), SAGE Research Methods Foundations. https://doi.org/10.4135/9781526421036889610

29. Dhamnetiya, Deepak & Goel, Manish & Jha, Ravi & Shalini & Bhattacharyya, Krittika. (2022). How to Perform Discriminant Analysis in Medical Research? Explained with Illustrations. Journal of Laboratory Physicians. 14. 10.1055/s-0042-1747675. https://www.thieme-connect.com/products/ejournals/pdf/10.1055/s-0042-1747675.pdf
30. Factor Analysis (2014). Richard L. Gorsuch, Routledge, 464 Pages.

Розробники програми:

Зайченко О.Ю., д.т.н., проф., професор кафедри математичних методів системного аналізу

Олена ЗАЙЧЕНКО

Булах Б.В., к.т.н., доцент кафедри системного проектування

Богдан БУЛАХ

Шаповалова С.І., к.т.н., доц., доцент кафедри автоматизації проектування енергетичних процесів і систем

Світлана ШАПОВАЛОВА

Павлов В.А., к.т.н., доцент кафедри біомедичної кібернетики

Володимир ПАВЛОВ