ЗМІСТ

ПЕРЕДМОВА

Штучний інтелект (або скорочено ШІ) є абсолютно революційною технологією, яка вже зараз задає нові тренди у розвитку суспільства, глобально видозмінюючи сфери життєдіяльності людства. На сьогодні не існує єдиного однозначного визначення самого терміну «штучний інтелект». Наразі це вже більше філософська категорія, яка потребує досконалого вивчення та переосмислення. У той же час ШІ більшість людства все ж пов’язує з інформаційними технологіями, які створюються завдяки аналізу, розробці й вдосконаленню програм та систем, здатних виконувати завдання, що, зазвичай, притаманні людському інтелекту. Вивчення ШІ включає в себе широкий спектр дисциплін, таких як машинне навчання, обробка природної мови, комп'ютерний зір тощо.

Машинне навчання є однією з ключових галузей вивчення ШІ. Вона досліджує алгоритми та моделі, які дають комп'ютерам здатність навчатися на основі даних і вдосконалювати свою продуктивність з часом. Вивчення машинного навчання вимагає розуміння статистики, процесів оптимізації та парадигм програмування.

Обробка звуків природної мови є ще однією важливою областю досліджень, пов’язаною з ШІ. Зокрема тут розглядаються методи аналізу та розуміння людської мови для створення систем, які можуть взаємодіяти з користувачами за допомогою голосових команд.

Комп'ютерний зір досліджує методи і алгоритми для розпізнавання й аналізу візуальної інформації. Науковці, які працюють у цій сфері, займається розробкою програм, що можуть розпізнавати об'єкти, людей, місця на зображеннях або ж відео. Вивчення комп'ютерного зору включає в себе методи обробки зображень, використання нейронних мереж та інших технологій, призначених для програмування автоматизованих систем візуального розпізнавання.

Звичайно, це не повний перелік сфер досліджень, але й він дозволяє підтвердити важливість штучного інтелекту для сьогодення. Саме тому багато університетів та наукових установ по всьому світу приділяють цьому питанню значну увагу.

Запропонований лабораторний практикум з систем штучного інтелекту містить загальні теоретичні основи, опис порядку виконання лабораторних робіт, перелік завдань до них. Кожна лабораторна робота завершується набором контрольних питань.

Усі завдання виконуються за допомогою мови програмування Python та бібліотек, створених на її основі. Для розв’язання завдань з машинного навчання рекомендується застосовувати онлайн сервіс Google Colaboratory, як найбільш доступний хмарний інструмент для створення і навчання моделей штучного інтелекту.

Студентам дозволяється використовувати і інші IDE, але тоді результати виконання роботи мають бути завантажені і оформлені за допомогою системи контролю версій Git та розміщені у публічному репозиторії на GitHub.com

ЛАБОРАТОРНА РОБОТА 1. Синтаксис та основні конструкції мови програмування Python.

Теоретичні відомості

Python – це особлива мова програмування, яка суттєво відрізняється від C, C++, Java та Php, але залишається досить простою для освоєння. Вона керується низкою правил, які важливо дотримуватися при написанні програм. Це зроблено для підвищення ефективності сприйняття коду програмістом.

Розглянемо ключові рекомендації щодо правильного форматування коду.

1. Кінець рядка є кінцем інструкції[1].

2. Вкладені інструкції поєднуються у блоки за величиною відступів.

3. Вкладені інструкції мови Python пишуться у відповідності до одного і того ж зразка.

Дії з числами

Додавання: **`32 + 7`**

Віднімання: **`31 – 8`**

Множення: **`35 * 3`**

Ділення: **`24 / 3`**

Цілочисельне ділення: **`20 // 3`**

Остача від ділення: **`20 % 3`**

Піднесення до степеня: **`20 ** 2`**

Бітові операції

Бітове «або»: ***`x | y`***[2]

Бітове виключне «або»: ***`x ^ y`***

[1] Якщо декілька інструкцій записуються в одному рядку, то вони відділяються за допомогою крапки із комою.

[2] Варто пам’ятати, що *x* та y мають бути цілими.

Бітове «і»: ***x & y***

Зсув бітів уліво: ***x << y***

Зсув бітів управо: ***x >> y***

Інверсія бітів: ***~x***

Запис цілого числа у двійковій, вісімковій та шістнадцятковій системах числення

bin(число) – запис числа у двійковій системі.

hex(число) – запис числа у шістнадцятковій системі числення.

oct(число) – запис числа у вісімковій системі числення.

Бітові операції

У мові Python для обробки дійсних чисел передбачені ті ж самі дії, що й для цілих чисел. Важливо пам'ятати про такі неточності під час виконання таких операцій.

Приклад 1.1.

```
0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1
```

Результат виконання: 0.9999999999999

Комплексні числа

У Python також передбачені операції з комплексними числами.

Приклад 1.2.

```
x = complex(1,3)
print(x)
```

Результат виконання команди у консолі Python: 1 + 3j

З цього прикладу стає очевидним, що уявну частину позначають літерою **j**, а не, як у математиці, літерою **i**. Це зроблено з метою уникнення плутанини з змінною **i**, яка часто використовується для позначення ітерацій у циклі. У Python уже закладені такі дії над комплексними числами як додавання, віднімання, множення та ділення.

Крім того, Python надає можливість виділяти спряжене комплексне число (**`x.conjugate()`**), його дійсну частину (**`x.real`**), уявну частину (**`x.imag`**) та порівнювати два комплексних числа на рівність.

Послідовність виконання арифметичних дій

Порядок виконання арифметичних операцій з числами визначається наступним чином: першочерговий пріоритет має піднесення до степеня, за ним йде множення, ділення, цілочисельне ділення і визначення остачі від ділення. Додавання та віднімання мають найнижчий пріоритет. У випадку, якщо вираз містить дії з однаковим пріоритетом, вони виконуються зліва направо по черзі. Пріоритетність може бути змінена за допомогою використання круглих дужок.

Оголошення змінних

Python немає строгої типізації даних. Зокрема, коли змінній присвоюється значення, вона автоматично ініціалізується. Цей підхід до роботи зі змінними називається динамічною типізацією, і його використовують також JavaScript та PHP. Інтерпретатор Python автоматично визначає тип даних, коли змінна створюється та ініціалізується.

По суті змінні в Python є посиланнями на об’єкти і не несуть інформації про тип даних. Під час виконання операцій зі змінними, вони замінюються об’єктами, на які вони посилаються.

Приклад 1.3.

```
b = 0.25
b = 4 # видаляється об’єкт 0.25 і створюється 4
b = "four" # видаляється об’єкт 4 і створюється
           #"four"
b = [1,2,3,4] # видаляється об’єкт "four"
```

Дії, які продемонстровано у прикладі можливі завдяки зберігання у об’єкта лічильника активних посилань на нього. Якщо кількість посилань

стає рівною 0, об’єкт видаляється з пам’яті. Це означає, що вивільненим ресурсом може скористатися інша частина програми. Такий процес, який виконує інтерпретатор автоматично, відомий під назвою «збірка сміття», і дозволяє розробникам використовувати об’єкти без потреби управління пам’яттю.

При створенні імен до змінних варто дотримуватися таких правил.

1. Ім'я змінної повинно починатися з літери (**a-z, A-Z**) або символу підкреслення (**_**).

2. Після першого символу може йти будь-яка кількість букв, цифр або символів підкреслення.

3. Імена змінних регістрозалежні (тобто **"variable"**, **"Variable"** і **"VaRiAbLe"** – різні змінні).

4. Не можна використовувати зарезервовані слова Python як імена змінних (наприклад, **"if", "else", "while", "for"** тощо).

5. Зазвичай використовують стиль lower_case_with_underscores для імен змінних **("my_variable", "total_count", "user_name"**). Це необов'язково, але рекомендується для полегшення читабельності.

Логічні операції та операції порівняння

Для порівняння значень змінних використовуються вже знайомі з математики символи: **>** –більше, **<** – менше, **>=** – більше або дорівнює, і **<=** – менше або дорівнює. Після виконання таких порівнянь отримується результат у вигляді булевого значення (True або False).

Також присутні логічні операції: **and** – логічне «і», **or** – логічне «або», **not** – логічне заперечення. Завдяки ним є можливість поєднувати логічні операції у цілі вирази.

Приклад 1.4.

```
(7 > 4) and (6 < 8)
```

Результат виконання команди у консолі Python: `True`

```
(5 > 6) or (6 >= 8)
```

Результат виконання команди у консолі Python: `False`

Стандартні команди вводу та виведення даних у консолі

Зчитування інформації для роботи програми з командного рядка та виведення результатів виконання коду до нього для мови програмування Python типово використовуються функціїи **`input()`** та **`print()`**.

Приклад 1.5.

```
a = int(input ("Enter a")) # Змінна a посилається
                           #на об'єкт цілого типу
print("a =", a) # Виводиться рядок з значенням
```

Коментування у Python 3

Для того, що зробити створюваний код більш доступним передбачені як багаторядкові так і однорядкові коментарі. Однорядковий коментар задається символом **#** і будь-який текст після нього не буде сприйматися інтерпретатором мови.

Приклад 1.6.

```
#Коментар у Python
"""Багаторядковий коментар у Python 3"""
```

Оператор умови if, else, elif. Блоки команд, відступи

Умовний оператор у мові програмування Python має такий вигляд.

```
if умова:
    інструкція 1 # Перша серія команд
    інструкція 2
else:
    інструкція 3 # Друга серія команд
    інструкція 4
```

Допустимий скорочений запис.

```
if умова:
    інструкція 1 # Перша серія команд
    інструкція 2
```

Для заміни команди **`switch`** у таких мовах як C++ або ж JavaScript застосовують розширенням умовного оператора командою **`elif`**.

```
if умова 1:
```

```
    інструкція 1 # Перша серія команд
    інструкція 2
elif умова 2:
    інструкція 3 # Друга серія команд
    інструкція 4
elif умова 3:
    інструкція 5 # Третя серія команд
    інструкція 6
else:
    інструкція 7
```

Цикли while та for

Команда **while** є циклом з передумовою та вважається одним з самих універсальних інструментів у програмування. Він виконується доти поки умова, яка вказана після службового слова істинна.

```
while умова:
    інструкція 1 # Серія команд
    інструкція 2
```

Цикл перебору **for** застосовують зазвичай для перебору елементів списку чи рядка, або ж інших інтерабельних структур даних.

Приклад 1.7.

```
for i in range(5):
    print(i)
```

Результат виконання програми у консолі:

```
0
1
2
3
4
```

Доречно зауважити, що для мови програмування не реалізовано інструкцію «цикл з післяумовою».

Важливим для роботи із циклами є команди **continue** та **break**. **continue** – дозволяє розпочати нову ітерацію циклу, не виконуючи

команди, які розміщені після зазначеної інструкції. Оператор **`break`** натомість достроково припиняє виконання циклу.

Завдання до лабораторної роботи

*Розв'яжіть задачі відповідно до Вашого варіанту (номер варіанту відповідає порядку запису задачі)**.** Розв'язок завантажте на відкритий репозиторій GitHub з назвою за зразком Прізвище_var_NN_Lab_1.*

Рівень 1[3]*.*

1. Обчислити висоту трикутника площею ***S***, якщо його основа більша за висоту на величину ***a***. Результат округлити до сотих.

2. Задано чотири точки паралелограма за допомогою координат його вершин. Визначити площу паралелограма та довжину його діагоналей. Результат округлити до тисячних.

3. Знайти площу бічної поверхні правильної чотирикутної піраміди об'ємом ***V*** і висотою ***h***. Результат представити з точністю до третього знаку після коми.

4. Обчислити площу чотирикутника, вершини якого задаються координатами на площині. Результат округлити до третього знаку після коми.

5. Обчислити площу повної поверхні зрізаного конуса, якщо відомо радіуси основ ***R***, ***r*** і висоти ***H***. Результат подати з точністю до тисячних.

Рівень 2[4]*.*

1. Вводиться натуральне число *n*. Чи ділиться воно одночасно на *a* і на *b* (числа *a* і *b* також вводяться користувачем)? Виведіть «YES» якщо *n* ділиться одночасно на *a* і на *b*. Виведіть «NO» в іншому випадку. [*eolymp.com, задача № 8531*]

---

[3] Задачі для першого рівня взято з посібника Мосіюк, О. О. *Штучний інтелект: вступ до машинного навчання.* Вид-во ЖДУ ім. І. Франка, м. Житомир, 2019. 76 с.

[4] Задачі до другого рівня взято з онлайн ресурсу *eolymp.com.*

2. Вводиться ціле число *n*. Виведіть, чи є воно додатним, від'ємним або дорівнює 0. Виведіть «Positive», «Negative» чи «Zero» в залежності від значення *n*. [*eolymp.com, задача № 8242*]

3. Знайдіть мінімум та максимум двох натуральних чисел, які вводяться у консоль. В одному рядку виведіть найменьше, а потім найбільше з двох чисел *a* і *b*. [*eolymp.com, задача № 2606*]

4. Визначіть назву пори року за заданим номером місяця (номер вводиться у консоль), використовуючи складені умови. Для весняних місяців виведіть «Spring», для літніх - «Summer», для осінніх - «Autumn» і для зимових - «Winter». [*eolymp.com, задача № 923*]

5. Визначіть тип трикутника (рівносторонній, рівнобедрений, різносторонній) за заданими довжинами його сторін. Виведіть 1, якщо трикутник рівносторонній, 2 — якщо рівнобедрений і 3 — якщо різносторонній. [*eolymp.com, задача № 905*]

Рівень 3[5].

1. Обрахувати значення *n*-того числа Фібоначчі, яке вводиться користувачем у консолі.

2. Вивести список всіх дільників числа *m* (число вводиться користувачем з консолі).

3. Реалізуйте серію з трьох ігор «Камінь, ножиці, папір» з комп’ютером. У фіналі виведіть статистику виграшів людини та машини.

4. Створть текстову гру «Вгадай число» в консолі, яка дає гравцю три спроби на відгадування числа, яке генерується з інтервала цілих чисел від 0 до 9 включно.

5. Згенерувати і вивести у консоль 10 серій із 0, 1 та 2 таких, що сума чисел була рівна 12.

[5] Окремі задачі для третього рівня взято з посібника Мосіюк, О. О. *Штучний інтелект: вступ до машинного навчання.* Вид-во ЖДУ ім. І. Франка, м. Житомир, 2019. 76 с.

Питання для самоперевірки

1. Розкрийте базові операції з простими числовими типами даних та вкажіть для них пріоритет дій.

2. Які особливості роботи з змінними у мові Python?

3. Команди введення та виведення даних у консолі.

4. Які особливості написання і форматування коду. Відступи.

5. Умовний оператор IF.

6. Оператори для описання циклів у мові програмування Python.

Довідкові матеріали

1. Офіційна сторінка Python. URL: https://www.python.org/

2. Програмування для всіх: основи Python. URL: https://prometheus.org.ua/course/course-v1:Michigan+PFE101+2023_T3

3. Python: Структури даних. URL: https://prometheus.org.ua/course/course-v1:Michigan+PDS101+2023_T3

4. Розробка та аналіз алгоритмів. Частина 1. URL: https://prometheus.org.ua/course/course-v1:KPI+Algorithms101+2015_Spring

Посилання на літературу

1. Алгоритми та структури даних. Навчальний посібник. Київ, НТУУ «КПІ ім. Ігоря Сікорського», 2022. 215 с.

2. Васильєв О. Програмування мовою Python. Навчальна книга – Богдан, 2019. 504 с.

3. Маттес Е. Пришвидшений курс Python. Львів, Видавництво Старого Лева, 2021. 600 с.

ЛАБОРАТОРНА РОБОТА 2. Опрацювання списків та рядків. Кортежі. Словники. Множини.

Теоретичні відомості

Рядки

Рядки є упорядкованими послідовностями символів. Для того щоб задати рядок у Python передбачено декілька способів: використання апострофів, звичних лапок та потрійно записаних подвійних лапок. При цьому рядок, який записаний за допомогою апострофів не відрізняється від рядка заданого за допомогою лапок.

Приклад 2.1.

```
S = 'Це рядок!' # Рядок задається апострофами
S = "Це також рядок" #Рядок задається лапками
```

Потрійний запис подвійних лапок дозволяє зберегти форматування.

Типові операції над рядками.

Об’єднання двох рядків

Приклад 2.2.

```
S1 = "ryad 1"
S2 = "ryad 2"
S = S1 + ' ' + S2
```

Результат.

```
"ryad 1 ryad 2"
```

Дублювання рядка.

Приклад 2.3.

```
S1 = "ryad"
print(3*S1)
```

Результат.

```
"ryadryadryad"
```

Отримання значення символу рядка відбувається за допомогою номера його індексу (індексація символів для рядків у мові

програмування Python розпочинається з 0). Доречим є зауваження, що змінити символ у рядку шляхом присвоєння нового символу, попередньо вказавши його індекс, не можна.

Приклад 2.4.

```
S = "ryad"
print(S[1])
```

Результат.

```
"y"
```

Створення зрізу символів заданого рядка має наступний вигляд **[X, Y]**, де **X** – це початок зрізу, **Y** – його кінець. При чому символ із індексом **Y** не входить у створюваний зріз. За замовчуванням перший індекс рівний 0, а другий – довжині рядка.

Приклад 2.5.

```
S = "ryad"
print(S[1,3])
```

Результат.

```
"ya"
```

Більше інформації про рядковий тип даних для мови програмування можна прочитати за посиланням.

https://docs.python.org/3/library/stdtypes.html#text-sequence-type-str

Списки

Список вважається упорядкована колекція об’єктів довільних типів, що можуть бути модифікованими у процесі виконання програми. Ключовою відмінністю від звичного масиву є те, що список може містити різні типи об’єктів, у той час як масив лише один. Окрім того, для мови програмування Python у класичному розумінні поняття масиву не має. З звичними числовими масивами у мові програмування Python можна працювати, підключивши, завчасно, модуль NumPy.

Приклад 2.6.

```
S = list('12345') #За допомогою команди list
S = [] #Створення порожнього списку
#Створення списку
#перерахунком елементів
lst = ['1', 'a', 3, ['343','a'], 2]
lst = [i for i in range(5)] # Створення списку за
                           #допомогою конструктора
```

Нумерація елементів списку починається з індексу 0.

Загалом, про команди для роботи з списками можна детально прочитати за посиланням.

https://docs.python.org/release/3.12.4/library/stdtypes.html#sequence-types-list-tuple-range

Кортежі

Кортежі представляють собою незмінні списки і при цьому займають менший обсяг пам'яті. При роботі з цим типом даних дозволяється використовувати всі функції і методи списків, які не змінюють його. Решта методів та функцій, які використовуються при роботі з списками, не застосовується, оскільки представлена структура є незмінюваною.

Приклад 2.7.

```
T = (1, 2, 3, 4, 5) # Перелічуємо у круглих дужках
                    # елементи кортежу.
T = tuple() #Створення порожнього кортежу
T = tuple(range(5)) #Створення кортежу із списку
```

Словники

Словники є неупорядкованими наборами різних об’єктів. Щоб звернутися до певного значення словника необхідно вказати спеціальний ключ, який однозначно визначає доступ до даних у зазначеній структурі.

Приклад 2.8.

```
D = {} # Задаємо порожній словник
D = {’key1’:25, ’key2’:42} # Словник задаємо
                           #перерахунком ключів
                           #та відповідним їм
                           #значеннями
D = dict.fromkeys([’key1’, ’key’])
# За допомогою
#цього  метода створюється словник з вказаними у
#списку ключами та значеннями None.
```

Більше про роботу з таким типом даних як **dict** у мові програмування Python можна дізнатися за посиланням.

https://docs.python.org/release/3.12.4/library/stdtypes.html#mapping-types-dict

Множини

Множини Python – це невпорядкована колекція окремих хешованих об'єктів. Така структура даних вже на базовому рівні підтримує типові математичні операції з множинами (об’єднання, перетин, різниця)

Приклад 2.9.

```
M = set() # Визначення порожньої множини за
допомогою
          # інструкції set
M = set(‘Hello’) #Перетворення рядка у множину за
```

```
                    #допомогою інструкції set
M = {1, 2, 3, 4} #Множина задається перерахунком
                    #елементів
M = {i for i in range(4)} #Генератор множини
```

Інформаці про роботу з такою структурою даних можна знайти за цим посиланням.

https://docs.python.org/release/3.12.4/library/stdtypes.html#set-types-set-frozenset

Завдання до лабораторної роботи[6]

*Розв'яжіть задачі відповідно до Вашого варіанту (номер варіанту відповідає порядку запису задачі)***.** *Розв'язок завантажте на відкритий репозиторій GitHub з назвою за зразком Прізвище_var_NN_Lab_2.*

Рівень 1

1. Формується список з *n* дійсних випадкових чисел. Вивести ті числа, які після впорядкування за зростанням збільшили свій номер.

2. Список формується з додатних та від'ємних цілих чисел загальною кількістю не менше 5. Обчислити суму кубів 5 його найбільших елементів.

3. У списку, який складається з *n* дійсних чисел, вивести ті елементи, які менші за середнє арифметичне значення елементів у списку.

4. Генерується список з 15 дійсних чисел. Впорядкувати його за спаданням та вивести всі елементи із непарними індексами, .

5. У заданому масиві з 20 випадково згенерованих цілих чисел знайти всі числа кратні 11 та вивести їх у порядку спадання.

[6] Задачі для різних рівнів складності взято з посібника Мосіюк, О. О. *Штучний інтелект: вступ до машинного навчання.* Вид-во ЖДУ ім. І. Франка, м. Житомир, 2019. 76 с.

Рівень 2

1. Цілі числа вводяться через пропуск у один рядок. Трансформуйте цей рядок чисел у двовимірний список. Забезпечте коректність введення даних. Перевірте, чи даний двовимірний список цілих чисел має вертикальну вісь симетрії.

2. Цілі числа вводяться через пропуск у один рядок. Трансформуйте цей рядок чисел у двовимірний список. Забезпечте коректність введення даних. Забезпечте коректність введення даних у двовимірний список. Перевірте, чи заданий двовимірний список цілих має горизонтальну вісь симетрії.

3. Цілі числа вводяться через пропуск у один рядок. Трансформуйте цей рядок чисел у двовимірний список. Забезпечте коректність введення даних. Забезпечте коректність введення даних у двовимірний список. перевірте, чи заданий двовимірний список [n][n] симетричний відносно головної діагоналі.

4. Цілі числа вводяться через пропуск у один рядок. Трансформуйте цей рядок чисел у двовимірний список. Забезпечте коректність введення даних.Забезпечте коректність введення даних у двовимірний список. перевірте, чи заданий двовимірний список [n][n] симетричний відносно не головної діагоналі.

5. Цілі числа вводяться через пропуск у один рядок. Трансформуйте цей рядок чисел у двовимірний список. Забезпечте коректність введення даних.Забезпечте коректність введення даних у двовимірний список [m][n]. Відобразіть заданий список [m][n] відносно горизонтальної осі.

Рівень 3

1. Задано рядок із слів, які розділені пробілом. Обчисліть кількість слів у рядку та виведіть на екран статистику кількості унікальних слів.

2. Петро та Олена грають у гру. Петро загадує натуральне число від 1 до 100. Олена намагається вгадати це число, називаючи деяку множину натуральних чисел через кому з пропуском. Якщо число присутнє у цій множині, то Петро каже «Yes», якщо його не має – «No». Після декількох запитань Олена заплуталася і не пам’ятає, які питання задавала та просить у Петра допомоги «HELP», у відповідь Петро називає ті числа, які казала Олена і серед яких є правильна відповідь. Кількість спроб Олени обмежена (задається з консолі).

3. У лінгвістичному класі із 10 учнів, кожен з них знає кілька мов. Виведіть на екран всі мови, які знає кожен учень та вкажіть учня, який знає найбільше мов. Дозволяється використовувати словники та множини.

4. Користувач вводить цілі числа через кому у рядок із діапазону [1, N]. Вивести на екран тільки ті числа з введеного рядка, які належать ряду Фібоначчі.

5. Відомості про автомобіль складаються із назви марки, унікального номера авто та прізвища власника. Словник формується наступним чином: ключ - номер авто та значення - список із двох елементів [марка, прізвище власника]. Реалізуйте алгоритм пошуку прізвища власника за допомогою номера та виведіть статистику унікальних марок.

Питання для самоперевірки

1. Рядки та функції роботи із рядками у мові програмування Python.
2. Поняття списку. Одновимірні та багатовимірні списки. Методи роботи із списками.
3. Особливості робот з множинами та кортежами.

4. Чим відрізняється робота з словники у Python, по відношенню до інших складних типів даних?

Довідкові матеріали

1. Офіційна сторінка Python. URL: https://www.python.org/
2. Програмування для всіх: основи Python. URL: https://prometheus.org.ua/course/course-v1:Michigan+PFE101+2023_T3
3. Python: Структури даних. URL: https://prometheus.org.ua/course/course-v1:Michigan+PDS101+2023_T3
4. Розробка та аналіз алгоритмів. Частина 1. URL: https://prometheus.org.ua/course/course-v1:KPI+Algorithms101+2015_Spring

Посилання на літературу

1. Алгоритми та структури даних. Навчальний посібник. Київ, НТУУ «КПІ ім. Ігоря Сікорського», 2022. 215 с.
2. Васильєв О. Програмування мовою Python. Навчальна книга – Богдан, 2019. 504 с.
3. Маттес Е. Пришвидшений курс Python. Львів, Видавництво Старого Лева, 2021. 600 с.

ЛАБОРАТОРНА РОБОТА 3. Оголошення функцій. Модулі у мові програмування Python. Поняття пакету.

Теоретичні відомості

Оголошення функцій

Для опису користувацької функції використовується наступна конструкція.

```
def Назва функції (змінна1, змінна2, …):
    дія 1
    дія 2
    return значення
```

Команда **`return значення`** вказує на те, що функція повертає число, рядок або якийсь інший об’єкт. Якщо ж така директива не використовується, то автоматично повертається – **`None`** .

Змінні, оголошені у «тілі» функції, будуть локальними і доступу до них поза ним буде відсутнім.

Ряд наступних прикладів демонструють як можна передати у саму функції декілька значень, встановити значення за замовчуванням,

Приклад 3.1.

```
# Передаємо довільної кількості значень у функцію
def func(*var):
    return var # Кортеж
func(0,1,2)
```

Приклад 3.2.

```
#Передаємо довільної кількості ключових значень
# у функцію
def func(**var):
    return var # Словник
```

```
func(key1 = 0, key2 = 1, key3 = 2)
```

Приклад 3.3.

```
# Встановлюємо значення за замовчуванням
def func(var1 =  10, var2 = 15):
    return var1, var2
print(func())
```

Функції є, по суті, об’єктом і нічим не відрізняються від змінних, а отже можуть передаватися як параметр у інші функції.

Приклад 3.4.

```
import math as m

def func(a, function):
    return function(a)

print(25, m.sqrt)
```

У Python 3 є можливість створення анонімних функцій, які задаються за допомогою інструкції **`lambda`**. Ключовою особливістю конструкції коду є те, що вона виконується швидше за класичну функцію і не потребує використання зарезервованого слова **`return`**, щоб повернути дані.

Приклад 3.5.

```
f = lambda a, b: a + b
print(f(3, 2))
```

Результат виконання програми:

```
5
```

Модулі

Модуль являє собою функціонально завершений фрагмент програми, що оформлено у вигляді окремого файлу з вихідним кодом,

призначений для використання в інших програмах. Мова програмування Python має чи не найкращий набір бібліотек та пакетів, які можуть використовуватися для вирішення різних типів задач. Наступні приклади демонструють основні команди роботи з модулями.

Приклад 3.6.

```
# Імпортуємо до програми бібліотеку math
import math
math.sqrt(25)
```

Приклад 3.7.

```
# Імпортуємо до програми бібліотеку math позначаючи
# її псевдонімом m для зручності
import math as m
m.sqrt(25)
```

Приклад 3.8.

```
# Імпортуємо окрему команду sqrt з бібліотеки math
from math import sqrt
sqrt(25)
```

Для того щоб створити власний модуль для мови програмування Python треба написати його код, зберегти з розширенням **`py`** у каталог з вже створеними модулями або у тій ж папці де розробляється проект та імпортувати його за допомогою інструкції **`import`**.

При розробці власних модулів варто пам’ятати, що їх не можна використовувати для їх іменування назви ключових слів. Також

враховувати що при його імпорті його код повністю вбудовується у тіло програми і виконується.

Модулі у Python дозволяється об'єднувати у пакети. Пакетом є каталог, який містить файл `__init__.py`. Він може бути порожнім, але зазвичай у цей файл поміщають код ініціалізації такого пакета. Для імпорту пакетів використовується аналогічний синтаксис, що і для модулів

Детальна інформація про роботу з функціями, модулями та пакетами подана за посиланням.

https://docs.python.org/release/3.12.4/tutorial/controlflow.html#function-annotations

https://docs.python.org/release/3.12.4/tutorial/modules.html

https://docs.python.org/release/3.12.4/tutorial/modules.html#packages

Завдання до лабораторної роботи

*Розв'яжіть задачі відповідно до Вашого варіанту (номер варіанту відповідає порядку запису задачі)**.** Розв'язок завантажте на відкритий репозиторій GitHub з назвою за зразком Прізвище_var_NN_Lab_3.*

Рівень 1

1. Рекурсивно обчислити факторіал заданого числа.
2. Записати рекурсію для знаходження *n* числа Фібоначчі.
3. Рекурсивно перевірити чи є задане число натуральним.
4. Вводиться послідовність чисел, яка завершується 0. Виведіть цю послідовність у зворотному порядку, застосовуючи для цього рекурсію.
5. Реалізувати піднесення до цілого степеня, незалежно від того від'ємний чи додатній степінь, використовуючи рекурсію.

Рівень 2

1. Реалізуйте функцію, яка виконує заміну малих літер на великі і навпаки, використовуючи лише класичні інструкції мови Python, властивість **`length`** та доступ до символу через індекс.

2. Створіть функцію, яка визначає кількість слів у заданому рядку, використовуючи лише класичні інструкції мови Python, властивість **`length`** та доступ до символу через індекс.

3. Напишіть функцію, яка повертає індекс першого входження підрядка в рядок, або –1, якщо підрядок не знайдено, використовуючи лише класичні інструкції мови Python, властивість **`length`** та доступ до символу через індекс.

4. Напишіть функцію, яка сортує список числових кортежів за другим елементом кожного кортежу, використовуючи лише класичні інструкції мови Python, властивість **`length`** та доступ до елементів через індекс.

5. Створіть функцію, яка підраховує частоту кожного символу в рядку, використовуючи лише класичні інструкції мови Python, властивість **`length`** та доступ до символу через індекс.

Рівень 3

1. Напишіть рекурсивний розв’язок задачі про Ханойські вежі для числа $n < 6$. У консоль виведіть текстову послідовність кроків.

2. Напишіть функцію, яка генерує всі можливі підмножини заданої множини символів без врахування перестановок та вивести їх на екран.

3. Напишіть функцію, яка перетворює ціле число від 0 до 10000 включно на запис українською мовою.

4. Напишіть функцію, яка розбиває задане ціле число (до 20) на суму простих чисел різними способами.

5. Напишіть функцію, яка виконує множення двох матриць $n * n$.

Питання для самоперевірки

1. Опис користувацької функцій у Python. Рекурсія.
2. Поняття зовнішніх модулів та їх методів.
3. Яким чином у мові програмування Python можна створити пакет.

Довідкові матеріали

1. Офіційна сторінка Python. URL: https://www.python.org/
2. Програмування для всіх: основи Python. URL: https://prometheus.org.ua/course/course-v1:Michigan+PFE101+2023_T3
3. Python: Структури даних. URL: https://prometheus.org.ua/course/course-v1:Michigan+PDS101+2023_T3
4. Розробка та аналіз алгоритмів. Частина 1. URL: https://prometheus.org.ua/course/course-v1:KPI+Algorithms101+2015_Spring

Посилання на літературу

1. Алгоритми та структури даних. Навчальний посібник. Київ, НТУУ «КПІ ім. Ігоря Сікорського», 2022. 215 с.
2. Васильєв О. Програмування мовою Python. Навчальна книга – Богдан, 2019. 504 с.
3. Маттес Е. Пришвидшений курс Python. Львів, Видавництво Старого Лева, 2021. 600 с.

ЛАБОРАТОРНА РОБОТА 4. Об’єктно-орієнтоване програмування.

Теоретичні відомості

Python за визначенням підтримує об’єктно-орієнтоване програмування. Для цієї мови програмування все вважається, певною мірою, об’єктом. Будь-які класи, які визначаються, є об’єктами, і, звичайно, екземпляри цих класів також є об’єктами.

У парадигмі мови програмування Python клас – це шаблон коду, який дозволяє створювати об’єкти певного зразка. Для оголошення класу використовують ключове слово **`class`**. Відповідно до правил іменування в мові Python назви класів прийнято оголошувати з великої літери. Користувацькі класи, є екземплярами більш загального – **`object`**, а отже мають свої атрибути і методи. Всі атрибути і методи класу при створенні екземпляра цього класу стають атрибутами і методами цього об’єкту. Зміни значень атрибутів об’єкта не впливають на зміну значень атрибутів класа.

Приклад 4.1.

```
# Оголошення класу
class Student:
    pass


class Pupil:
    age = 12
    school = 7
# Створення екземпляра Jack  класа Pupil
Jack = Pupil()
```

Конструктори класів є фундаментальною частиною об’єктно-орієнтованого програмування на Python. Вони дозволяють створювати та правильно ініціалізувати об’єкти даного класу, роблячи ці об’єкти готовими до використання.

Конструктори класів внутрішньо запускають процес створення екземпляра Python, який проходить через два основні кроки: створення екземпляра класу та ініціалізація цього екземпляра.

Функція **`__init__`** є конструктором, яку можна у процесі написання класу переозначити та передавати до неї дані.

`self` – дозволяє викликати поточний екземпляр класу всередині цієї структури.

Приклад 4.2.

```
# Оголошення класу та переозначення __init__
class Student:
    def __init__(self) -> None:
        self.age = 18
        self.course = 1
        self.name = "Oleh"

person_one = Student()
```

Приклад 4.3.

```
# Оголошення класу та переозначення __init__,
# коли при створенні екземпляра класу передаються
# певні значення
class Student:
    def __init__(self, age:int, name:str) ->
None:
```

```
        self.age = age
        self.name = name

    def learn(self) -> None:
        print(["Ukrainian", "Algebra",
               "Geometry", "English"])

person = Student(12, "Oleh")
```

Для того щоб оголосити методи класу необхідно у конструкції класуоголосити функцію, яка обов'язково приймає першим параметром значення **self**.

Public, Private, Protected властивості та методи класу

Всі атрибути за замовчуванням вважаються публічними. За домовленістю захищеними вважаються атрибути, які починаються з одного підкреслення, але до них доступ зберігається. Приватними вважаються всі атрибути, у яких назва розпочинається на подвійне підкреслення.

Класичні методи Python не забезпечують повного обмеження доступу до даних. Для того щоб повністю обмежити доступ до даних і методів класу варто використовувати вбудованих модуль **accessify**.

Приклад 4.4.

```
# Оголошення класу та публічних, захищених та
# приватних змінних
class Student:
    def __init__(self, age:int,
                       name:str,
                       mark:int) -> None:
```

```
            self.age = age #Публічна змінна
            self.name = name #Публічна змінна
            #Захищена змінна
            self._faculty = "Faculty of Physics and
                            Mathematics"
            self.__mark = mark #Приватна змінна
        def show_mark(self):
            return self.__mark

    person = Student(18, "Oleh", 90)

    print("Public attr name: ", person.name)
    print("Public attr age: ", person.age)
    print("Protected attr _faculty:",
person._faculty)
    print("Return value of the private attr: ",
          person.show_mark())
    # Тут буде помилка
    print("Private attr __mark", person.__mark)
```

Успадкування класів у Python

Для того щоб виконати успадкування класу від іншого необхідно при оголошенні нового класу вказати у дужках назву батьківського класу, а при створенні конструктора для наслідуваного класу, обов'язковим є виклик конструктора батьківського класу в тілі конструктора дочірнього класу.

Приклад 4.5.

```
# Приклад наслідування класів у Python
class Mammals:
```

```
        def __init__(self) -> None:
            self.paws = 4
        def eat(self):
            print("I like eat")

    class Dog(Mammals):
        def __init__(self) -> None:
            Mammals.__init__(self)
            self.name = "Jack"
            self.breed = "Labrador"

        def can_woof(self) -> None:
            print("I like bark out")

        def eat(self):
            print("I like eat bones and meat.")
```

Більше прикладів роботи з класами в мові програмування Python можна знайти за посиланням.

https://github.com/XandrMos/SampleForTeachers_Feb_2024

Завдання до лабораторної роботи

Розв'яжіть задачі відповідно до Вашого варіанту (номер варіанту відповідає порядку запису задачі). Розв'язок завантажте на відкритий репозиторій GitHub з назвою за зразком Прізвище_var_NN_Lab_3.

Варіант 1

Реалізуйте текстову гру битви героя (воїн з мечем - має гарний захист або луком - має ефективне ураження) із драконом (дракон дихає

вогнем або б’є лапою). Герой має ім'я, яке вводиться з клавіатури. Кількість здоров'я задається програмно. Рівень атаки (числове значення) задається програмно. Рівень захисту (числове значення) задається програмно. Аналогічні параметри має дракон, ім'я дракону задається автоматично. Тип героя або дракона наслідується від одного загального класу.

Процес бою.

Бій відбувається по раундах, які оголошуються.

На початку раунду герой вирішує атакувати чи захищатися.

Для дракона ця дія реалізується генерацією випадковго числа.

Сила атаки і рівень захисту генерується автоматично та випадковим чином (максимальне значення атаки і захисту відповідає заданому значенню).

Бій відбувається до тих пір, поки рівень здоров'я учасників не буде рівним 0 або менше 0, після чого виводиться повідомлення про переможця.

Ситуації бою.

Співпадіння атак.

У кожного учасника бою віднімається кількість здоров'я, яке рівне половині атаки противника.

Атака одного учасника більша за атаку іншого.

Виграє раунд той супротивник, у кого вона більша і, відповідно, у учасника, який програв, віднімається та кількість здоров'я, яка рівна атаці переможця.

Атака та захист.

Атака зараховується тільки тоді, коли вона більша за захист опонента (віднімається кількість здоров'я, яка рівна різниці атаки та захисту). У іншому випадку атака вважається відбитою.

Герой та дракон виставляють захист одночасно

Нічия.

Врахувати всі можливі ситуації, які можуть виникнути у процесі гри. Реалізація має бути на основі ООП.

Варіант 2

Напишіть покрокову гру, яка дозволяє посадити на планету космічний корабель. Гра представляє собою серію кроків, для яких треба вводити координати наступної точки, до якої має переміститися корабель.

Позиціонування корабля, планети та перешкод задається псевдографічним символьним інтерфейсом і виводиться на екран з кожним кроком.

Гравець виграє, якщо космічний корабель заходить в атмосферу планети (окіл на 1 більший за розмір планету).

Космічний корабель розбивається об перешкоду, якщо попадає у одиничний її окіл або на саму перешкоду або ж його лінійний шлях перетинає цю перешкоду чи її окіл.

Космічний корабель займає одне ігрове поле 1*1, перешкоди (до чотирьох на гру) мають варіативні розміри квадратів від 1*1 до 3*3, розміри планети варіюються від 4*4 до 5*5.

Між перешкодами має бути відстань мінімум 3 клітинки.

Розміщення всіх ігрових елементів задається випадковим чином на початку гри.

Реалізація має бути на основі ООП.

Варіант 3

Написати програму, що моделює гру «Життя» Конвея з використанням класичних правил і декількома доповненнями:

1. Гра йде по кроках

2. Гра ведеться на тороїдальному полі 10 на 10 (координати від A-J, 0-9) сусідами нижніх (горизонталь 9) клітин є верхні (горизонталь 0), сусідами правих клітин (вертикаль J) є ліві (вертикаль A).

3. Кожна клітина має 8 сусідів (по горизонталі та діагоналі).

4. Якщо у порожній (мертвої) клітини рівно три сусіда-організму (живі клітини), то на

наступний хід в ній зароджується життя (організм).

5. Якщо у живої клітини менш двох живих сусідів то вона вмирає від самотності.

6. Якщо у живої клітини більше трьох живих сусідів то вона вмирає від перенаселення

7. Гра закінчується якщо:

-- на поле не залишається живих клітин;

-- жива конфігурація не змінює свій стан.

Реалізувати класи, які визначають модель гри (поле 10 на 10).

Реалізувати інтерфейс у вигляді поля 10 на 10 з підписаними координатами в символьному режимі (для порожньої клітки можна використовувати символ підкреслення '_', для живої - символ зірочки '*', а також з відображенням лічильника ходів.

Команди вводяться в текстовому режимі:

`reset` – очистити поле і лічильник ходів

`set N` – згенерувати N довільних організмів на полі (2 <= N < 100)

`clear` – очистити поле

`step N` – прокрутити гру вперед на N кроків (показник може бути відсутнім, тоді потрібно його вважати рівним 1)

`back N` – прокрутити гру назад на N кроків (показник може бути відсутнім, тоді потрібно його вважати рівним 1), але не більшим ніж загальна кількість кроків, що була зроблена.

Варіант 4

Напишіть програму, яка дозволяє згенерувати поле лабіринту з одним входом та одним виходом розмірністю 25 на 25 позицій та реалізуйте ігровий процес, який дозволяє покроково провести персонажа від початку до кінця лабіринту.

Позиція персонажа (клітинка 1*1) задається указуванням пари координат.

При кожному кроці відбувається виведення поточного розміщення на псевдо карті, яка також по новому перерисовується.

Персонаж може переміщатися лише по горизонталі і вертикалі, та не може проходити через стіни лабіринту.

Гравець виграє, якщо персонаж досягає виходу.

Варіант 5

Реалізуйте комп’ютерну гру «Морський бій» з комп’ютером на полі 10*10 клітинок з випадковим розміщенням 1 лінійного корабля на 4 клітинки, 2 лінійними кораблями на дві клітинки та 2 кораблями на одну клітинку.

Розміщення кораблів як для гравця так і для комп’ютера задаються випадковим чином, але при умові, що жоден корабель не може бути розміщений ближче ніж на 1 клітинку по діагоналям, горизонталі і вертикалі до вже існуючих кораблів.

Ігровий простір являє собою два набори полів 10*10, які відображають розміщення кораблів гравця і поле, з прихованими кораблями для комп’ютера.

Постріл задається парою координат і відображається на кожному полі. Комп’ютер не може стріляти по клітинкам, по яким уже був здійснений постріл.

Ігровий простір виводиться з кожним ходом.

Виграє той, хто знищить усі кораблі противника, або на момент завершення гри гравцем буде знищено або пошкоджено найбільше кораблів.

Питання для самоперевірки

1. Як задається клас у мові програмування Python?
2. Які основні принципи ООП Ви знаєте і як вони реалізовані в Python?
3. Які є особливості управління доступом до атрибутів та методів класу в Python?
4. Що таке методи класу та методи екземпляра? Яка їх різниця?
5. Які способи успадкування підтримує Python?

Довідкові матеріали

1. Офіційна сторінка Python. URL: https://www.python.org/
2. Програмування для всіх: основи Python. URL: https://prometheus.org.ua/course/course-v1:Michigan+PFE101+2023_T3
3. Python: Структури даних. URL: https://prometheus.org.ua/course/course-v1:Michigan+PDS101+2023_T3

4. Розробка та аналіз алгоритмів. Частина 1. URL: https://prometheus.org.ua/course/course-v1:KPI+Algorithms101+2015_Spring

Посилання на літературу

1. Алгоритми та структури даних. Навчальний посібник. Київ, НТУУ «КПІ ім. Ігоря Сікорського», 2022. 215 с.

2. Васильєв О. Програмування мовою Python. Навчальна книга – Богдан, 2019. 504 с.

3. Маттес Е. Пришвидшений курс Python. Львів, Видавництво Старого Лева, 2021. 600 с.

ЛАБОРАТОРНА РОБОТА 5. Бібліотеки для аналізу статистичних даних та приклади їх застосування.

Теоретичні відомості

Бібліотека NumPy

NumPy – це бібліотека Python для створення та обробки масивів даних. Вона дає змогу виконувати математичні обчислення і обробляти великі масиви числових даних.

Центральним поняттям є *n*-вимірні масиви, а також уніфіковані команди, які надають можливість опрацьовувати дані незалежно від того якої розмірності має бути масив.

Для того щоб використовувати бібліотеку NumPy у програмі її необхідно підключити. Нижче наведено код, який демонструє приклад підключення зазначеної бібліотеки.

Приклад 5.1[7].

```
# Приклад наслідування класів у Python
import numpy #підключає всю бібліотеку
from numpy import array #з всієї бібліотеки
                        #імпортується тільки
                        #команда array
import numpy as np  #підключаю бібліотеку та
#призначає скорочення np
```

Основний об'єкт NumPy є гомогенний багатовимірний масив. Фактично, це таблиця елементів (переважно чисел), одного й того ж типу, індексація елементів таблиці починається з 0. Розмірність у NumPy позначають «осями».

[7] Код, який поданий у подальших прикладах розрахований на виконання в онлайн срердивищі розробки Google Colaboratory

ndarray - основний клас бібліотеки, для якого створений псевдонім **array**. При цьому цю команду не потрібно плутати з стандартною бібліотекою Python **array.array**. Це не одне і те саме.

Наступний ряд прикладів показує важливі операції, які можна виконувати за допомогою команд бібліотеки NumPy.

Приклад 5.2.

```
#Створенння масиву в NumPy
import numpy as np
arr = np.array([1, 2, 3]) #Створює масив NumPy з
                          #числового списку Python
print(arr)
```

Приклад 5.3.

```
#Створення упорядкованих одновимірних масивів
import numpy as np
arr = np.arange(10) #Задає масив чисел від 0 до
                    #9 з кроком 1
print("NumPy array: ", arr)
```

Приклад 5.4.

```
#Створення упорядкованих одновимірних масивів
import numpy as np
arr = np.arange(-6, 5, 0.5) #Задає масив дійсних
                            #чисел від -5 до 5 з
                            #кроком 0.5
print("NumPy array: ", arr)
```

Для створення двовимірних масивів NumPy використовують такі команди: **numpy.eye**, **numpy.diag** та **numpy.vander**.

Приклад 5.5.

```
#Створення двовимірного масиву з одниницями по
#головній діагоналі
import numpy as np
arr = np.eye(4)
print("Квадратний двовимірний масив з одиницями
по діагоналі\n", arr)
arr = np.eye(3, 5)
print("Прямокутний двовимірний масив з одиницями
по діагоналі\n", arr)
```

Приклад 5.6.

```
#Створення двовимірного масиву з заданими числами
#по головній діагоналі

import numpy as np
arr = np.diag([1, 2, 3, 4])
print("Квадратний двовимірний масив з заданими
      числами, які задаються списком, по
      діагоналі\n", arr)
arr = np.diag([1, 2, 3], 1)
print("Квадратний двовимірний масив з заданими
      числами, які задаються списком, та зміщеною
      діагоналлю на 1 праворуч\n", arr)
arr = np.array([[1, 5], [7, 8]])
print("Значення всіх елементів таблиці по
головній діагоналі\n", np.diag(arr))
```

Також існує можливість одразу заповнювати матриці нулями та одиницями.

Приклад 5.7.

```
#Заповнюємо масив нулями
import numpy as np
arr = np.zeros((4, 5)) #Розмірність задається за допомогою кортежу
print("Масив заповнений всіма нулями\n", arr)
```

Приклад 5.8.

```
#Заповнюємо масив одиницями
import numpy as np
arr = np.ones((3, 5)) #Розмірність задається за
                      #допомогою кортежу
print("Масив заповнений всіма нулями\n", arr)
```

Для отримання важливої інформації про масив NumPy використовують ряд методів: **ndim**, **shape**, **size**, **dtype**, **itemsize**, **data**.

Приклад 5.9.

```
import numpy as np
arr = np.array([[1, 2, 3],[4, 5, 6],[7, 8, 9]],
               dtype = float)
print("Розмірність масиву: ", arr.ndim)
print("Форма масиву: ", arr.shape)
print("Тип об'єктів масиву: ", arr.dtype)
print("Кількість байт, що виділяється для 1
      елемента масиву (в Байтах): ",
arr.itemsize)
```

```
print("Адреса пам'яті, яка фактично містить поточні елементи масису: ", arr.data)
```

Масиви NumPy можна додавати, віднімати, множити, ділити, визначати остачу від ділення, виконувати цілочисельне ділення, підносити до степені, порівнювати як поелементно так і на число, але треба щоб масиви мали однакову розмірність

Приклад 5.10.

```
#Додавання
import numpy as np
arr = np.array([[1, 2, 3],[4, 5, 6],[7, 8, 9]],
                dtype = int)
print("Масив до виконання дії додавання\n", arr)
rezult = arr + 2
print("Масив після виконання дії додавання\n",
      rezult)
```

Приклад 5.11.

```
#Дії на самими масивами
import numpy as np
arr_a = np.array([[1, 2, 3],[4, 5, 6],[7, 8, 9]], dtype = int)
print("Перший масив\n", arr_a)
arr_b = np.ones((3, 3), dtype = int)
print("Другий масив\n", arr_b)
rezult = arr_a + arr_b
print("Результат дії\n", rezult)
```

Бібліотека NumPy має ряд вбудованих функцій, які можуть спростити обчислення та дії з елементами масивів. До них відносять такі

функції: **all, any, apply_along_axis, argmax, argmin, argsort, average, bincount, ceil, clip, conj, corrcoef, cos, cov, cross, cumprod, cumsum, diff, dot, exp, floor, inner, invert, lexsort, max, maximum, mean, median, min, minimum, nonzero, outer, prod, re, round, sin, sort, std, sum, trace, transpose, var, vdot, vectorize, where.**

Більш детально про них можна прочитати у документації https://numpy.org/devdocs/user/index.html.

Доступ до кожного елемента масиву відбувається за індексом аналогічно до списків у Python. Схожим чином виконуються зрізи та перебір елементтів за допомогою циклу.

Форма масиву може бути змінена за допомогою ряду команд.

Приклад 5.12.

```
#Функція ravel зводить будь-який масив до
#одновимірного
import numpy as np
arr = np.array([[1, 2, 3],
                [4, 5, 6],
                [7, 8, 9]], dtype = int)
print("Двовимірний масив\n", arr)
arr = arr.ravel()
print("Двовимірний масив перетворився у
       одновимірний\n", arr)
```

Приклад 5.13.

```
#Перетворити масив з n*m на масив k*l можна за
#допомогою функції reshape
```

```
import numpy as np
arr = np.array([[1, 2, 3],
                [4, 5, 6],
                [7, 8, 9],
                [10, 11, 12]], dtype = int)
print("Двовимірний масив 4 рядками і 3
       стовпчиками\n", arr)
arr = arr.reshape((2, 6))
print("Двовимірний масив 2 рядками і 6
       стовпчиками\n", arr)
```

Приклад 5.14.

```
#Змінити форму масиву також можливо і за
допомогою
#функції
#бібліотеки NumPy -- resize
import numpy as np
arr = np.array([[1, 2, 3],
                [4, 5, 6],
                 [7, 8, 9],
                 [10, 11, 12]], dtype = int)
print("Початковий масив\n", arr)
arr = np.resize(arr, (3, 5))
print("Модифікований масив\n", arr)
```

Звичайно це ну повний перелік дій, які можна виконувати на масивами NumPy, але він дає розуміння про можливості бібліотеки. Більше про неї можна прочитати за посиланням: https://numpy.org/.

Бібліотека Pandas

Pandas – це безкоштовна та відкрита бібліотека мови програмування Python для аналізу і маніпулювання даними. Вона заснована на структурі даних «DataFrame», яка є аналогом електронних таблиць.

До основних структур даних бібліотеки Pandas відносять **`Series`** та **`DataFrame`**.

`Series` – це одновимірний масив, який замість індексів можуть бути використані асоційовані мітки. Звернення до значення у серії відбувається аналогічно до того як це відбувається у звичному словнику Python.

Приклад 5.15.

```
import numpy as np
#Оскільки Pandas є високорівневою
#надбудовою над NumPy, то ці
#бібліотеки часто доповнюють один одного
import pandas as pd
#Власне підключення бібліотеки Pandas
new_series = pd.Series([1, 2, 3, 4, 5])
print("Створена нова серія\n", new_series)
```

Результат роботи коду

```
Створена нова серія
0    1
1    2
2    3
3    4
4    5
dtype: int64
```

У представленому прикладі формується об'єкт Series і при друці у консолі у лівому стовпчику виводяться індекси, а праворуч подаються відповідні значення. Якщо формат міток не задано, то Python формує числові індекси від 0 до N - 1, де N - кількість елементів у серії.

Звернутися до окремого елемента можна таким чином як до значення у словнику Python (приклад 5.16).

Приклад 5.16.

```
import numpy as np
import pandas as pd
new_series = pd.Series([1, 2, 3, 4, 5])
print("new_series[3] = ", new_series[3])
```

Наступні приклади показують основні особливості роботи з серіями (робити вибірку по окремим індексам, здійснювати множинне присвоєння, застосовувати математичні операції, фільтрувати дані тощо).

Приклад 5.17.

```
#Індекси можна задати явно.
import numpy as np
import pandas as pd
new_series = pd.Series([1, 2, 3, 4, 5],
                       index=['a', 'b', 'c', 'd', 'e'])
print("Створена нова серія\n", new_series)
print("new_series['d'] = ", new_series['d'])
```

Приклад 5.18.

```
# Вибірка значень
import numpy as np
import pandas as pd
```

```
    new_series = pd.Series([1, 2, 3, 4, 5],
                 index=['a', 'b', 'c', 'd', 'e'])
    print("Вибірка значень\n",
          new_series[['b','d', 'e']])
```

Приклад 5.19.

```
    # Множинне присвоєння
    import numpy as np
    import pandas as pd
    new_series = pd.Series([1, 2, 3, 4, 5],
                 index=['a', 'b', 'c', 'd', 'e'])
    print("Вибірка занчень\n", new_series)
    new_series[['b','d', 'e']] = 0
    print("Після множинного присвоєння\n",
new_series)
```

Приклад 5.20.

```
    #Асоційовані мітки можна змінювати у процесі
    #виконання коду.
    import numpy as np
    import pandas as pd
    new_series = pd.Series([1, 2, 3, 4, 5],
                 index=['a', 'b', 'c', 'd', 'e'])
    print("Серія з початковими індексами\n",
new_series)
    new_series.index = ['A', 'B', 'C', 'New_id', 'E']
    print("Серія із зміненими індексами\n",
new_series)
```

DataFrame представляє собою багаторозмірні таблиці, стовпчиками якої є серії.

Приклад 5.21.

```
import pandas as pd

#Створюємо словник
data = {
  "Серія 1": [420, 380, 390],
  "Серія 2": [50, 40, 45]
}

#Трансформуємо словник у DataFrame
new_df = pd.DataFrame(data)
print("Створення DataFrame\n", new_df)
```

Для того щоб перегянути рядок даних варто використати метод **loc**.

Приклад 5.22.

```
import pandas as pd

#Створюємо словник
data = {
  "Серія 1": [420, 380, 390],
  "Серія 2": [50, 40, 45]
}

#Трансформуємо словник у DataFrame
new_df = pd.DataFrame(data)
```

```
    #Виводимо дані рядка
    print("Виводимо 1 рядок DataFrame\n",
new_df.loc[0])

    #Список рядків з DataFrame
    #Тут варто вказати список рядків
    print("Виводимо список рядків DataFrame\n",
          new_df.loc[[0, 1]])
```

У DataFframe є можливість помістити таблиці CSV та JSON.

Приклад 5.23.

```
    import pandas as pd
    url =
'./sample_data/california_housing_test.csv'
    df = pd.read_csv(url)

    print(df)
```

Приклад 5.24.

```
    import pandas as pd
    df = pd.read_json('./sample_data/anscombe.json')

    print(df)
```

Для того щоб переглянути перші (останні) *n* рядків таблиці варто використати метод **`head() (tail()`**). Загальне інформація про DataFrame дозволяє вивести метод **`info()`**.

Приклад 5.25.

```
    import pandas as pd
```

```
df = pd.read_json('./sample_data/anscombe.json')
print("Перші 12 рядків\n",df.head(12))
#Загальна інформація про DataFrame
print("\n",df.info())
```

Звичайно цей перелік команд неповний, але того щоб дізнатися більше про особливості роботи з бібліотекою Pandas зверніться до офіційної документації: https://pandas.pydata.org/.

Бібліотека Matplotlib

Matplotlib є низькорівневою бібліотекою для створення графічного представлення статистичних даних. Вона є відкритою та може вільно використовуватися у наукових дослідженнях.

Розглянемо на прикладах особливості роботи з графічною бібліотекою. Для створення першого графіка згенеруємо набір точок за домомогою модуля NumPy.

Приклад 5.26.

```
import matplotlib.pyplot as plt
import numpy as np

xpoints = np.array([0, 8])
ypoints = np.array([0, 240])

plt.plot(xpoints, ypoints) #Формужмо графік
plt.show() #Виводимо графік на екран
```

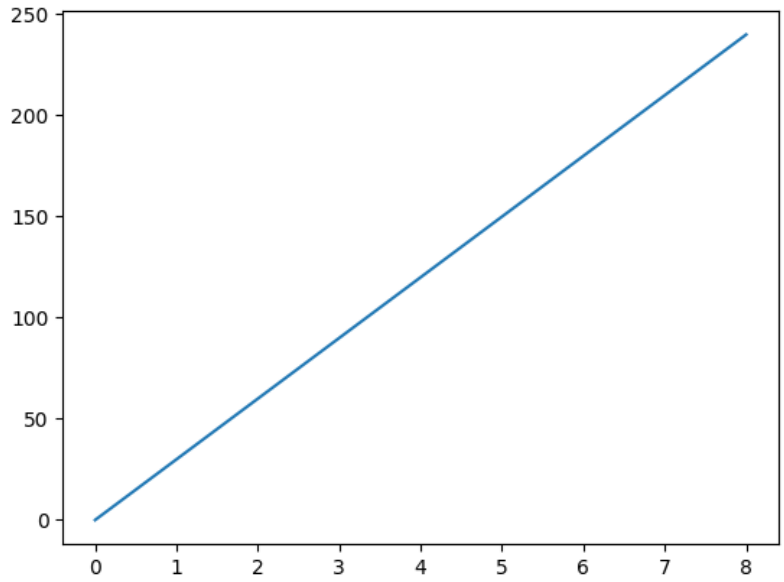

Рис. 5.1. Результат роботи програми, представленої у прикладі 5.26.

У попередньому прикладі функція **`plot()`** будує за замовчуванням лінію від однієї точки до іншої. Фунція приймає два параметри масив, що містить *x*-ві координати точок, та масив з *y*-вими значеннями цих точок. Для попереднього прикладу початок лінії міститиметься у точці (0, 0), а кінець - у точці (8, 240).

Ось як буде виглядати графік, якщо будуть відображатися початкові та кінцеві точки.

Приклад 5.27.

```
import matplotlib.pyplot as plt
import numpy as np

xpoints = np.array([0, 8])
ypoints = np.array([0, 240])

plt.plot(xpoints, ypoints, 'o')
#мітка 'o' вказує, що відображатимуться точки
plt.show() #Виводимо графік на екран
```

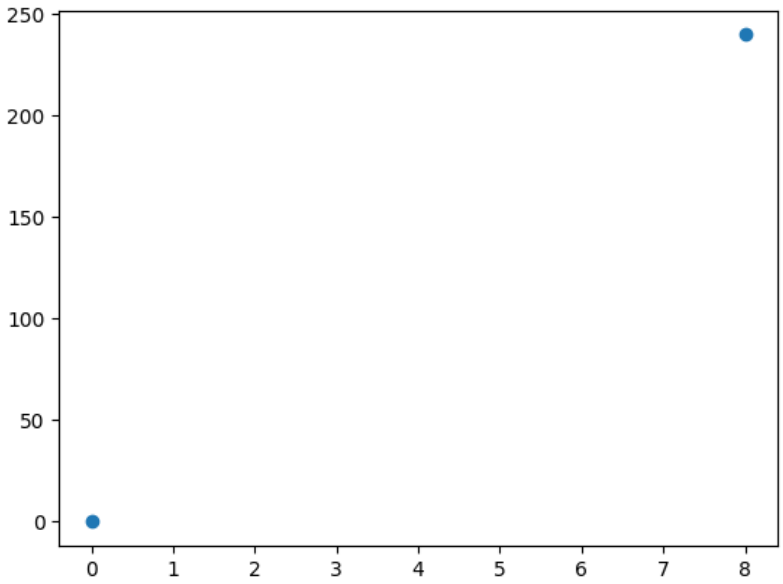

Рис. 5.2. Результат роботи програми, представленої у прикладі 5.27.

Схожим чином можна побудувати ламану. Скопіюйте код з наступного прикладу 5.28., вставте у комірку Google Colaboratory і перевірте його виконання.

Приклад 5.28.

```
import matplotlib.pyplot as plt
import numpy as np
ypoints = np.array([0, 85, 300, 240])
plt.plot(ypoints) #Формужмо графік
plt.show() #Виводимо графік на екран
```

За допомогою додаткових параметрів дозволяється якісно оформити графік функції. Так параметр **`marker`** задає зовнішній вигляд точки.

Форматування ліній графіка також присутнє у бібліотеці matplotlib. Для цього використовується атрибут (параметр) linestyle або його скорочення ls. Типовими значеннями можуть бути:**`'solid', dotted', 'dashed', 'dashdot', 'None'`**.

Приклад 5.29.

```
import matplotlib.pyplot as plt
```

```
import numpy as np

ypoints = np.array([0, 85, 300, 240])

plt.plot(ypoints, "<:r",
         ms = 10, mec = 'y',
         mfc = 'b') #Формужмо графік
plt.show() #Виводимо графік на екран
```

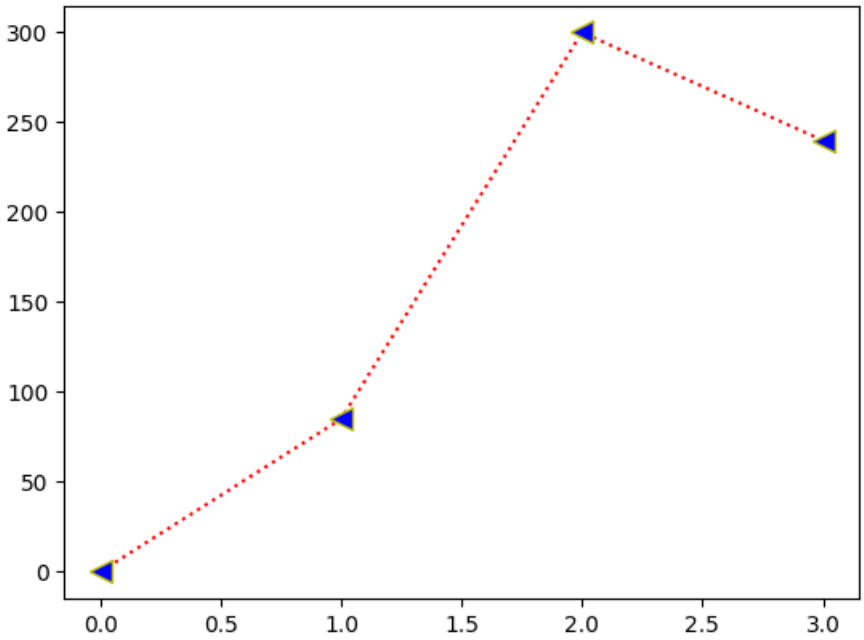

Рис. 5.3. Результат роботи програми, представленої у прикладі 5.29.

Накладанням декількох графіків на одному робочому полотні відбувається наступним чином (приклад 5.30).

Приклад 5.30.

```
import matplotlib.pyplot as plt
import numpy as np

ypoints_1 = np.array([7, -3, 1, 10])
ypoints_2 = np.array([4, 0, 1, 6])
```

```
plt.plot(ypoints_1, linestyle = 'dotted')
plt.plot(ypoints_2, linestyle = 'solid')
plt.show()
```

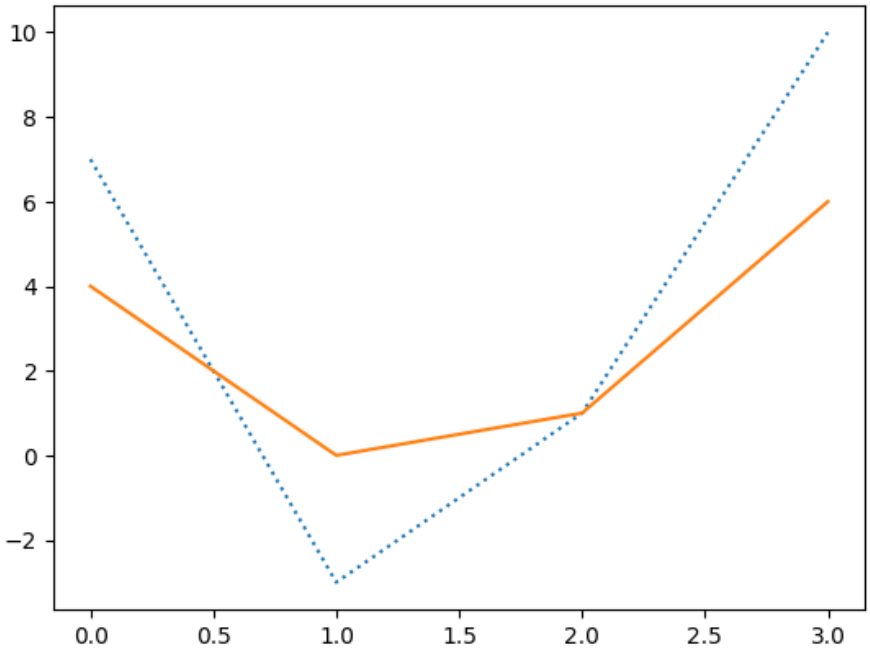

Рис. 5.3. Результат роботи програми, представленої у прикладі 5.30.

У процесі створенням графіків за допомогою бібліотеки **matplotlib** досить важливо правильно оформляти їх. Так для того, щоб підписати осі використовують методи **xlabel()** та **ylabel()**.

Заголовок графіка задається за допомогою методу **title()**. Центрування відбувається за допомогою атрибута **loc**, який може набувати значень: **left, center, right**.

Шрифт та його параметри задаються за допомогою словника і атрибута **fontdict**.

Важливим моментом є створення легенди для графіка. За таку функцію відповідає метод **legend()**. Для того щоб задати розміщення легенди на графіку використовують вже знайомий параметр **loc**, який може набирати таких значень: **left, upper right, upper left,**

lower left, lower right, right, center left, center right, Lower center, upper center, center.

Написи у легенді можна розбити на колонки.

Для відображення координатної сітки використовують метод **grid()**. Якщо необхідно задати горизонтальні або вертикальні лінії, то варто скористатися параметром **axis** з значеннями: **'x'** або **'y'**.

Налаштування зовнішнього вигляду сітки задається аналогічними параметрами як і для графіків **grid(color = 'color', linestyle = 'linestyle', linewidth = number).**

Приклад 5.31.

```
import matplotlib.pyplot as plt
import numpy as np

ypoints_1 = np.array([7, -3, 1, 10])
ypoints_2 = np.array([4, 0, 1, 6])

font1 = {'family':'serif',
         'color':'blue',
          'size':12}
font2 = {'family':'serif',
         'color':'darkred',
         'size':14}

plt.plot(ypoints_1, linestyle = 'dotted')
plt.plot(ypoints_2, linestyle = 'solid')

plt.title("Приклад графіка")
plt.xlabel("Вісь X", fontdict = font1)
```

```
plt.ylabel("Вісь Y", fontdict = font2)
plt.legend(["blue", "orange"],
           loc = "upper center",
           ncol = 2)
plt.grid(color = 'green',
         linestyle = '--',
         linewidth = 0.5)

plt.show()
```

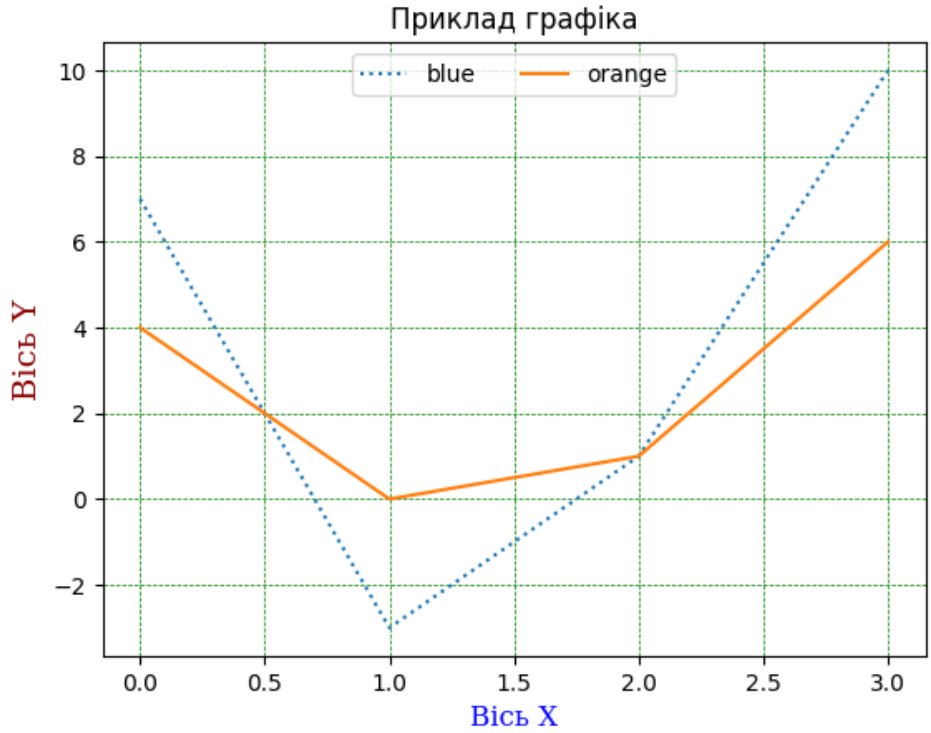

Рис. 5.4. Результат роботи програми, представленої у прикладі 5.31.

У багатьох випадках необхідно на одному полотні розмістити 2 і більше графіків для порівняння результатів дослідження. Для цього використовують метод **`subplot(row, column, number_graph)`**, де **`row`** – задає число рядків для позиціонування графіків; **`column`** - число колонок; **`number_graph`** – номер графіка.

Щоб додати загальну назву для всього полотна варто скористатися методом **`suptitle()`**.

Приклад 5.32.

```
import matplotlib.pyplot as plt
import numpy as np

#plot 1
ypoints_1 = np.array([7, -3, 1, 10])

font1 =
{'family':'serif','color':'blue','size':12}

plt.subplot(1, 2, 1)
plt.plot(ypoints_1, linestyle = 'dotted')
plt.title("Графік 1")
plt.xlabel("Вісь X", fontdict = font1)
plt.ylabel("Вісь Y", fontdict = font1)
plt.legend(["blue"], loc = "lower center")
plt.grid(color = 'orange',
         linestyle = '--', linewidth = 0.5)

#plot 2
ypoints_2 = np.array([4, 0, 1, 6])

font2 = {'family':'serif',
         'color':'darkred','size':10}

plt.subplot(1, 2, 2)
```

```
plt.plot(ypoints_2, linestyle = 'solid')
plt.title("Графік 2")
plt.xlabel("Вісь X", fontdict = font2)
plt.ylabel("Вісь Y", fontdict = font2)
plt.legend(["blue"], loc = "lower center")
plt.grid(color = 'green',
         linestyle = '--',
         linewidth = 0.5)

plt.show()
```

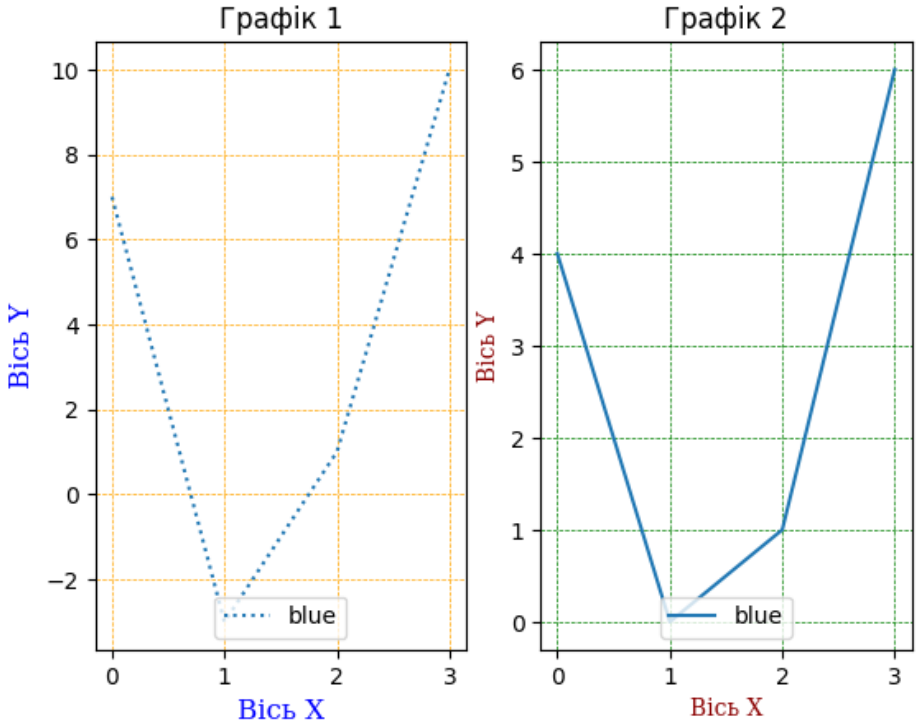

Рис. 5.5. Результат роботи програми, представленої у прикладі 5.32.

Також бібліотека matplotlib дозволяє виконувати побудову точкових (метод **`scatter()`**), кругових (метод **`pie()`**) та стовпчикових (метод `bar()`) діаграм, а також гістограми (метод **`hist()`**).

Коротко підсумовуючи огляд можливостей графічної бібліотеки зауважимо, що у представленому практикумі наведено лише незначну частину всіх можливостей ресурсу. Для більш глибшого дослідження цього інструмента графічного аналізу даних скористайтеся офіційним сайтом: https://matplotlib.org/.

Завдання до лабораторної роботи

1. Розмістіть на Вашому Google Disk файл бази даних з інформацією про загиблих на британському трансатлантичному пароплаві «Титанік» та виконайте завантаження до DataFrame.
https://github.com/datasciencedojo/datasets/blob/master/titanic.csv

Розшифровка полів даних

Pclass — клас пасажира (1 — високий, 2 — средній, 3 — низький);

Name — ім’я;

Sex — стать;

Age — вік;

SibSp — кількість братів, сестер на борту корабля;

Parch — кількість батьків, дітей на борту корабля;

Ticket — номер білета;

Fare — оплата;

Cabin — номер каюти;

Embarked — порт (C — Шербур; Q — Квінстаун; S — Саутгемптон).

2. Відповідно до Вашого варіанту виконайте наступні завдання:

Варіант 1.

1) Виведіть загальну інформацію про зазначену базу даних.
2) Виведіть перші 5 та останні 10 записів.

3) Виведіть тільки тих хто вижив після катастрофи та збережіть інформацію у окремий DataFrame.

4) Визначте кількість людей, які вижили мали білети 1 класу, 2 класу, 3 класу.

5) Створіть секторну діаграму, яка показує кількість дітей, підлітків, молодих людей, людей працездатного віку та пенсійного віку, що вижили після трагедії.

Варіант 2.

1) Виведіть загальну інформацію про зазначену базу даних.

2) Виведіть перші 5 та останні 10 записів.

3) Виведіть тільки тих хто не вижив після катастрофи та збережіть інформацію у окремий DataFrame.

4) До яких класів білетів переважно належали ці люди.

5) Побудуйте стовпчасту діаграму, яка показувала кількість дітей, підлітків, молодих людей, людей працездатного віку та пенсійного віку, що не вижили після трагедії.

Варіант 3.

1) Виведіть загальну інформацію про зазначену базу даних.

2) Виведіть перші 5 та останні 10 записів.

3) Виведіть тільки тих хто мав родичів або ж батьків на пароплаві та збережіть інформацію у окремий DataFrame.

4) У яких портах було взято на борт найбільше пасажирів, які мали 3 та більше родичів пароплаві.

5) Побудуйте стовпчасту діаграму, яка показувала кількість пасажирів, що було взято у кожному в порту.

Варіант 4.

1) Виведіть загальну інформацію про зазначену базу даних.

2) Виведіть перші 5 та останні 10 записів.

3) Виведіть тільки тих хто був старше 30 і вижив у катастрофі та збережіть інформацію у окремий DataFrame.

4) Виведіть інформацію про всіх членів екіпажу, які мали хоча б одного брата чи сестру.

5) Побудуйте стовпчасту діаграму, яка показувала кількість дітей, підлітків, молодих людей, людей працездатного віку та пенсійного віку, що вижили після трагедії.

Варіант 5.

1) Виведіть загальну інформацію про зазначену базу даних.

2) Виведіть перші 5 та останні 10 записів.

3) Виведіть тільки всіх підлітків (до 18), які вижили у катастрофі, та збережіть інформацію у окремий DataFrame.

4) Знайдіть усіх пасажирів, які зійшли на борт пароплава у місті Шербур.

5) Побудуйте стовпчасту діаграму, яка показувала кількість дітей, підлітків, молодих людей, людей працездатного віку та пенсійного віку, що зійшли на борт пароплава у місті Саутгемптон.

3. Кожна дія має виконуватися у окремій комірці Google Colaboratory та містити текстову комірку для опису дії.

4. Вивід даних мають бути належним чином організовано.

5. Всі роботи мають бути виконані в Google Colaboratory та посилання надіслані на пошту викладача.

Питання для самоперевірки

1. Що являє собою бібліотека NumPy і для чого вона використовується в Python?

2. Як створити одновимірний і двовимірний масиви (arrays) у NumPy?

3. Які основні операції дозволяється виконувати з масивами NumPy? Наведіть приклади.

4. Які відмінності між масивами NumPy та списками Python?

5. Які функції NumPy використовуються для роботи з масивами: знаходження максимуму, мінімуму, середнього значення, сортування тощо?

6. Як створити DataFrame в Pandas зі списку, словника або іншого DataFrame?

7. Які основні операції з даними можна виконати з DataFrame в Pandas?

8. Як відібрати дані за допомогою умовного індексування (conditional indexing) в Pandas?

9. Як об'єднати два DataFrame за допомогою операції злиття (merge) або з'єднання (join) в Pandas?

10. Для чого використовується бібліотека Matplotlib?

11. Які існують типи графіків (plots) у Matplotlib і як створити кожен з них?

12. Як змінити осі координат графіка (axes) в Matplotlib?

13. Як створити графік з декількома підграфіками (subplots) у Matplotlib?

14. Як вставити підписи осей, заголовок та легенду до графіка у Matplotlib?

Довідкові матеріали

1. Офіційна сторінка Python. URL: https://www.python.org/

2. Програмування для всіх: основи Python. URL: https://prometheus.org.ua/course/course-v1:Michigan+PFE101+2023_T3

3. Python: Структури даних. URL: https://prometheus.org.ua/course/course-v1:Michigan+PDS101+2023_T3

4. Розробка та аналіз алгоритмів. Частина 1. URL: https://prometheus.org.ua/course/course-v1:KPI+Algorithms101+2015_Spring

5. Burkov A. Machine Learning Engineering. URL: http://www.mlebook.com/wiki/doku.php

Посилання на літературу

1. Алгоритми та структури даних. Навчальний посібник. Київ, НТУУ «КПІ ім. Ігоря Сікорського», 2022. 215 с.

2. Васильєв О. Програмування мовою Python. Навчальна книга – Богдан, 2019. 504 с.

3. Маттес Е. Пришвидшений курс Python. Львів, Видавництво Старого Лева, 2021. 600 с.

4. VanderPlas J. Python Data Science Handbook. URL: https://jakevdp.github.io/ PythonDataScienceHandbook/

5. Shai Shalev-Shwartz, Shai Ben-David Understanding machine learning. Cambridge University Press, 2014. 499 p.

ЛАБОРАТОРНА РОБОТА 6. Поняття штучного інтелекту. Машинне навчання. Лінійна регресія

Теоретичні відомості

Незважаючи на надзвичайно велику зацікавленість цією тематикою як прикладною наукою, як суспільством й бізнесом так і філософією, єдиного визначення цього поняття на сьогодні не має. Розглядуваний науковий напрям є, більшою мірою, філософським питанням ніж суто технологічною проблемою. На даному етапі розвитку науки та техніки під терміном «штучний інтелект» варто розуміти галузі науки та техніки, які досліджують, проектують, розробляють та вивчають властивості комплексів (інформаційних, програмно-алгоритмічних і апаратних), результатом дій яких є висновки та умовиводи, аналогічні до когнітивних, мисленнєвих та комунікаційних процесів людини[8]. У контексті цього трактування будемо розглядати і термін «машинне навчання».

Вперше думку про машини, які вміють навчатися, висловив таку думку в статті «Computing Machinery and Intelligence»[9] у 1950 році. Одним із перших, хто дав визначення поняттю «машинне навчання» був A. L. Samuel. На його думку машинне навчання є процесом, результатом якого умовна машина (комп'ютер) здатна демонструвати поведінку, на яку вона явно не була запрограмована[10].

На даний час одним із найбільш точним означенням процесу здобуття знань обчислювальною системою представлено Томом

[8] Нікольський Ю. В. Системи штучного інтелекту: навчальний посібник. Львів: «Магнолія-2006», 2015 р. – 279 с. – С. 11.
[9] Turing A. Computing Machinery and Intelligence / A. Turing // Mind. – 1950. –Vol. 59. – P. 433 – 460.
[10] Samuel A. L. Some Studies in Machine Learning Using the Game of Checkers / A. L. Samuel // IBM Journal. – July 1959. – P. 210–229.

Мітчелом. *Вважається, що певна комп'ютерна програма навчається на основі досвіду E по відношенню до деякого класу задач T, яка вимірюється за допомогою P, якщо якість P розв'язків задач T покращується у процесі отримання досвіду E*[11].

Машинне навчання як важливий науковий і прикладний напрям штучного інтелекту багато запозичив із психології людини. Зокрема розрізняють такі підходи до формування необхідних знань моделі штучного інтелекту.

1. Навчання із керівником (supervised learning) – означає, що існує один набір даних, на яких система здобуває досвід, та другий, що дозволяє перевірити ефективність моделі.

2. Навчання без керівника (unsupervised learning) – це створення моделі машинного навчання на основі єдиного масиву даних інформації і передбачає самостійне виявлення структури даних. Такий підхід можна розглядати як підготовчий етап для реалізації концепції навчання із керівником.

3. Навчання із підкріпленням (reinforcement learning) – передбачає безпосередньо взаємодію системи із певним середовищем. Модель здобуває досвід аналізуючи це середовище і при цьому саме середовище виступає засобом перевірки ефективності моделі.

Для побудови різних моделей машинного навчання важливим моментом є підбір необхідних даних. Проте не завжди можна скористатися якісно опрацьованим матеріалом. Зокрема, для певних характеристик (наприклад числових) може міститися інформація у різних одиницях вимірювання, відрізнятися порядком, також у самих даних можуть бути помилки, пропуски або ж інші некоректні значення, які заважатимуть якісному опрацюванні інформації. Отже маємо

[11] Mitchell T. M. Machine Learning. McGraw-Hill, 1997. – 432 p. – P. 2.

важливу проблему підготовки даних для створення моделей машинного навчання.

Шляхи вирішення вказаної проблемної ситуацію можна наступним чином.

1. Відновлення пропущених даних за допомогою основних статистичних величин.

2. Виконання нормалізації даних.

Важливо зазначити наступне: єдиного підходу, який дозволяв однозначно покращити структуру даних не існує. Все залежить від типу опрацьованих даних, кількості помилок та досвіду фахівця, який виконує програмування певної моделі машинного навчання.

Ще одним методом відновлення даних є використання метрики[12].

Говорять, що непорожня множина X наділена метрикою (між елементами множини X задано відстань), якщо кожній парі елементів x і y з X ставлять у відповідність число $\rho(x,y)$*, яке задовольняє такі умови.*

1) $(\forall x,y \in X)\ \rho(x,y) > 0$ *та* $\rho(x,y) = 0 \Leftrightarrow x = y$.

2) $(\forall x,y \in X)\ \rho(x,y) = \rho(y,x)$ *– аксіома симетрії* .

3) $(\forall x,y,z \in X)\ \rho(x,y) \le \rho(x,z) + \rho(z,y)$ *– аксіома трикутника* .

До найбільш часто застосовуваних метрик у машинному навчанні відносять.

1. Евклідова метрика. Нехай задано два об’єкти $X(x_1,x_2,\ldots,x_n)$ та $Y(y_1,y_2,\ldots,y_n)$, тоді $\rho(X,Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$.

2. Метрика Манхеттен. Для заданих $X(x_1,x_2,\ldots,x_n)$ і $Y(y_1,y_2,\ldots,y_n)$ відстань задаватиметься наступним чином $\rho(X,Y) = \sum_{i=1}^{n}|x_i - y_i|$.

[12] Томусяк А. А., Трохименко В. С. Математичний аналіз. Вінниця: ВДПУ, 1999. – 488 с. – С. 370.

3. Max-метрика. $\rho(X,Y) = \max\left\{\left|x_i - y_i\right|\right\}_{i=1}^{n}$.

Алгоритм з використанням метрики має наступний вигляд.

1. Визначаємо об'єкт з однією з властивостей, яка містить пошкоджену інформацію.

2. Виключаємо цю властивість з розгляду.

3. Використовуючи значення інших властивостей, знаходимо об'єкт, який найбільше схожий на той, для якого відновлюємо дані. Найближчим буде об'єкт з найменшим значенням за визначеною метрикою.

4. Заміщуємо відсутню або пошкоджену інформацію значенням найбільш близького об'єкта чи використовуємо формулу для обчислення відповідного значення.

$$P(A) = \frac{1}{\sum_{i=1}^{n} \rho(A, A_i)} - \sum_{i=1}^{n} \frac{P(A_i)}{\rho(A_i)}$$

Ще одним кроком для якісної підготовки даних для побудови моделі машинного навчання є нормалізація. Найчастіше для нормалізації даних використовують такі формули.

Standart Scaling (Z-score njrmalization).

$$z = \frac{x_i - \overline{\mu}}{s}$$

x_i – поточне значення даних.

$\overline{\mu}$ – середнє значення вибірки.

s – стандартне відхилення.

MinMax Scaling

$$x_{norm} = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}$$

x_i – поточне значення даних.

x_{min} – найменше значення у вибірці.

x_{max} – найбільше значення у вибірці.

Для нормалізації даних також часто використовують логарифмування, особливо тоді коли опрацьовуються дуже великі значення властивостей.

Одним з важливих напрямків застосування машинного навчання на основі підготовлених даних є прогнозування числових даних, зокрема за допомогою лінійної регресії. Уперше термін регресія був введений англійським дослідником Френсісом Гальтоном[13], коли досліджував зріст 205-ти подружніх пар і 930-ти їх дітей. Він встановив, що якщо зріст когось з батьків відрізняється від загальної середньої величини по сукупності., то для їх нащадків це відхилення уже може бути меншим. Фактично фізичні ознаки регересують (по суті повертаються) до свого середнього значення. Діти дуже високих дітей можуть бути нижчеми за батьків, а діти дуже низьких навпаки – вищими за батьків.

У сучасних умовах лінійна регресія використовується тоді, коли необхідно встановити лінійну залежність між змінними. У машинному навчанні така залежність дозволить передбачати майбутні значення на основі створеної моделі.

Замість того, щоб чітко програмувати комп'ютер для виконання конкретної задачі, у машинному навчанні використовуються методи, що дозволяють системам адаптуватися до нової інформації, вдосконалювати свої результати та визначати закономірності без явного програмування.

Розглянемо приклад використання бібліотек мови програмування Python для створення моделі лінійної регресії. Для її побудови скористаємося бібліотекою SciPy.

Приклад 6.1.

```
#Підключаємо необхідні бібліотеки
# Для того щоб вивести необхідні графіки
import matplotlib.pyplot as plt
# Власне бібліотека, яку використовуватимемо для
```

[13] Чекотовський Е. В., Потапова М. Ю. Френсіс Гальтон: життя та внесок у розвиток статистичної науки. Статистика України, 1, 2011. С. 66 – 71.

```
    #побудови моделі лінійної регресії
    from scipy import stats

    #Задамо значення, на основі яких будемо
    #створювати модель
    x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
    y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

    #Будуємо модель лінійної регресії
    slope, intercept, r, p, std_err_slope =
stats.linregress(x, y)

    #Створюємо фунцію для передбачення на основі
    #нових даних
    def PredictFunc_LineReg(x, slope, intercept):
      return slope * x + intercept

    mymodel = []
    for item in x:
      mymodel.append(PredictFunc_LineReg(item,
slope, intercept))

    #Представляємо все на графіку
    plt.scatter(x, y)
    plt.plot(x, mymodel)
    plt.show()
```

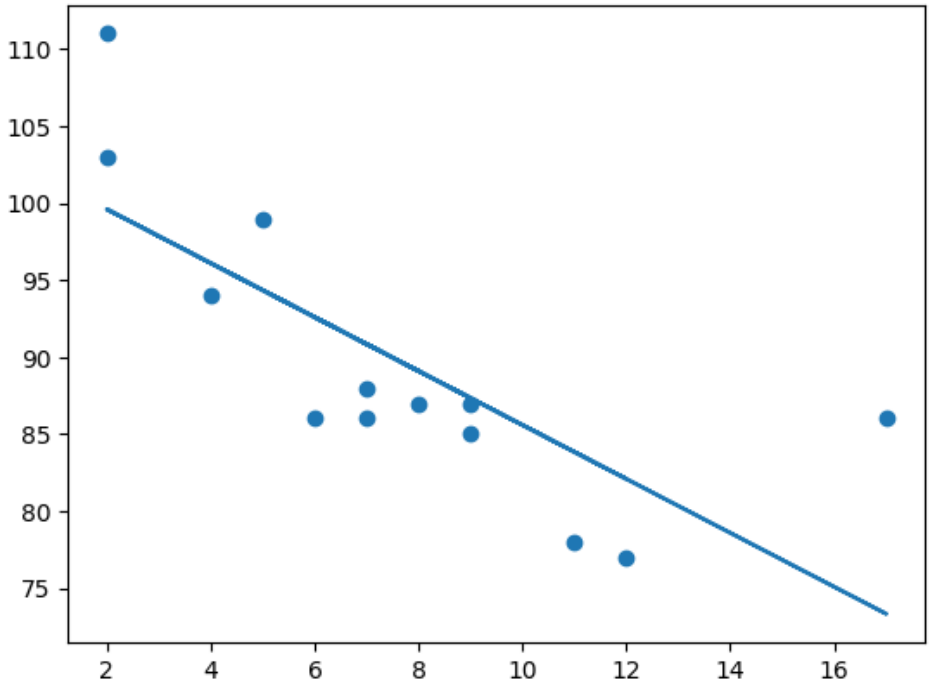

Рис. 6.1. Результат роботи програми, представленої у прикладі 6.1.

Розберемо більш детально запис **`slope, intercept, r, p, std_err_slope = stats.linregress(x, y).`**

Де змінні відповідають за такі значення

`slope` – тангенс кута нахилу прямої до осі абцис.

`intercept` – зміщення по вертикалі лінії.

`r` – коефіцієнт кореляції Пірсона. Квадрат r дорівнює коефіцієнту детермінації.

`p` – значення p для перевірки статистичних гіпотез. Для цього випадку нульовою гіпотезою є те, що нахил дорівнює нулю, альтернативною – кут нахилу прямої до осі X відмінний від 0. Перевірка відбувається з використанням тесту Вальда.

`std_err_slope` – стандартна помилка оціненого нахилу.

Цікво, що для передбачення значення на основі нових даних достатньо лише **`slope`** та **`intercept`**. Нижче подано власне розрахунок необхідного значення (приклад 6.2.).

Приклад 6.2.

```
print(PredictFunc_LineReg(12, slope, intercept))
```

Результат виконання передбачення для числа 12.

```
82.09050772626932
```

Досить часто значення залежить лінійно не від одного значення, а від багатьох різних ознак у визначеному наборі даних. У такому випадку для передбачення потенційного значення на основі вже нових даних варто скористатися множинною лінійною регресією.

Розглянемо приклад побудови моделі множинної лінійної регресії з використанням бібліотеки **Scikit-learn**. Зокрема визначимо залежність кількості викиду вуглекислого газу від маси авто і об'єму двигуна.

Приклад 6.3.

```
#Підключаємо потрібні бібліотеки
import pandas as pd
import numpy as np
from sklearn import linear_model

#Завантажуємо необхідні дані
df = pd.read_csv("./sample_data/car.csv")

print(df.head())

#Вибираємо незалежні та залежні ознаки
# Незалежні ознаки
X = np.array(df[['Weight', 'Volume']])
# Залежна ознака
y = np.array(df['CO2'])
```

```
# print(X)

# Створюємо саму модель
#Створюємться відповідний об'єкт
mlr = linear_model.LinearRegression()
mlr.fit(X, y)

#Застосовуємо модель для визначення значення на основі нових значень
print("Для авто вагою {} кг і двигуном {} cm3 викиди CO2 становитимуть {}г.".format(2000, 1800, mlr.predict([[2000, 1800]])[0]))
```

Результат виконання передбачення для числа 12.

	Car	Model	Volume	Weight	CO2
0	Toyoty	Aygo	1000	790	99
1	Mitsubishi	Space Star	1200	1160	95
2	Skoda	Citigo	1000	929	95
3	Fiat	500	900	865	90
4	Mini	Cooper	1500	1140	105

```
Для авто вагою 2000 кг і двигуном 1800 cm3 викиди CO2 становитимуть 108.84607738170558г.
```

Розглянемо більш детальніше клас **linear_model.LinearRegression**. Для нього характерні такі методи:

fit (X, y[, sample_weight]) – створити лінійну модель.

`get_metadata_routing()` – надає маршрутизацію метаданих цього об’єкта.

`get_params([deep])` – дає змогу отримати параметри для цієї оцінки.

`predict (X)` – дозволяє прогнозувати за допомогою лінійної моделі.

`score (X, y[, sample_weight])` – повертає коефіцієнт детермінації прогнозу.

`set_fit_request(*[, sample_weight])` – метадані запиту, які передаються методу fit.

`set_params (**params)` – дозволяє встановити параметри цієї оцінки.

`set_score_request (*[, sample_weight])` – метадані запиту передані в метод оцінки.

Завдання до лабораторної роботи

1. Відповідно до Вашого варіанту виконайте наступні завдання:

Варіант 1.

1) Розмістіть на Вашому Google Disk файл бази даних з інформацією про дослідження **(insurance.csv)**.

2) Адаптуйте дані до використання лінійної регресії.

3) Розділіть дані на тестову та навчальну вибірки.

4) Побудуйте модель машинного навчання, яка дозволяє прогнозувати вартість страхування в залежності від віку, статі та факту, що людина палить.

5) Порівняйте спрогнозовані результати із реальними даними. Знайдіть різницю між реальними даними тестової вибірки та прогнозованими, зокрема обчисліть, який відсоток становить абсолютне значення похибки до реальних даних.

6) Побудуйте гістограму відсоткових значень похибки для кожного прогнозованого результату.

7) Напишіть висновок про ефективність побудованої моделі.

Варіант 2.

1) Розмістіть на Вашому Google Disk файл бази даних з інформацією про дослідження **(insurance.csv)**.

2) Адаптуйте дані до використання лінійної регресії.

3) Розділіть дані на тестову та навчальну вибірки.

4) Побудуйте модель машинного навчання, яка дозволяє прогнозувати вартість страхування в залежності від віку, індексу маси тіла та регіону, де людина проживає.

5) Порівняйте спрогнозовані результати із реальними даними. Знайдіть різницю між реальними даними тестової вибірки та прогнозованими, зокрема обчисліть, який відсоток становить абсолютне значення похибки до реальних даних.

6) Побудуйте гістограму відсоткових значень похибки для кожного прогнозованого результату.

7) Напишіть висновок про ефективність побудованої моделі.

Варіант 3.

1) Розмістіть на Вашому Google Disk файл бази даних з інформацією про дослідження **(insurance.csv)**.

2) Адаптуйте дані до використання лінійної регресії.

3) Розділіть дані на тестову та навчальну вибірки.

4) Побудуйте модель машинного навчання, яка дозволяє прогнозувати вартість страхування в залежності від кількості дітей, індексу маси тіла та факту, що людина палить.

5) Порівняйте спрогнозовані результати із реальними даними. Знайдіть різницю між реальними даними тестової вибірки та

прогнозованими, зокрема обчисліть, який відсоток становить абсолютне значення похибки до реальних даних.

6) Побудуйте гістограму відсоткових значень похибки для кожного прогнозованого результату.

7) Напишіть висновок про ефективність побудованої моделі.

Варіант 4.

1) Розмістіть на Вашому Google Disk файл бази даних з інформацією про дослідження **(insurance.csv)**.

2) Адаптуйте дані до використання лінійної регресії.

3) Розділіть дані на тестову та навчальну вибірки.

4) Побудуйте модель машинного навчання, яка дозволяє прогнозувати вартість страхування в залежності від віку, регіону, де вона проживає та факту, що людина палить.

5) Порівняйте спрогнозовані результати із реальними даними. Знайдіть різницю між реальними даними тестової вибірки та прогнозованими, зокрема обчисліть, який відсоток становить абсолютне значення похибки до реальних даних.

6) Побудуйте гістограму відсоткових значень похибки для кожного прогнозованого результату.

7) Напишіть висновок про ефективність побудованої моделі.

Варіант 5.

1) Розмістіть на Вашому Google Disk файл бази даних з інформацією про дослідження **(insurance.csv)**.

2) Адаптуйте дані до використання лінійної регресії.

3) Розділіть дані на тестову та навчальну вибірки.

4) Побудуйте модель машинного навчання, яка дозволяє прогнозувати вартість страхування в залежності від того факту, що людина палить, статі та індексу маси тіла.

5) Порівняйте спрогнозовані результати із реальними даними. Знайдіть різницю між реальними даними тестової вибірки та прогнозованими, зокрема обчисліть, який відсоток становить абсолютне значення похибки до реальних даних.

6) Побудуйте гістограму відсоткових значень похибки для кожного прогнозованого результату.

7) Напишіть висновок про ефективність побудованої моделі.

2. Кожна дія має виконуватися у окремій комірці Google Colaboratory та містити текстову комірку для опису дії.

3. Самі дані треба можна взяти за посиланням: https://www.kaggle.com/datasets/awaiskaggler/insurance-csv.

Розшифровка полів даних

age – вік бенефіціара

sex – стать

bmi – індекс маси тіла, що вказує наскільки високою чи низькою відносно зросту є вага тіла, ідеальне значення розміщене у інтервалі від 18,5 до 24,9

children – число дітей, яке покриває страхування здоров'я

smoker – вказує чи палить бенефіціар

region – територія проживання бенефіціара в США,

charges – вартість страхування

4. Вивід даних мають бути належним чином організовано.

5. Всі роботи мають бути виконані в Google Colaboratory та посилання надіслані на пошту викладача.

Питання для самоперевірки

1. Що таке машинне навчання і які підходи до його визначення?

2. Що таке лінійна регресія і в яких випадках її застосовують?
3. Як відбувається навчання моделі лінійної регресії?
4. Як обирати змінні для моделі лінійної регресії?
5. Які методи використовуються для відбору значущих змінних?

Довідкові матеріали

1. Офіційна сторінка Python. URL: https://www.python.org/

2. Програмування для всіх: основи Python. URL: https://prometheus.org.ua/course/course-v1:Michigan+PFE101+2023_T3

3. Python: Структури даних. URL: https://prometheus.org.ua/course/course-v1:Michigan+PDS101+2023_T3

4. Розробка та аналіз алгоритмів. Частина 1. URL: https://prometheus.org.ua/course/course-v1:KPI+Algorithms101+2015_Spring

5. Burkov A. Machine Learning Engineering. URL: http://www.mlebook.com/wiki/doku.php

Посилання на літературу

1. Алгоритми та структури даних. Навчальний посібник. Київ, НТУУ «КПІ ім. Ігоря Сікорського», 2022. 215 с.

2. Васильєв О. Програмування мовою Python. Навчальна книга – Богдан, 2019. 504 с.

3. Маттес Е. Пришвидшений курс Python. Львів, Видавництво Старого Лева, 2021. 600 с.

4. VanderPlas J. Python Data Science Handbook. URL: https://jakevdp.github.io/ PythonDataScienceHandbook/

5. Shai Shalev-Shwartz, Shai Ben-David Understanding machine learning. Cambridge University Press, 2014. 499 p.

ЛАБОРАТОРНА РОБОТА 7. Лінійні класифікатори.

Теоретичні відомості

Поряд із задачами прогнозуванням числового значення за допомогою регресії, у машинному навчанні, важливе місце посідають моделі, які займаються класифікацією (розподілом по групах у відповідності до наперед вказаних ознак). Одним з найпростіших варіантів задачі класифікації, коли певний об’єкт за сукупністю ознак відноситься до одного з двох класів (бінарна класифікація). Для такого розподілу можуть застосовати різні математичні підходи, зокрема класифікація може відбуватися за допомогою лінійної гіперплощини (у двовимірному просторі це буде пряма).

Щоб створити модель, яка буде виконувати лінійну бінарну класифікацію об’єктів, необхідно вибірку розділити на дві частини тренувальну (дозволяє навчити модель) та тестову (надає змоги перевірити ефективність системи). Для визначення якісних характеристик моделі машинного навчання при класифікації об’єктів застосовують ряд описових характеристик та матрицю помилок (або англійською мовою *confusion matrix*).

Матриця помилок дозволяє найбільш точно описати ефективність класифікації для машинного навчання.

$$\begin{array}{c} \\ 1 \\ 2 \\ \vdots \\ C \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & \ldots\ldots & C \end{array} \\ \begin{pmatrix} n_{11} & n_{12} & & \\ n_{21} & n_{22} & & \\ & & \ddots & \\ & & & n_{CC} \end{pmatrix} \end{array}$$

Елементи, які розміщені по діагоналі відповідають коректній роботі певного класифікатора, тоді як решта елементів вважаються неправильно класифікованими.

Розглянемо ситуацію для двох класів. Сформуємо матрицю так, як показано нижче.

		Істинна вибірка	
		0	1
Передбачувана вибірка	0	TN	FN
	1	FP	TP

Значення, які представлені у таблиці розшифровуються наступним чином.

TP або *True Positive* – правильно класифікований об’єкт до класу 1.

TN або *True Negative* – правильно віднесений об’єкт до класу 0.

FN або *False Negative* – не правильно віднесений об’єкт до класу 0.

FP або *False Positive* – не правильно віднесений об’єкт до класу 1.

При цьому важливо оперувати такими параметрами, які дозволяють визначити ефективність створеного класифікатора. До них варто віднести такі.

Точність (precision) $P = \frac{TP}{TP + FP}$.

Повнота (recall) $\mathrm{Re} = \frac{TP}{TP + FN}$

Загальна точність (accurancy) $A = \frac{TP + TN}{TP + TN + FP + FN}$

$$F - value = \frac{1}{\frac{1}{P} + \frac{1}{\mathrm{Re}}}$$

Помилкова правильна класифікація $FPR = \frac{FP}{TN + FP}$.

Ймовірність виявлення об’єкта, заданого класу $TPR = \frac{TP}{TP + FN}$.

Алгоритми для створення моделей машинного навчання для вирішення задач класифікації містять багато бібліотек, зокрема: Scikit-Learn, PyTorch, TensorFlow тощо. Більшість з них створенні на основі нейронних мереж, для яких задається функція активації. Найчастіше, для бінарної класифікації, за таку функцію обирають логістичну регресію. Це пов'язано з особливістю моделі логістичної функції. Так як вона набуває значень на проміжку від 0 до 1 не включно, а це дуже зручно використовувати саме для бінарної класифікації.

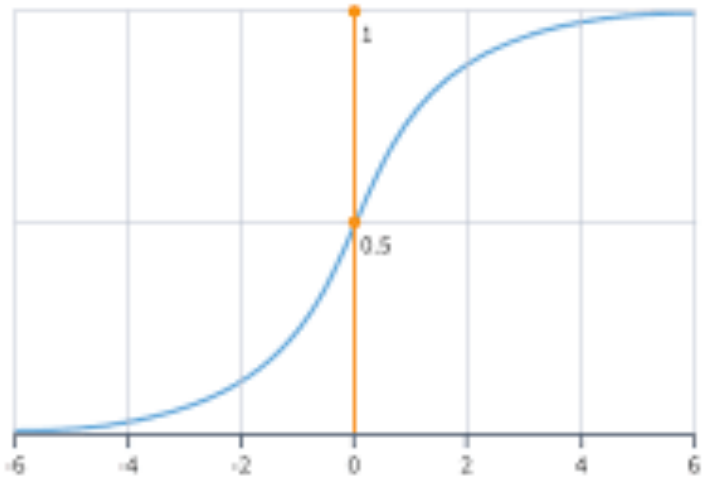

Рис. 7.1. Область значень логістичної регресії.

Розглянемо більш детально приклад створення моделі бінарної класифікації за допомогою бібліотеки Scikit-Learn. Для цього використаємо набір даних, у яких необхідно класифікувати банкноти (фальшиві або не фальшиві) на основі визначених параметрів (приклад 7.1).

Приклад 7.1.

```
#Завантажуємо необхідні модулі
import pandas as pd
import numpy as np
from sklearn.linear_model import SGDClassifier
```

```
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import StandardScaler

    #Завантажуємо необхідні дані
    df = pd.read_csv("./sample_data/banknote-authentication.csv")
    print(df.head())

    # Розділяємо незалежні та залежні дані
    X = np.array(df[["variance", "skew", "curtosis", "entropy"]])
    Y = np.array(df["class"])
    # print(Y)

    # Вибираємо тренувальну та тестову вибірки
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.25)

    # Виконуємо стандартизацію даних
    scaler = StandardScaler().fit(X_train)

    X_train = scaler.transform(X_train)
    X_test = scaler.transform(X_test)

    # Створюємо модель класифікатора
    clf = SGDClassifier(loss="log_loss")
    clf.fit(X_train, y_train)
```

```
    from sklearn.metrics import confusion_matrix,
classification_report

    y_predict = clf.predict(X_test)

    print(confusion_matrix(y_test, y_predict))
    print(classification_report(y_test,   y_predict,
target_names= ["0", "1"]))
    print(clf.coef_)
```

Важливо розуміти, що існують багато задач, де необхідно віднести певний предмет не одного з двох класів, а перевірити його приналежність до однієї з трьох, чотирьох чи більше категорій.

Одним з алгоритмів класифікації елементів, у якому застосовується метрика, є метод k-найближчих сусідів (або скорочено k-NN). Його суть полягає у наступному. Нехай існує певний об’єкт А, що певні характеристики. Вважатимемо цей об’єкт А відноситься до певного класу, якщо серед k найближчих сусідів знаходиться найбільше об’єктів цього класу.

Загалом алгоритм має таку послідовність кроків.

1. Визначення параметра k: Спершу задаємо значення параметра k, яке вказує, скільки найближчих сусідів буде використовуватися для визначення класу.

2. Визначення типу метрики відстані: Далі визначається тип метрики, за допомогою якої будуть визначатися сусіди класифікованого об’єкта.

3. Пошук найближчих сусідів: Обчислюємо відстані між об’єктом, який класифікуємо та рештою з навчального набору даних.

4. Вибір *k* найближчих сусідів: Об'єкти, що мають найменші значення відстані до нового об'єкта, обираються як k найближчих сусідів.

5. Визначення результату: У випадку класифікації, клас нового об'єкта визначається на основі більшості класів серед k найближчих сусідів.

Наведемо приклад його реалізації з використанням бібліотеки Scikit-Learn (приклад 7.2.). Для демонстрації методу kNN класифікації використаємо датасет Iris (опис квіток ірисів). Він створений у 1936 році ботаніками Едгаром Андерсоном і Рональдом Фішером.

Датасет Iris містить інформацію про 150 спостережень із чотирма ознаками для кожного спостереження. Зокрема.

Довжина чашолистка (sepal length) в сантиметрах.

Ширина чашолистка (sepal width) в сантиметрах.

Довжина пелюстки (petal length) в сантиметрах.

Ширина пелюстки (petal width) в сантиметрах.

Кожне спостереження в датасеті відноситься до одного з трьох видів ірису: Iris setosa, Iris versicolor та Iris virginica.

Приклад 7.2.

```
#Завантажуємо необхідні дані
import pandas as pd
import numpy as np

df = pd.read_csv("./sample_data/iris.csv")
print(df.head(150))

#Розділяємо незалежні та залежні змінні
X = np.array(df[["sepal.length", "sepal.width", "petal.length", "petal.width"]])
```

```
    Y = np.array(df["variety"])

    #Створюємо тренувальну та тестову вибірку
    from sklearn.model_selection import
train_test_split

    X_train, X_test, Y_train, Y_test =
train_test_split(X, Y, test_size = 0.25)

    #Виконуємо стандартизацію даних
    from sklearn.preprocessing import StandardScaler
    scaler = StandardScaler().fit(X_train)
    X_train = scaler.transform(X_train)
    X_test = scaler.transform(X_test)

    #Створюємо модель класифікації
    from sklearn.neighbors import
KNeighborsClassifier
    #тут вказуємо кількість сусідів,
    #за якими будемо відносити об'єкт
    #до того чи іншого класу
    knn = KNeighborsClassifier(n_neighbors=3)
    knn.fit(X_train, Y_train)

    #Перевіряємо ефективність на
    #основі тестової вибірки
    Y_predict = knn.predict(X_test)
```

```
#Оцінюємо точність моделі
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score

print(confusion_matrix(Y_test, Y_predict))
print(classification_report(Y_test, Y_predict, target_names=["Setosa", "Versicolor", "Virginica"]))
print("Accurancy: ", accuracy_score(Y_test, Y_predict))
```

Результат виконання програми

```
[[14  0  0]
 [ 0 11  1]
 [ 0  1 11]]

              precision    recall  f1-score   support

      Setosa       1.00      1.00      1.00        14
  Versicolor       0.92      0.92      0.92        12
   Virginica       0.92      0.92      0.92        12

    accuracy                           0.95        38
   macro avg       0.94      0.94      0.94        38
weighted avg       0.95      0.95      0.95        38

Accurancy:  0.9473684210526315
```

Завдання до лабораторної роботи

1. Відповідно до Вашого варіанту виконайте наступні такі види робіт.

2. Розмістіть на Вашому Google Disk файл бази даних з інформацією про дослідження.

3. Проаналізуйте структуру даних.

4. Адаптуйте дані, щоб було можливим створити найбільш ефективну модель класифікації.

5. Розділіть дані на тестову та навчальну вибірки.Побудуйте модель машинного навчання, яка дозволяє прогнозувати придатність до споживання води.

6. Виведіть на екран характеристики ефективності моделі: матрицю помилок, загальну точність, повноту.

7. Зробіть висновки про ефективність створеної моделі.

Варіанти завдань.

1. **Dataset water_potability.csv**

https://www.kaggle.com/datasets/uom190346a/water-quality-and-potability

2. **Dataset wdbc.csv**

https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data

3. **Dataset haberman.csv**

https://www.kaggle.com/datasets/gilsousa/habermans-survival-data-set

4. **Dataset DiabetesHealthDatasetAnalysis.csv**

https://www.kaggle.com/datasets/rabieelkharoua/diabetes-health-dataset-analysis

5. **Dataset Cleveland_Heart_Disease_Dataset.csv**

https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cleveland.data

8. Кожна дія має виконуватися у окремій комірці Google Colaboratory та містити текстову комірку для опису дії.

9. Вивід даних мають бути належним чином організовано.

10. Всі роботи мають бути виконані в Google Colaboratory та посилання надіслані на пошту викладача.

Питання для самоперевірки

1. Які задачі можна вирішувати за допомогою лінійних класифікаторів?
2. Наведіть приклади задач бінарної класифікації.
3. Які основні моделі лінійних класифікаторів існують?
4. Як здійснюється навчання моделі логістичної регресії?
5. Як оцінити якість моделі бінарної класифікації?

Довідкові матеріали

1. Офіційна сторінка Python. URL: https://www.python.org/
2. Програмування для всіх: основи Python. URL: https://prometheus.org.ua/course/course-v1:Michigan+PFE101+2023_T3
3. Python: Структури даних. URL: https://prometheus.org.ua/course/course-v1:Michigan+PDS101+2023_T3
4. Розробка та аналіз алгоритмів. Частина 1. URL: https://prometheus.org.ua/course/course-v1:KPI+Algorithms101+2015_Spring
5. Burkov A. Machine Learning Engineering. URL: http://www.mlebook.com/wiki/doku.php

Посилання на літературу

1. Алгоритми та структури даних. Навчальний посібник. Київ, НТУУ «КПІ ім. Ігоря Сікорського», 2022. 215 с.

2. Васильєв О. Програмування мовою Python. Навчальна книга – Богдан, 2019. 504 с.

3. Маттес Е. Пришвидшений курс Python. Львів, Видавництво Старого Лева, 2021. 600 с.

4. VanderPlas J. Python Data Science Handbook. URL: https://jakevdp.github.io/ PythonDataScienceHandbook/

5. Shai Shalev-Shwartz, Shai Ben-David Understanding machine learning. Cambridge University Press, 2014. 499 p.

ЛАБОРАТОРНА РОБОТА 8. Методи кластеризації.

Теоретичні відомості

Ще одним важливим прикладом задач, які вирішуються методами машинного навчання є кластеризація. Постановка задачі є наступна. *Нехай дано множину об'єктів X. Необхідно розбити її на групи (певні кластери), які міститимуть подібні елементи.*

Кластеризацію визначають як модель машинного навчання без керівника (unsupervised learning). Досить часто її використовують для: спрощення структури для подальшої обробки даних (у певній зменшеній групі подібних об'єктів набагато легше виявити залежності); зменшення об'єму даних (заміна групи ознак однією, еталонною характеристикою дозволяє зменшити розмірність); пошук викидів та аномалій.

Розбиття множини об'єктів на кластери є ітеративним процесом, який вимагає знань предметної області та людського судження для коригування даних і параметрів моделі для досягнення бажаного результату. А отже класичні показники ефективності моделі, характерні для моделей класифікації, не можуть бути використані для моделей кластеризації. Для оцінки ефективності таких програм більшою мірою використовують якісні показники, які дають, переважно, відповідь на такі типові питання:

Наскільки ефективно можна інтерпретувати отримані результати?

Чи корисний результат кластеризації для бізнесу або науки?

Чи дізналися фахівці нову інформацію або виявили нові закономірності в даних, про які не знали до кластеризації?

Вибір алгоритмів кластеризації залежить від багатьох параметрів, але найважливішими з них є такі:

Параметри, необхідні для побудови моделі.

Можливість масштабування.

Метрика, яка використовується для обчислення метрики.

Розглянемо найбільш часто використовувані алгоритми для вирішення задач кластеризації.

Одним з найбільш поширеним, проте не найбільш ефективним, є k – means. Він має наступну послідовність кроків.

1. Користувач передбачає (задає) кількість кластерів.

2. Випадковим чином задаємо центроїди (умовні скупчення об’єкта). Кількість центроїдів має відповідати кількості задних кластерів.

3. Обчислюємо відстань між кожною точкою та визначеними центрами та відносимо зазначену точку до одного з кластерів.

4. Перераховуємо середнє значення центроїда на основі всіх призначених точок даних, що дозволяє змінити положення центроїда.

5. Запускаємо новий етап перегляду всіх точок, з тією відмінністю, що центоїди уже змінили своє розміщення.

6. Процес відбувається до того часу поки положення центроїдів не буде змінюватися або сам процес не зупинить користувач.

Його головними недоліками є те, що вибір кількості кластерів залежить від користувача і його досвіду роботи з числовими даними, а також для нього складно визначати кластери умовно «неправильної» форми.

Приклад 8.1 показує особливості створення моделі для кластеризації точок у двовимірному просторі.

Приклад 8.1.

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

```
    data = pd.read_csv('./Sample.csv')

    x = np.array(data['x'])
    y = np.array(data['y'])
    #У цьому прикладі дані не варто нормалізувати,
    # оскільки вони всі цілі числа
    X_train_test_norm = np.array(data)
    from sklearn.cluster import KMeans

    #Імпортуємо метод k-Means та
    #створюємо модель кластеризації.
    kmeans = KMeans(n_clusters = 4,random_state = 1,
n_init='auto')
    kmeans.fit(X_train_test_norm)

    #Виводимо точковий графік кластеризації
    x = np.array(data['x'])
    y = np.array(data['y'])

    colors = kmeans.labels_

    plt.scatter(x, y, c=colors)
    plt.show()
```

На відміну від алгоритму k-means, алгоритм MeanShift не вимагає від користувача вказувати кількості кластерів. Алгоритм сам автоматично визначає кількість кластерів, що є досить великою

перевагою над k-means. Він є досить ефективним, якщо дійсно присутні закономірностях у даних.

MeanShift базується на оцінці щільності ядра. Подібно до алгоритму k-means, алгоритм MeanShift ітеративно відносить кожну точку даних до найближчого центроїда кластера, який ініціалізується випадковим чином, і кожен новий (уточнений) центроїд ітеративно переміщується в просторі на основі того, де знаходиться найбільша кількість точок, тобто мода (мода - це найвища щільність точок даних в регіоні, в контексті алгоритму MeanShift).

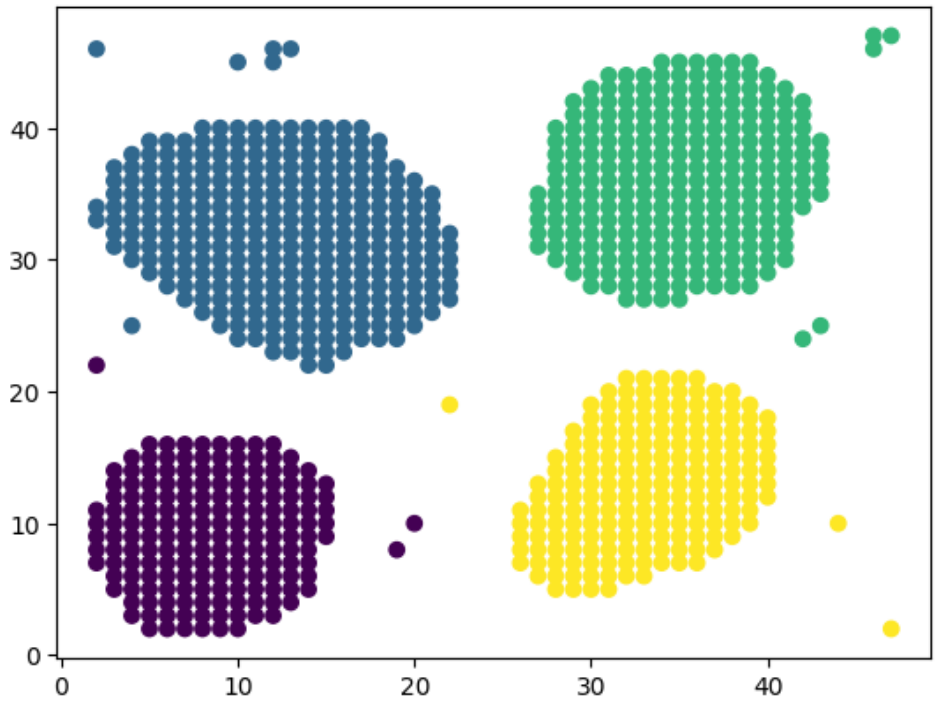

Рис. 8.1. Результат роботи програми, представленої у прикладі 8.1.

Загалом алгоритм можна описати такими кроками.

1. Вибирається довільно точка та окреслюється певна область навколо неї.

2. Обчислюється середнє значення всіх точок всередині цієї області та визначається центр нового центроїда.

3. Окреслюємо таку ж область навколо нового центроїда і алгоритм дій повторюємо.

4. Кроки повторюються до тих пір поки зміна центра буде мінімальною або ж буде зупинена користувачем. Всі точки які попадають у виокремлену область оголошуються певним кластером та вилучаються з огляду, а сам алгоритм має розпочати пошук нового потенційного кластеру.

5. Всі дії відбуваються доти доки дані не будуть розділені на клестери.

Зазвичай, застосування такого алгоритму ефективне саме на початкових етапах дослідження і може бути скоригованим за допомогою k-means.

Застосування алгоритму наведено у прикладі 8.2.

Приклад 8.2.

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

data = pd.read_csv('./Sample.csv')

x = np.array(data['x'])
y = np.array(data['y'])

#Сформуємо вибірку
X_train_test = np.array(data)

#Застосовуємо алгоритм MeanShift
from sklearn.cluster import MeanShift

mean_shift_cluster = MeanShift(bandwidth=8)
```

```
mean_shift_cluster.fit(X_train_test)

#Виводимо точковий графік кластеризації
x = np.array(data['x'])
y = np.array(data['y'])

colors = mean_shift_cluster.labels_

plt.scatter(x, y, c=colors)
plt.show()
```

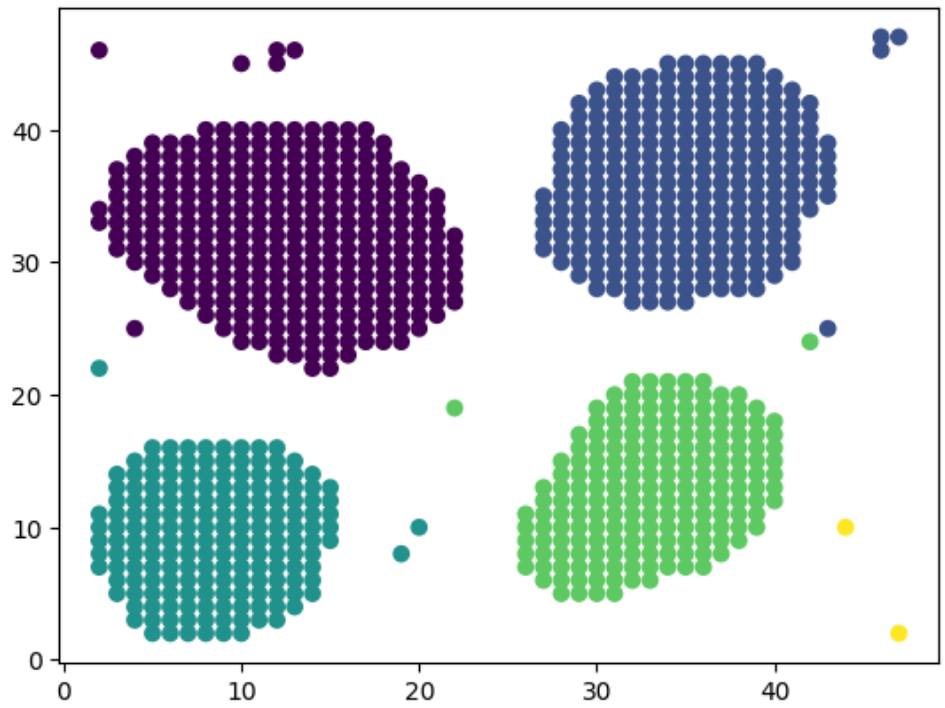

Рис. 8.2. Результат роботи програми, представленої у прикладі 8.2.

DBSCAN або Density-Based Spatial Clustering of Applications with Noise – це алгоритм кластеризації, який працює на основі припущення, що кластери є щільними групами об’єктів у певному визначеному регіоні, які розділяються областями з меншою щільністю. Він вимагає введення від користувача лише двох параметрів:

1) радіусу кола, яке буде створено навколо кожної точки даних;

2) мінімальна кількість точок, які повинні міститися у середині кола, щоб ця точка даних була класифікована як точка визначеного кластеру.

При роботі алгоритму всі точки діляться (об’єкти) кластеру на три важливі категорії.

1. Основні точки: Точка даних є основною, якщо в околі певного радіуса міститься не менше заданого мінімуму точок цього ж кластеру.

2. Гранична точка: Точка називається граничною, якщо у заданому радіусом околі міститься менше ніж мінімально необхідна кількість точок кластеру.

3. Викид: Точка є викидом, якщо вона не є основною або граничною.

Основні кроки алгоритму DBSCAN наступні.

1. Задається радіус околу та мінімальна кількість точок (min_points), яка у ньому має міститися щоб позначити точку як основну.

2. Вибираємо певну точку і перевіряємо кількість її сусідів, які потрапляють у заданий окіл.

3. Якщо кількість точок більша або рівна параметру min_points, то тоді цю точку оголошуємо основною для умовного кластера А, а її сусідів оголошуємо такими, що належать до кластеру А.

4. Якщо кількість точок менша за заданий параметр min_points і всі вони класифіковані як такі, що належать кластеру А, то тоді цю точку оголошуємо граничною для умовного кластера А.

5. Якщо кількість сусідів рівна 0, то таку точку одразу оголошуємо як викид(шум).

6. Далі переглядаємо позначених сусідів та рахуємо кількість точок, які міститимуться у околі і класифікуватимемо їх.

7. Якщо ж кількість точок більша або рівна параметру min_points, але жодна з них не належить до класу А, то тоді цю точку оголошуємо основною для нового умовного кластера Б, а її сусідів оголошуємо такими, що належать до кластеру Б і вже тепер переглядаємо лише точки кластера Б.

8. Алгоритм припиняє роботу, якщо всі точки будуть віднесені до певного кластера або позначені як викид.

Як і інші вже описані алгоритми DBSCAN вже реалізований у бібліотеці Scikit-Learn (приклад 8.3.).

Приклад 8.3.

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

data = pd.read_csv('./Sample.csv')

x = np.array(data['x'])
y = np.array(data['y'])

#Задаємо набір даних.
X_train_test = np.array(data)

#Створюємо власне саму модель
from sklearn.cluster import DBSCAN

clasters_DBSCAN = DBSCAN(eps = 2, min_samples=3)
clasters_DBSCAN.fit(X_train_test)
```

```
# Графічне представлення результатів
x = np.array(data['x'])
y = np.array(data['y'])

colors = clasters_DBSCAN.labels_

plt.scatter(x, y, c=colors)
plt.show()
```

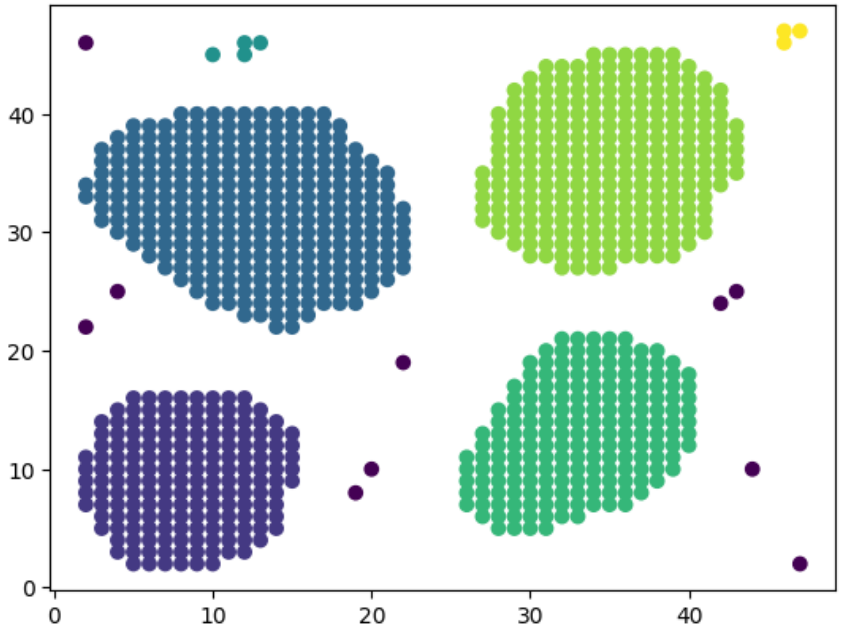

Рис. 8.3. Результат роботи програми, представленої у прикладі 8.3.

Окрім вже наведених алгоритмів кластеризації визначених даних застосовують і інші підходи. Зокрема ієрархічна кластеризація дозволяє будувати ієрархію кластерів. В окремих випадках для оптимізації обчислювальних потужностей і подальшого аналізу даних використовують методи зменшення розмірності (наприклад метод головних компонент).

Завдання до лабораторної роботи

1. Відповідно до Вашого варіанту виконайте наступні завдання:

1) Розмістіть на Вашому Google Disk файл бази даних з інформацією про дослідження.

2) Адаптуйте дані, щоб було можливим створити найбільш ефективну алгоритм кластеризації (відповідно до Вашого варіанта).

3) Виконайте кластеризацію даних.

4) Кожен окремий кластер збережіть у окремий dataFrame.

5) Побудуйте точкову діаграму, яка відображатиме виокремлені кластери кольором та міститиме легенду діаграми, що позначатиме кожен кластер.

6) Зробіть висновки про ефективність створеної моделі.

Варіанти завдань.

1. **Dataset Lab_08_Var_01.csv**

Алгоритм кластеризації. **k - means**

2. **Dataset Lab_08_Var_02.csv**

Алгоритм кластеризації. **MeanShift**

3. **Dataset Lab_08_Var_03.csv**

Алгоритм кластеризації. ***DBSCAN***

4. **Dataset Lab_08_Var_04.csv**

Алгоритм кластеризації. **k - means**

5. **Dataset Lab_08_Var_05.csv**

Алгоритм кластеризації. ***DBSCAN***

2. Кожна дія має виконуватися у окремій комірці Google Colaboratory та містити текстову комірку для опису дії.

3. Вивід даних мають бути належним чином організовано.

4. Всі роботи мають бути виконані в Google Colaboratory та посилання надіслані на пошту викладача.

Питання для самоперевірки

1. Що таке кластеризація і в яких випадках її використовують?
2. Які основні методи кластеризації існують?
3. Як працює метод k-means?
4. Що таке метод DBSCAN і як він відрізняється від k-means?
5. Як оцінити якість кластеризації?
6. Як використовуються кластери у реальних прикладних задачах?
7. Які проблеми можуть виникнути при використанні методів кластеризації?

Довідкові матеріали

1. Офіційна сторінка Python. URL: https://www.python.org/
2. Програмування для всіх: основи Python. URL: https://prometheus.org.ua/course/course-v1:Michigan+PFE101+2023_T3
3. Python: Структури даних. URL: https://prometheus.org.ua/course/course-v1:Michigan+PDS101+2023_T3
4. Розробка та аналіз алгоритмів. Частина 1. URL: https://prometheus.org.ua/course/course-v1:KPI+Algorithms101+2015_Spring
5. Burkov A. Machine Learning Engineering. URL: http://www.mlebook.com/wiki/doku.php

Посилання на літературу

1. Алгоритми та структури даних. Навчальний посібник. Київ, НТУУ «КПІ ім. Ігоря Сікорського», 2022. 215 с.

2. Васильєв О. Програмування мовою Python. Навчальна книга – Богдан, 2019. 504 с.

3. Маттес Е. Пришвидшений курс Python. Львів, Видавництво Старого Лева, 2021. 600 с.

4. VanderPlas J. Python Data Science Handbook. URL: https://jakevdp.github.io/ PythonDataScienceHandbook/

5. Shai Shalev-Shwartz, Shai Ben-David Understanding machine learning. Cambridge University Press, 2014. 499 p.

ДОДАТОК
Рекомендована література та інформаційні джерела

Основна:

1. Алгоритми та структури даних. Навчальний посібник. Київ, НТУУ «КПІ ім. Ігоря Сікорського», 2022. – 215 с.

2. Васильєв О. Програмування мовою Python. Навчальна книга – Богдан, 2019. 504 с.

3. Мосіюк О. О. Штучний інтелект: вступ до машинного навчання : навч.-метод. посіб. Житомир : Вид-во ЖДУ ім. Івана Франка, 2019. 76 с.

4. Іванченко Г. Ф. Система штучного інтелекту: навчальний посібник К.: КНЕУ, 2011. 382 с.

5. Креневич А. П**.** Алгоритми і структури даних. Київ.: ВПЦ «Київський Університет», 2021. 200 с.

6. Нікольський Ю. В. Системи штучного інтелекту: навчальний посібник. Львів: «Магнолія-2006», 2015 р. 279 с.

7. Стратегія розвитку штучного інтелекту в Україні / за заг. редакцією А. І. Шевченка. Київ: ІПШІ, 2023. 305 с.

8. Ямпольський Л. С. Системи штучного інтелекту в плануванні, моделюванні та управлінні :підручник для студентів вищ. навч. закл. К.:ДП Видавничий дім «Персонал», 2011. 544 с.

9. Poole D., Mackworth A. Artificial intelligence. Foundations of computational agents. URL: http://artint.info/

Додаткова:

1. Burkov A. Machine Learning Engineering. URL: http://www.mlebook.com/wiki/doku.php

2. Leskovec J., Rajaraman A., Ullman J. Mining of massive datasets / J. Leskovec, A. Rajaraman, J. Ullman.URL: http://infolab.stanford.edu/~ullman/mmds/book.pdf.

3. Shai Shalev-Shwartz, Shai Ben-David Understanding machine learning. Cambridge University Press, 2014. 499 p.

4. Stevens E., Antiga L., Viehmann Th. Deep Learning with PyTorch. Manning Publications Co., 2020. 522 p. URL: https://isip.piconepress.com/courses/temple/ece_4822/resources/books/ Deep-Learning-with-PyTorch.pdf

5. VanderPlas J. Python Data Science Handbook. URL: https://jakevdp.github.io/ PythonDataScienceHandbook/

Інтернет ресурси:

1. Автоматична система оцінювання для програм. URL: https://eolymp.com/uk.

2. Машинне навчання. URL: https://courses. prometheus.org.ua/courses/ IRF/ML101/2016_T3 /about .

3. Основи програмування. URL: https://courses.prometheus.org.ua/courses/ KPI/ Programming101/ 2015_T1/about

4. Офіційна сторінка Anaconda. URL: https://www.anaconda.com/

5. Офіційна сторінка Python. URL: https://www.python.org/

6. Towards Data Science. URL: https://towardsdatascience.com/

7. Kaggle. URL: https://www.kaggle.com/