ЗМІСТ

Передмова

Незважаючи на бурхливий розвиток сучасних інформаційних технологій, зростання обчислювальних потужностей та місткості засобів збереження даних, все ж існують задачі, які достатньо важко розв'язати за допомогою класичного програмування. До них варто віднести такі як: медична діагностика, засоби комп'ютерного зору, інтерпретації, розпізнаванню та перекладу текстів, фільтрація спаму чи створення рекомендаційної системи тощо. Над вирішенням переважної більшості із них працюють науковці, які розробляють теоретичні засади для створення моделей штучного інтелекту (зокрема і машинного навчання) та практичною перевіркою ефективністю. На даний час, це один із самих пріоритетних напрямів розвитку інформаційно-комунікаційних технологій. Саме тому програма підготовки майбутніх учителів інформатики передбачає вивчення такого предмету як «Системи штучного інтелекту».

Отже метою представленого навчально-методичного посібника є ознайомлення майбутніх педагогічних фахівців із принципами роботи моделей машинного навчання та особливостями і засобами їх розробки за допомогою мови програмування Python.

Загалом навчально-методичний посібник складається з із двох частин, завдань до лабораторних робіт і додатків.

У першій частині розкриваються особливості створення програм за допомогою Python 3.

Друга частина присвячена розкриттю базових понять машинного навчання; питань, пов'язаних із відновленням пропущених даних, їх нормалізацією. Наведено також алгоритми для побудови моделей лінійної регресії, кластеризації та класифікації даних, дерев рішень тощо.

Додатки містять посилання на важливі ресурси, присвячені штучному інтелекту та статистичні дані для виконання лабораторних робіт.

Навчально-методичний посібник, що пропонується, може бути рекомендованим студентам ЗВО, а саме студентам фізико-математичних факультетів спеціальності 014.09 Середня освіта (Інформатика), вчителям інформатики, слухачам курсів підвищення професійної кваліфікації в процесі професійного вдосконалення.

ЧАСТИНА 1.
МОВА ПРОГРАМУВННЯ PYTHON

Історія мови програмування Python

На даний час існує велика кількість мов програмування, які використовуються для вирішення різних видів завдань із багатьох сфер життєдіяльності людини. C, C++, C#, Fortran, Php, Java, JavaScript, Pascal, Lisp, Prolog, Python, Ruby, Swift, Go, Objective-C, Perl є лише невеликим переліком засобів, які використовуються у системному програмуванні, web-розробці, створенні ігор, адмініструванні систем, математичних та статистичних обрахунках, робототехніці та створенні штучного інтелекту. Серед такого різноманіття варто виділити ряд мов, які, останнім часом, посідають провідні місця у різних рейтингах та опитуваннях. Якщо оперувати статистичними даними[1], то найпопулярнішими мовами програмування являються: Java, JavaScript, C#, Php, Python та C++. При цьому вже декілька років поспіль Java очолює рейтинги, а решта змінюють свої місця відповідно до уподобань користувачів.

Проте із розвитком систем штучного інтелекту, значного поширення набув Python. Унікальна мова програмування, яка відзначається якісним записом коду та використовується у багатьох галузях.

Мова програмування Python була створена програмістом Гвідо ван Росумом у кінці 80-х років минулого століття. У 1991 році автором був опублікований код, розробленої ним мови програмування. Завдяки своїй лаконічності Python почав стрімко набувати популярність і на даний час використовується у багатьох сферах (опрацювання xml/html файлів, робота із http-запитами, програмування графічних інтерфейсів, виконання математичних та наукових обчислень, програмування машинного навчання та комп'ютерного зору, створенні візуальних ефектів тощо). Важливою особливістю мови є наявність можливостей для обробки зображень, аудіо та відео файлів. Як наслідок, інтерпретатор Python вбудований у такі відомі системи створення тривимірного контенту як Maya та Houdini, а вільно поширюваний 3D додаток Blender повністю написаний за допомогою зазначеної мови

[1] Рейтинг мов програмування 2019: JavaScript майже зрівнялася з Java, популярність Go знижується [Електронний ресурс]. – Режим доступу: https://dou.ua/lenta/articles/langu age-rating- jan-2019/.

програмування і дозволяє створювати спеціалізовані розширення для цієї програми.

Сьогодні Python – це високорівнева мова програмування, яка продовжує активно розвиватися. Він підтримує декілька, найбільш популярних на даний час, парадигми програмування (структурний, функціональний, об'єктно-орієнтований підходи до створення програм). Важливою особливістю мови є й те, що вона повністю відповідає вимогам, які висуваються Американським національним інститутом стандартів.

Python – це інтерпретована мова програмування. Тобто виконання коду відбувається послідовно, за допомогою інтерпретатора. Інтерпретатор – це спеціалізована програма, яка транслює написаний Вами код у зрозумілий для операційної системи набір інструкцій (так званий «байт-код»), що потім можуть виконуватися процесором. На даний час існує велика кількість реалізацій для інтерпретатора зазначеної мови. Еталонною, або ж класичною реалізацією інтерпретатора Python є CPython, який написаний на мові C. Окрім вказаного, є і інші версії. Так, до основних версій інтерпретатора Python відносять Jython (реалізація мови програмування для віртуальної машини Java із можливістю використання Java-аплетів) та Iron Python, призначений для платформи Microsoft.NET і повністю написаний на C#. Існує версія PyPy, розробники якої ставлять за мету збільшення швидкості виконання коду, реалізуючи концепцію Justin-time complation.

Основні версії Python

Тепер опишемо основні віхи у розвитку цієї мови програмування. Як уже зазначалося, у 1991 році (в лютому місяці) Гвідо ван Росум виклав у відкритий доступ вихідний код інтерпретатора Python, а також його основні інструкції. Він помітив його як версію 0.9.0. Перший ключовий реліз, із поміткою 1.0, був представлений у січні 1994 року. До жовтня 2000 р. було представлено ще два проміжних реліза: 31.12.1997 р. вийшла версія Python 1.5 та 05.09.2000 р. презентовано Python 1.6.

Наступним ключовим кроком у розвитку мови програмування Python стало представлення версії 2.0., яка вийшла 16 жовтня 2000 р. Розробники додали можливість працювати із списковими включенням (запозичили із функціональних мов програмування SETL та Haskell), а також додано до інтерпретатора «систему збору сміття» із підтримкою циклічних посилань. Починаючи з цієї версії, майже кожні 2 роки виходили оновлення до другої версії Python.

Важливим етапом становлення розглядуваної мови програмування є створення її третьої версії у 2008 році. Вона виправила фундаментальні помилки, але внесені зміни порушили сумісність між 2-гою та 3-тьою редакцією. Варіант 2.7.X залишився для підтримки вже створеного програмного забезпечення, у той же час пріоритет був відданий розробці та удосконаленню саме третього видання. На даний час самим актуальним релізом Python 2 є мова програмування із індексом 2.7.16 від 4 березня 2019 р., а самим новим представником Python 3 – 3.7.3 від 25 березня 2019 р.

«Третій пайтон» успадкував ту ж філософію створення програм, що й його попередники, але основний пріоритет був відданий принципу: «повинен існувати один і, бажано, тільки один спосіб виконати певну дію». У той же час Python залишився мультипарадигмальною мовою програмування, який дозволяє вибирати структурний, функціональний чи об’єктно-орієнтований підхід до програмування.

На даний час Python 3 є однією із затребуваніших мов програмування серед фахівців із різних сфер діяльності для вирішення різноманітних задач (створення сайтів, опрацювання графічної інформації, програмування моделей машинного навчання і комп’ютерного зору тощо).

Інтегровані середовища розробки для Python

Базовим середовищем для розробки програм на мові Python є *IDLE – Integrated Development and Learning Environment*. Вона постачається разом із класичним інтерпретатором Python та може використовуватися для написання і запуску скриптів у таких популярних операційних системах як Windows, OS X та Linux-

дистрибутивах. IDLE може виконувати звичні для середовища розробки задачі: писати і переглядати код, редагувати його, «підсвічувати» іншим кольором ключові слова, запускати і відкладати програму тощо.

Eric Python IDE – це повноцінне інтегроване середовище, яке містить функції редагування і перевірку на ефективність та помилки коду написаного мовами Python та Ruby. Програма містить функції для перевірки синтаксису, браузер класів, має можливості спільного редагування програм та керування проектами, підтримка систем контролю версій Git, Mercurian і Subversion. Програма має багатомовний інтерфейс та поширюється із відкритою ліцензією.

PyDev являється спеціалізованим плагіном для розробки програм у багатофункціональному середовищі Eclipse. У ньому вбудовані можливості покрокового виконання програми та відслідковування значень змінних, інструментарій для рефакторингу. Важливим є і те, що у плагіні передбачена інтеграція із таким відомим фреймворком як Django, зокрема існує можливість виконувати його команди за допомогою спеціалізованих клавіш. Загалом PyDev є якісним середовищем та містить багато інших корисних функцій.

PyCharm є однією із найкращих середовищ для розробки програм за допомогою Python, JavaScript, CoffeScript, HTML/CSS, AngularJS, NodeJS тощо. У ньому передбачено: можливість інтегрованого модульного тестування, перевірку коду, вбудовану система контролю версій (наприклад Git), іструменти рефакторингу коду, продуману система навігації по проекту тощо. Особливо важливою перевагою PyCharm є підтримка та інтеграція із такими фреймворками як Django, web2py, Flask, що перетворює зазначену IDE в універсальний інструмент швидкої розробки web-сервісів. Середовище має версії для всіх найбільш поширених операційних систем.

На даний час існує декілька версій програми: комерційне використання, для студентів і викладачів, індивідуальне використання. Остання редакція має повноцінну підтримку всіх актуальних релізів та базових пакетів і модулів, але у той же час відсутня підтримка фреймворків. Комерційна та освітня зразки програми мають повну підтримку всіх функцій та засобів для створення програм.

Окрім названих інтегрованих середовищ розробки програмного забезпечення за допомогою мови програмування Python варто також згадати і такі: WingWare, KomodoIDE, Geany, PyScripter тощо.

Інтерфейс IDE Spyder

Для подальшого вивчення мови програмування Python 3 та одного із ключових напрямів штучного інтелекту – машинного навчання розглянемо вільнопоширюване середовище розробки IDE Spyder 3, який входить до дистрибутиву Anaconda. Розкриємо докладніше зазначений програмний засіб.

Anaconda є вільно поширюваним дистрибутивом, що включає такі мови програмування як Python та R, які використовуються для опрацювання великих масивів даних із різних галузей науки та техніки (машинне навчання, візуалізація даних, прогнозування майбутніх подій та поведінки об’єктів тощо). Він містить простий пакетний менеджер (Conda packege) і інтегроване середовище розробки (Spyder 3). На даний час дистрибутив Anaconda містить більше ніж 1400 пакетів для виконання багатьох задач при проведенні досліджень. Окрім цього реліз містить Anaconda Navigator – програму із графічним інтерфейсом, яка дозволяє керувати встановленням нових і оновленням вже розміщених модулів дистрибутиву. Серед таких варто виділити такі найбільш часто використовувані: Jupyter Notebook, QtConsole, Glueviz, Orange, RStudio, Spyder, Visual Studio Code). Його загальний вигляд представлений на рисунку 1.1.

Серед відомих бібліотек, які використовуються для аналізу даних та входять до дистрибутиву варто назвати такі: NumPy, SciPy, Numba, pandas, DASK. Модулі для візуалізації статистичних даних представлені такими популярними пакетами як: Bokeh, HoloViews, Datashader, matplotlib. Інструментарій для машинного навчання представлений модулями TensorFlow, Scikit Learn, H_2O.ai, theano тощо.

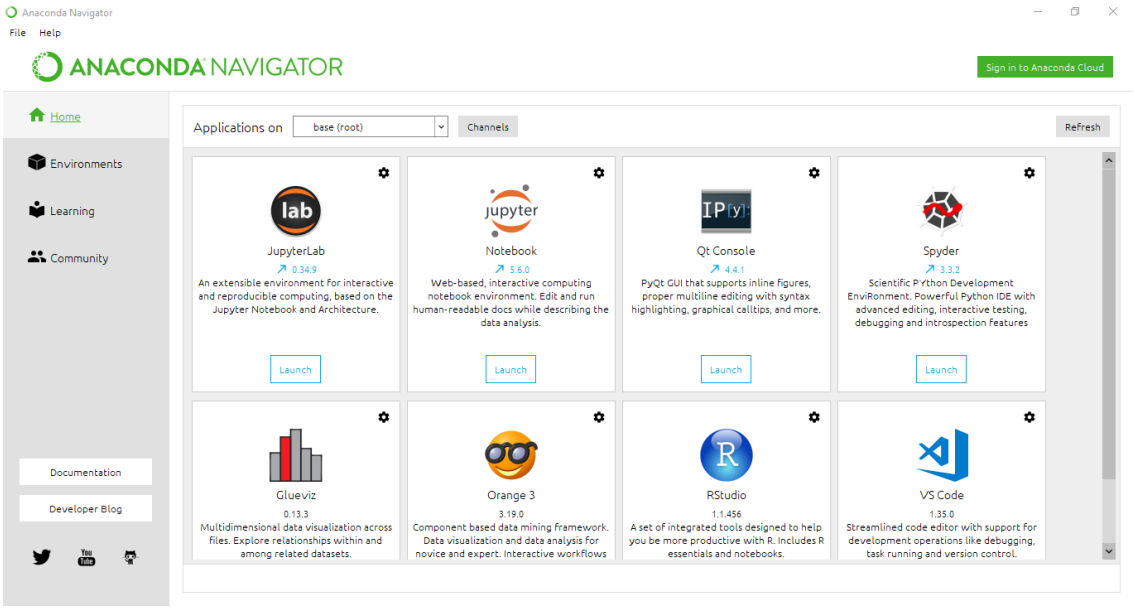

Рис. 1.1. Зовнішній вигляд Anaconda Navigator.

На даний час існують версії дистрибутиву Anaconda для Python 3.7 та Python 2.7. У подальшому орієнтуватимемося на версію, саме із інтерпретатором Python 3.7.

Розглянемо докладно інтерфейс середовища розробки додатків Spyder IDE. Загалом робоча область середовища розділена на такі важливі елементи (рис. 1.2).

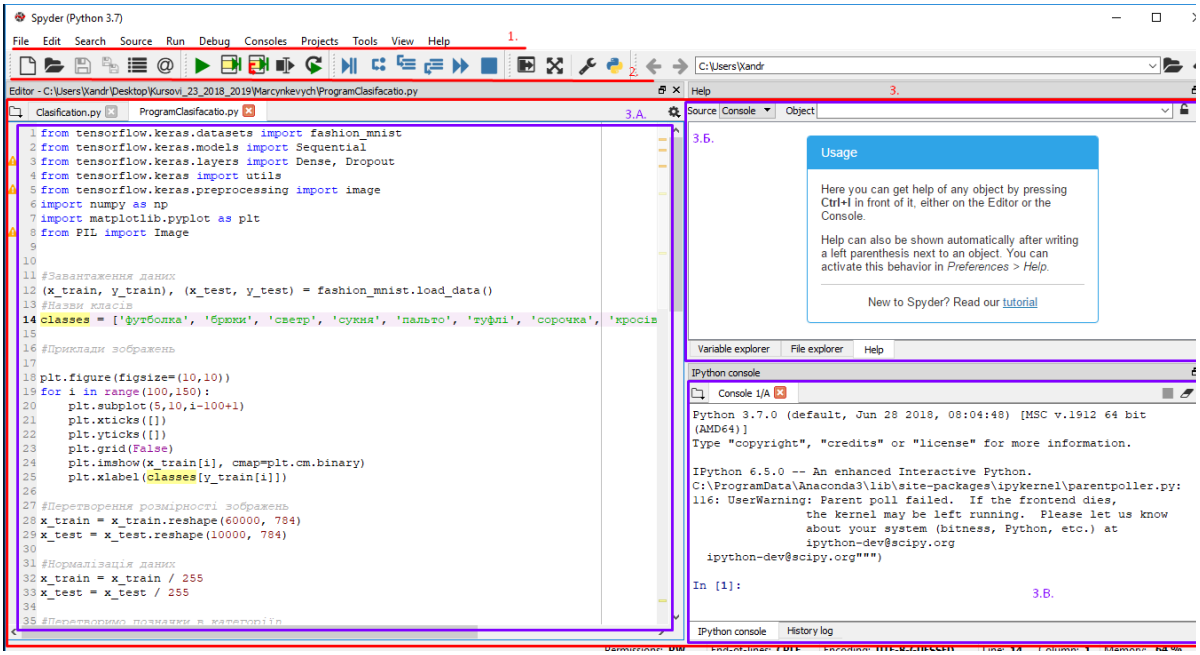

Рис. 1.2. Інтерфейс IDE Spyder 3.

1. Панель меню – включає всі списки команд для керування системою.

2. Панель швидкого доступу до основних команд містить найбільш застосовувані команди (створити новий файл проекту, відкрити вже створений документ, зберегти зміни у поточному файлі, зберегти зміни у всіх файлах, вибрати необхідний із списку відкритий файл, швидкий пошук файлів, запуск програми, покрокове виконання програми, керування відображенням панелей та налаштування середовищ тощо).

3. Робочий простір, який поділений на три робочі зони.

3.А. Редактор коду – має вбудоване кольорове виділення ключових слів та автозаповнення команд при натисканні клавіши Tab. У окремих вкладках відкриті робочі файли; наст

Панель, яка позначається на рисунку позначкою **3.Б.** містить вкладки Variable Explorer, File Explorer, Help. Вони дозволяють переглядати значення змінних, при покроковому виконанні програми, вести список відкритих для роботи файлів, а також тут розміщений доступ до всієї документації дистрибутиву Anaconda.

Останнє вікно **3.В.** є, власне консолью мови Python, що є дуже зручним для перегляду результатів виконання програми. А також присутній журнал історії, який дозволяє переглядати запити, які здійснюються під час інтерпретації програмного коду і, у разі помилки, аналізувати інформацію про неї.

Основи мови Python

Для кращого розуміння принципів побудови систем штучного інтелекту, зокрема і машинного навчання, важливим є знання основних конструкцій мови Python. На час написання навчально-методичного посібника актуальною є версія Python 3.7.3 і тому весь код програм, наведених у подальших прикладах, буде представлений саме для цього релізу мови, але з високою долею ймовірності можна допустити, що він буде виконуватися і для інших старіших варіантів Python 3.

Особливості синтаксису

Python є особливою мовою програмування, яка значно відрізняється від таких як C, C++, Java, Php, але, разом із тим, залишається достатньо простою для освоєння. Для нього характерно

дотримання ряду правил, які необхідно враховувати при написанні програми. Це зроблено для того, щоб ефективність сприймання коду програмістом була значно вищою. Наведемо ключові рекомендації, які стосуються правильного форматування коду.

1. Кінець рядка є кінцем інструкції.

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

Приклад 1.1.

```
s = 1; c = 7; print (s, b)
```

2. Вкладені інструкції поєднуються у блоки за величиною відступів.

Доречно зауважити, що відступ може бути будь-яким, але таким, що є однаковим у рамках одного вкладеного блоку інструкцій.

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

Приклад 1.2.

```
Інструкція:
␣␣␣␣Вкладений набір інструкцій
```

Більш детальний опис стандартів написання програм описуваною мовою представлено у специфікації PEP 8[2], а документування коду розкривається в PEP 257[3].

Операції із числами

Важливою особливістю мови програмування Python є здатність обраховувати цілі, дійсні та комплексні числа. Нижче наведемо перелік основних математичних операцій.

```
34 + 7 #Додавання
```

Результат виконання команди у консолі Python: 41

[2] Style Guide for Python Code [Електронний ресурс]. – Режим доступу: https://www. python. org/ dev/ peps/pep-0008/

[3] Docstring Conventions [Електронний ресурс]. – Режим доступу: https://www. python.org/dev/peps/pep-0257/

81 – 8 #Віднімання

Результат виконання команди у консолі Python: 73

35 * 3 #Множення

Результат виконання команди у консолі Python: 105

20 / 3 #Ділення

Результат виконання команди у консолі Python: 6.666666666666667

20 // 3 #Цілочисельне ділення

Результат виконання команди у консолі Python: 6

20 % 3 #Остача від ділення

Результат виконання команди у консолі Python: 2

20 ** 2 #Піднесення до степеня

Результат виконання команди у консолі Python: 400

3 ** 150 #Підтримка довгої арифметики

Результат виконання команди у консолі Python: 369988485035126972924700782451696644186473100389722973815184405301748249

Бітові операції із цілими числами мають такий вигляд.

x | y #Бітове «або»

x ^ y #Бітове виключне «або»

x & y #Бітове «і»

x << y #Зсув бітів уліво

x >> y #Зсув бітів управо

~x #Інверсія бітів

При чому *x* та *y* мають бути цілими.

У Python присутні також функції перетворення чисел із однієї системи числення в іншу. Для цього використовують такі функції.

int(число, основа системи числення) – функція приведення до цілого числа у десятковій системі числення, якщо основа числення не задана. Задати основу можна за допомогою цілого числа із інтервалу від 2 до 36.

Пристутні також і спеціалізовані функції.

bin(число) – запис числа у двійковій системі.

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

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

Дійсні числа

Для дійсних чисел у мові Python передбачено аналогічні дії, як і для цілих чисел. Про необхідно враховувати і такі неточності при виконні оперцій.

Приклад 1.3.

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

Результат виконання команди у консолі Python: 0.9999999999999

Також зазначена мова програмування підтримує операції із великими дійсними числами. Обмеження накладаються лише на обчислювальні потужності комп'ютера та обсяг оперативної пам'яті.

Комплексні числа

Окрім дій над цілими та дійсними числами у мові програмування Python передбачені операції із комплексними числами.

Приклад 1.4.

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

Результат виконання команди у консолі Python: 1 + 3j

Із прикладу видно, що уявна частина позначається літерою j, а не так як у математиці (літерою i). Це зроблено для того, щоб уникнути плутанини із змінною i, яка часто використовується при позначенні ітерацій циклу.

Комплексні числа можна додавати, віднімати, множити та ділити. Окрім цього у Python дозволяється виділяти спряжене комплексне число (x.conjugate()), дійсну частину (x.real), уявну частину (x.imag) та перевіряти рівність двох комплексних чисел.

Послідовність виконання арифметичних дій

Порядок виконання дій із числами має наступний вигляд. Найбільший пріоритет має піднесення до степеня, далі множення, ділення, цілочисельне ділення, остача від ділення. Найменший пріоритет мають дії додавання та віднімання.

Якщо у рядку, який описує арифметичний вираз, стоять дії із однаковим пріоритетом, то операції виконуються із ліва на право послідовно.

Пріоритетність дій завжди можна змінити за допомогою дужок.

Змінні

Як і будь-яка інша мова програмування, Python опрацьовує дані за допомогою змінних. Назва змінної використовується для того, щоб програма могла посилатися на збережене значення.

У Python змінна оголошується дещо відмінно від строго типізованих мов програмування. Наприклад у мові програмування C++ спочатку оголошується змінна і лише потім ініціалізується.

Приклад 1.5.

```
int a; /*Оголошення змінної цілого типу*/
a = 24; /*Ініціалізація змінної цілого типу*/
```

Приклад 1.6.

```
int b = 25; /*Оголошення та ініціалізація виконані у одному рядку, але у правильній послідовності*/
```

Але Python не має строгої типізації даних і цим він суттєво відрізняється від таких мов програмування як C, C++, Java, Pascal. Зокрема ініціалізація змінної відбувається відразу ж після присвоєння їй певного значення.

Приклад 1.7.

```
b = 41 # Оголошення та ініціалізація змінної b
```

Такий підхід до роботи із змінними називається динамічною типізацією і її використовують також такі мови програмування як JavaScript та Php.

Загалом інтерпретатор Python визначає тип даних автоматично, у той момент коли сама змінна була створена та ініціалізована. Розглянемо це процес докладніше.

1. Змінна створюється автоматично, коли їй у коді присвоюється певне значення. Всі подальші дії із цією змінною будуть змінювати значення, що асоціюється із визначеним іменем.

2. Змінні не мають жодної інформації про тип даних, пов’язаних із ними. Фактично вони є посиланнями на зазначені об’єкти у певний момент часу.

3. Під час виконання дії із змінною вона замінюється об’єктом, на який вона посилається, незалежно від того, що це за об’єкт. Варто зауважити, що перш ніж використати змінну її необхідно ініціалізувати, у іншому випаду буде виведено повідомлення про помилку і інтерпретатор зупинить програму.

Отже маємо, що всю інформацію про тип даних містять об’єкти, а змінна – це посилання на цей об’єкт. Тому наведений нижче код буде правильним.

Приклад 1.8.

```
a = 8
a = ”eight”
a = 0.125
```

Із прикладу видно, що змінна a посилається на різні об’єкти, які відрізняються типом даних. При цьому, як тільки відбулося переприсвоєння іншого значення, попередній об’єкт видаляється із пам’яті.

Приклад 1.9.

```
b = 0.25
b = 4 # видаляється об’єкт 0.25
b = “four” # видаляється об’єкт 4
b = [1,2,3,4] # видаляється об’єкт “four”
```

Це досягається завдяки тому, що об’єкт містить лічильник активних посилань на нього і як тільки їх кількість рівна 0, об’єкт видаляється із пам’яті та звільняє її для наступного використання. Цей процес, який виконує інтерпретатор називається збором «сміття» та дозволяє розробникам вільно використовувати об’єкти і не турбуватися за вивільнення пам’яті.

Наведені нижче приклади демонструють ці особливості мови програмування Python.

Приклад 1.10.

```
a = 6 # Об’єкт 6 має два посилання
b = a
```

Приклад 1.11.

```
x = 7 # Змінна y посилається на об’єкт 7,
y = a # тоді як змінна x посилається уже на
x = 8 # новостворений об’єкт 8
```

Логічні операції та операції порівняння

Для порівняння значень змінних використовуються вже знайомі із математики символи: > – знак більше, < – знак менше, >= – більше або дорівнює, <= – менше або дорівнює. Результатом виконання таких операцій буде два значення бульового типу (True або False).

Приклад 1.12.

```
5 > 4
```

Результат виконання команди у консолі Python: True

```
6 < 8
```

Результат виконання команди у консолі Python: True

```
6 >=7
```

Результат виконання команди у консолі Python: False

Окрім цього у Python присутні такі логічні операції: and – логічне «і», or – логічне «або», not – логічне заперечення.

Приклад 1.13.

```
(5 > 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

Для створення програм, яким необхідно взаємодіяти із користувачами необхідні функції, які дозволяють зчитувати необхідну інформацію із командного рядка або ж виводити результати роботи у консоль. У Python 3 стандартною функцією введення є input(), де в дужках можна вказати пояснення.

Приклад 1.14.

```
a = input (”Enter a”)
```

Варто зауважити, що у такому випадку змінній a буде присвоєно значення рядкового типу, що введене у консолі. Для того, щоб змінній a було присвоєне числове значення необхідно скористатися функціями переведення типів.

Приклад 1.15.

```
a = int(input (”Enter a”)) # Змінна a посилається на об’єкт цілого типу
a = float(input (”Enter a”)) # Змінна a посилається на об’єкт дійсного типу
```

Класичною функцією виведення даних у Python 3 є функція print(), де у дужках вказується текст, необхідний для представлення у консолі. Наступні приклади демонструють приклади застосування цієї інструкції.

Приклад 1.16.

```
a = 7
print(a) # Виводиться рядок із значенням 7
```

Результат виконання програми у консолі: 7

```
a = 8
print(“a =”, a) # Виводиться рядок із значенням a = 8
```

Результат виконання програми у консолі: a = 8

Коментування у Python 3

Будь-який код програми призначений не тільки для його виконання комп’ютером або певною обчислювальною системою, а й для того щоб інші спеціалісти, що працюють у команді розробників, мали змогу зрозуміти та використати його при написанні своєї частини програми. Для цього необхідно текст коду необхідно оформляти відповідним чином. Звичайно Python створений так, щоб забезпечити максимальне сприйняття інструкцій програми за допомогою оформлення спеціалізованих частин коду за допомогою відступів, але це не сприяє повноцінному розумінню частини коду, його призначення та функціоналу.

Щоб зробити код зрозумілішим або виключити частину коду із виконання при перевірці ефективності його роботи використовують коментування

У Python 3 передбачені як багаторядкові так і однорядкові коментарі. Однорядковий коментар позначається символом # і будь-який текст після нього не буде вважатиметься програмою.

Багаторядкові коментарі позначаються потрійними лапками.

Приклад 1.17.

```
#Це однорядковий коментар у Python 3
"""А це багаторядковий коментар
        у Python 3"""
```

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

Конструкція умовного оператора у мові Python 3 має такий вигляд.

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

Якщо умова правильна, то виконується перша серія команд, у протилежному випадку виконується друга серія інструкцій.

Дозволяється використовувати скорочену форму умови if.

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

Є у мові програмування Python 3 і конструкція, яка заміняє команду switch у таких мовах як C++ або ж JavaScript. По суті це є розширенням умовного оператора.

```
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
```

Використання інструкції продемонстровано у наступному прикладі.

Приклад 1.18.

```
if x < -6:
   print(x*x)
elif -6 <= x < 0:
   print(-x*x)
else:
   print(x*x-6)
```

Важливо зауважити, що вкладені блоки інструкцій виділяються за допомогою пробілів. Цей відступ має бути однаковим для всієї програми. Якщо ця умова не виконується, то можуть виникати помилки при запуску власне самої програми. Це спонукає програмістів слідкувати за оформленням коду і сприяє збереженню «читабельності» коду для інших фахівців.

При використанні умовного оператора важливим є використання складних умов. Для цього використовуються такі команди: and – логічне «і», or – логічне «або» та not – логічне заперечення. Наступний приклад демонструє їх застосування.

Приклад 1.19.

```
if (x > 6) or (x < 3):
   print(x*x)

if (x == 7) and (y == 8):
   print(x*y)
```

Цикли while та for

Розглянемо ще одну важливу інструкцію Python 3 для програмування – цикли.

While – це цикл із передумовою. Він є одним із самих універсальних інструментів у програмування. Конструкція команди має наступний вигляд.

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

Цикл while виконується до тих пір поки умова, яка вказана після службового слова істинна. У випадку, коли умова залишається істинною та не змінюється упродовж виконання кожної ітерації циклу маємо ситуацію «вічного циклу» або зацикленості програми. Щоб уникнути цього необхідно у тіло циклу вводити інструкцію, яка дозволить міняти змінну, що бере участь у формуванні умови.

Приклад 1.20.

```
i = 1
while i <= 10:
    print(i)
    i += 1
```

Дещо відрізняється застосування циклу for у Python та його конструкція від традиційної. Найчастіше for використовують для перебору елементів списку або ж рядка. В цей час будуть виконуватися команди, які формують тіло циклу.

Приклад 1.21.

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

Як результат виконання такого блоку команд, буде представлено п’ять чисел від 0 до 4 включно у кожному рядку консолі (приклад).

Приклад 1.22.

Результат виконання програми у консолі:

```
0
1
2
3
4
```

Функція range(5) формує список у порядку зростання 0, 1, 2, 3, 4.

Приклад застосування циклу for до перелічування елементів рядка наведений нижче.

Приклад 1.23.

```
a = "hello world"
for i in a:
   print(i)
```

У такому випадку отримаємо наступне.

Результат виконання програми у консолі:

```
h
e
l
l
o

w
o
l
d
```

Важливим для роботи із циклами є команди continue та break**.** Continue – дозволяє розпочати нову ітерацію циклу, не виконуючи команди, які розміщені після зазначеної інструкції.

Приклад 1.24.

```
for i in range(6):
   if i == 3:
      continue
   print(i)
```

Результатом виконання буде наступне.

```
0
1
2
4
5
```

Оператор break натомість достроково припиняє виконання циклу.

```
for i in range(6):
    if i == 3:
        break
    print(i)
```

У цьому випадку маємо.

```
0
1
2
```

Важливо зауважити, що у Python 3 дозволяється використання команди else для перевірки чи відбувся вихід із циклу без застосування інструкції break.

Приклад 1.25.

```
for i in "abcdefg":
    if i == "x":
        break
else:
    print("Літера x відсутня у рядку")
```

Як бачимо у прикладі вихід із циклу відбудеться стандартним чином, після того як буде переглянутий весь рядок. Оскільки у рядку відсутня літера x, то дострокового припинення циклу не буде. Тобто спрацює команда else і буде виведено повідомлення: «Літера x відсутня у рядку».

Лабораторна робота № 1.

Тема. Вступ до мови програмування Python.

Мета. Набути умінь та навичок розробки лінійних програм, програм із використанням умовних операторів та циклів за допомогою мови програмування Python.

Програмне забезпечення. Програмний комплекс Anaconda для версії Python 3.7.

Контрольні запитання.

1. Базові операції. Пріоритет дій.
2. Поняття змінної у мові Python.
3. Команди введення та виведення даних.

4. Особливості написання коду. Відступи.
5. Умовний оператор IF.
6. Цикли.

Завдання до лабораторного заняття

Рівень 1.

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.

1. Програмі на вхід подаються три значення: радіус орбіти першої планети, період обертання першої планети, радіус орбіти другої планети. Обчислити період обертання планети навколо Сонця.
2. Обчислити нарахування для депозитного вкладу S через n років при ставці річних p, при умові, що відбувається капіталізація коштів.
3. Задано значення температури у градусах Цельсія. Виведіть на екран цю ж температуру в градусах Кельвіна, Фаренгейта та Реомюра.
4. Автомобіль починає гальмувати. Обчисліть прискорення гальмування, якщо відомо початкову швидкість та час гальмування.
5. Обчислити суму та добуток цифр у чотирицифровому числі.

Рівень 3.

1. Перевірити чи є вказаний рік високосний.
2. Перевірити чи належить точка, яка задається своїми трьома координатами, внутрішній області сфери.
3. Перевірити чи існує трикутник із заданими сторонами.
4. Написати простий калькулятор на чотири арифметичні дії.
5. Визначити тип трикутника за двома його кутами.

Рівень 4.

1. Обрахувати n-те число Фібоначчі.
2. Вивести список всіх дільників числа m.
3. Реалізуйте серію ігор “Камінь, ножиці, папір” із комп’ютером. У фіналі виведіть статистику виграшів людини та машини.
4. Вивести 3 випадкових числа від 0 до 100 без повторень.
5. Згенерувати 10 серій із 0, 1 та 2 таких, що сума чисел була рівна 12.

Операції із рядками у Python 3

У Python 3 рядки є упорядкованими послідовностями символів, які дозволяють зберігати текстову інформацію. Щоб задати рядок у мові передбачено декілька можливостей: використання апострофів, звичних лапок та потрійних лапок. При цьому рядок, який записаний за допомогою апострофів не відрізняється від рядка заданого за допомогою лапок.

Приклад 1.26.

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

Окрім того для виведення певних символів використовується спеціальний знак екранізації \.

Приклад 1.27.

\n – переведення курсора на новий рядок.

\t – горизонтальна табуляція

\v – вертикальна табуляція

Якщо використати потрійні лапки, то мова програмування дозволяє записати багаторядковий форматований блок тексту.

Приклад 1.28.

```
text = """1 рядок
        2 рядок"""
```

Окрім способів задання рядкових даних у мові Python, варто розкрити операції, які дозволяється виконувати із змінними такого типу даних. До базових операцій із рядками відносять.

Конкатенація або об’єднання двох рядків.

```
S1 = "ryad 1"
S2 = "ryad 2"
S = S1 +’␣’ + S2
```

Результат.

"ryad 1 ryad 2"

Дублювання рядка.

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

Результат.

"ryadryadryad"

Отримання символу рядка за допомогою номера його індексу. Зауважимо, що індексація для рядків у мові програмування Python розпочинається із 0.

Приклад 1.29.

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

Результат.

"y"

Оператор 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 зріз. За замовчуванням перший індекс рівний 0, а другий – довжині рядка.

Приклад 1.30.

```
S = "ryad"
```

```
print(S[1,3])
```

Результат.

```
"ya"
```

Опишемо основні функції та методи для роботи із рядками.

len(S) – функція, яка визначає довжину рядка.

S.find(str, start, end) – пошук підрядка str у рядку S, починаючи із індексу start та завершуючи end, не включаючи останнього. За замовчуванням start дорівнює 0, а end довжині рядка. У випадку успішного виконання функції вказує на номер індексу першого співпадаючого символу або -1, якщо нічого не виявлено.

S.rfind(str, start, end) діє аналогічно до попереднього методу, але перевірка розпочинається із кінця виділеного інтервалу індексів.

S.index(str, start, end) та S.rindex(str, start, end) виконують аналогічну дію як і попередні функції, але із тією різницею, що у випадку негативної відповіді замість -1 повертає виключення ValueError.

S.replace(зразок, заміна) – у рядку заміняє визначений зразок символів на вказану у заміні послідовність символів.

S.split(розділювач) – розбиває рядок S на частини за вказаним розділювачем.

S.isdigit() – перевіряє чи вказаний рядок складається із цифр та повертає значення True, якщо так, у протилежному значенні представляє False.

S.isalpha() – перевіряє чи вказаний рядок складається тільки із літер та повертає значення True, якщо так, у протилежному значенні представляє False.

S.isalnum() – перевіряє чи вказаний рядок складається тільки із літер і цифр та повертає значення True, якщо так, у протилежному значенні представляє False.

S.islower() – дозволяє визначити чи рядок складається із літер, записаних у нижньому регістрі.

S.isupper() – схожа на попередню функцію, але перевіряє чи літери записані у верхньому регістрі.

S.isspace() – дозволяє визначити наявність спеціалізованих символів у рядку.

S.istitle() – надає змогу перевірити факт того, що слова у рядку будуть розпочинатися із великої літери.

S.upper() – перетворює всі символи нижнього регістру у верхній.

S.lower() – діє аналогічно до попередньої, але працює навпаки – всі великі літери перетворює у маленькі.

S.startswith(str) – визначає чи рядок S розпочинається із заданого зразка str.

S.endswith(str) – подібна до попередньої тільки перевіряє чи зазначений рядок завершується зразком str.

S.join(list) – об'єднує всі елементи списку list у один рядок.

ord(symbol) – повертає код символу symbol у таблиці ASCII.

chr(num) – перетворює код із ASCII у відповідний для нього символ.

S.count(str, start, end) – підраховує кількість входжень рядка str у рядку S починаючи із символу з номером індексу start та завершуючи значенням end не включно.

S.strip(symbol) – прибирає символи на початку та вкінці рядка.

Загалом це не всі функції, які використовуються для роботи із рядками, але вони є основними та дозволяють виконувати переважну кількість операцій із вказаним типом даних. Про решту функцій можна дізнатися на офіційному сайті Python 3.

Списки

Список – це упорядкована колекція об'єктів довільних типів, які можуть змінюватися у процесі виконання програми. Основною відмінністю від звичного масиву є те, що список може містити різні типи об'єктів.

Створити списки можна такими способами.

```
list('12345') #За допомогою команди list
```

Результат виконання команди: ['1','2','3','4','5']

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

Зауважимо, що нумерація елементів списку розпочинається із 0, а за допомогою індексу дозволяється доступ до самого елемента та його зміна.

Приклад 1.31.

```
lst = [1, 2, ’a’, ’b’]
print(lst [1])
```

Результат виконання скрипта буде 0, а якщо записати lst[2] = 4, то модифікований список виглядатиме так – [1, 2, 4, ’b’].

Подамо перелік основних методів та функцій для роботи із списками.

Lst.append(el) – додає елемент el у кінець списку.

Lst.extend(ListN) – розширює список, додаючи у кінець всі елементи зі списку ListN.

Lst.insert(i, x) – розміщує елемент x на місці елемента із індексом i, а попередній i-й елемент зміщується на позицію i + 1.

Lst.remove(el) – видаляє перший елемент у списку, що має значення el, а випадку відсутності такого елемента повертає виключення ValueError.

Lst.pop(i) – видаляє елемент із індексом i та повертає його значення. Якщо індекс не вказати, то буде видалений останній елемент списка.

Lst.index(el, start, end) – повертає індекс першого елементу із значенням el. Зауважимо, що пошук ведеться від початкового індексу start до кінцевого end. За замовчуванням start дорівнює 0, а end – довжині списку.

Lst.count(el) – розраховує кількість елементів el у списку Lst.

Lst.sort(key) – сортує список із урахуванням налаштувань ключа key.

Lst.reverse() – записує рядок у зворотному порядку.

Lst.copy() – створює копію списку Lst.

Lst.clear() – видаляє всі елементи, перетворюючи його на порожній.

Кортежі

Кортежі є незмінними списками і при цьому займають менший обсяг пам'яті. Часто їх використовують як фіксовані ключі для словників.

Оголошення кортежів відбувається наступним чином.

```
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
```

Для роботи із таким типом даних як кортеж дозволяється використовувати всі функції та методи списків, які не змінюють власне самий список. Решта інструментів списків не застосовується, оскільки представлена структура є незмінюваною.

Словники

Словники у Python 3 є неупорядкованими наборами різних об'єктів. Щоб звернутися до певного елемента словника необхідно вказати спеціальний ключ, який однозначно визначає доступ до даних у зазначеній структурі.

Словник можна створити декількома способами.

```
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 із # вказаними у
                                   списку # ключами та
                                   # значеннями None.
D = {i: i*i for i in [0, 1, 2, 3]} # Генерує словник # беручи значення # із вказаного
                                   # списку
```

Доступ до об'єкту можливий тільки через ключ.

Приклад 1.32.

```
D = {'key1':25, 'key2':42}
print(D['key1'])
```

Результат виконання програми:

```
25
```

При виконанні зазначеного коду у консоль буде виведено значення 25, яке задається ключем 'key1'.

Серед основних методів та функцій, які використовуються для роботи із словниками варто назвати такі.

Dict.clear() – перетворює будь-який словник у порожній.

Dict.copy() – повертає копію словника.

Dict.fromkeys(seq, value) – створює словник, у якому кожному ключу із списку seq ставиться у відповідність значення із списку value. За замовчування значення послідовності value позначається як None.

Dict.get(key, default) – повертає значення об'єкта, який асоціюється із ключем key. Якщо ключ відсутній у словнику, то виводиться значення, записане у полі default. За замовчуванням default рівне None.

Dict.items() – повертає пари (ключ, значення).

Dict.keys() – повертає значення всіх ключів.

Dict.pop(key, default) – видаляє із словника пару (ключ, значення) за вказаним ключем key. У випадку відсутності вказаного ключа повертається значення, вказане у default. Стандартним значенням змінної default є None.

Dict.popitem() – видаляє та повертає (ключ, значення). Якщо словник порожній, то виводиться повідомлення про виключення KeyError.

Dict.setdefault(key, default) – повертає значення для вказаного ключа, але якщо цього ключа не має у словнику, то створюється нова пара із ключем key та значенням вказаним у default.

Dict.update(Dict1) – оновлює словник, додаючи пари (ключ, значення) із словника Dict1.

Dict.update() – повертає всі значення у словнику.

Множини

Великою особливістю і перевагою Python є підтримка класичних математичних структур та операцій фактично на базовому рівні. Так у цій мові програмування реалізована можливість роботи із такими об'єктами як множини.

Множини Python є певними контейнерами, які містять неповторювані та невпорядковані елементи.

Для того, щоб створити множину необхідно виконати такі дії.

```
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
```

Важливою особливістю множин є те, що зазначена структура містить тільки унікальні елементи, які не повторюються. Досить часто цю особливість використовують для видалення повторюваних значень із списку.

Приклад 1.33.

```
Num = [2, 2, 1, 3, 2, 1, 3]
print(set(Num))
```

Результат виконання програми:

```
{1, 3, 2}
```

Тепер важливо описати методи роботи із множинами.

len(m) – повертає кількість елементів у множині m.

x in m – перевіряє чи елемент x належить множині m.

m.isdisjoin(n) – перевіряє чи множини m та n не мають спільних елементів. Якщо це правда, то повертається значення true.

m == n – перевіряє чи всі елементи множини m є елементами множини n і навпаки.

m.issubset(n) або m <= n – перевіряє чи елементи m є у множині n.

m.issuperset(n) або m >= n – вказує чи множина n є підмножиною множини m.

m.union(n) – об'єднання двох множин.

m.intersection(n) – перетин двох множин.

m.difference(n) – різниця двох множин.

m.copy() – копія множини m.

m.update(n) – додає до множини m елементи множини n.

m.add(x) – додає до множини m елемент x.

m.remove(x) – видаляє елемент x із множини m та повертає виключення KeyError, коли такого елемента не існує.

m.discard(x) – видаляє елемент із множини, якщо він там знаходиться.

m.pop() – видаляє перший елемент вказаної множини.

m.clear() – видаляє всі елементи із множини.

Лабораторна робота № 2.

Тема. Списки, множини, словники, рядки у Python.

Мета. Набути умінь та навичок розробки програм із використанням множин, рядків, словників і списків у Python.

Програмне забезпечення. Програмний комплекс Anaconda для версії Python 3.7.

Контрольні запитання.

1. Рядки та функції роботи із рядками у Python.
2. Поняття списку. Одновимірні та багатовимірні списки. Методи роботи із списками.
3. Множини та кортежі.
4. Словники у Python.

Завдання до лабораторного заняття

Рівень 1

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

2. Список формується із додатних та від'ємних цілих чисел. Обчислити суму кубів 5 найбільших елементів.

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

4. Дано масив із 25 дійсних чисел. Впорядкувати його спаданням та вивести елементи із непарними індексами.

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

Рівень 2

1. Реалізуйте введення даних у один рядок до двовимірного масиви цілих чисел. Забезпечте коректність введення даних у двовимірний список. Перевірте, чи даний двовимірний список цілих чисел має вертикальну вісь симетрії.

2. Реалізуйте введення даних у один рядок до двовимірного масиви цілих чисел. Забезпечте коректність введення даних у двовимірний список. Перевірте, чи заданий двовимірний список цілих має горизонтальну вісь симетрії.

3. Реалізуйте введення даних у один рядок до двовимірного масиви цілих чисел. Забезпечте коректність введення даних у двовимірний список. перевірте, чи заданий двовимірний список [n][n] симетричний відносно головної діагоналі.

4. Реалізуйте введення даних у один рядок до двовимірного масиви цілих чисел. Забезпечте коректність введення даних у двовимірний список. перевірте, чи заданий двовимірний список [n][n] симетричний відносно не головної діагоналі.

5. Реалізуйте введення даних у один рядок до двовимірного масиви цілих чисел. 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. 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.

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.

Файли

Окрім роботи із вбудованими типами об'єктів у Python 3, важливо ознайомитися із читанням інформації з файлів та можливостями їх редагування та запису.

Робота із файлами розпочинається із його відкриттям. Для цього створюються у пам'яті спеціальний об'єкт, до якого записуються дані і посилання на нього.

```
Fl =open('file.txt', 'r')
```

Функція open відкриває доступ до файлу file.txt та дозволяє або ні виконати зчитування та модифікацію інформації розміщеної у ньому, залежно від того , який «флаг» розміщений другим параметром.

Для того щоб прочитати вміст файлу використовуються такий метод як read. Він дозволяє прочитати файл повністю, якщо не вказані додаткові параметри.

Приклад 1.34.

```
Fl.open('file.txt')
print(Fl.read())
print(Fl.read(1))
```

Результат виконання програми:

```
'Hi my world'
'H'
```

Також файл можна прочитати за допомогою циклу for.

Приклад 1.35.

```
Fl.open('file.txt')
for ln in Fl:
   print(ln)
```

Результат виконання програми:

```
'Hi my world\n'
'HI!!!\n'
```

Всі значення другого параметру представлені у таблиці 1.1.

Таблиця 1.1.

Значення другого аргументу функції open	Опис
’r’	Відкриття файлу із можливістю читання. Це значення встановлено за замовчуванням, якщо другий аргумент не вказано.
’w’	Відкриття документу на запис, але вміст файлу видаляється, якщо файлу не існує то він створюється.
’x’	Відкриття файлу вайлу для запису. Якщо файлу не існує то повертається виключення.
’a’	Дозволяє додавати інформацію у кінець файлу.
’b’	Відкриття файлу в двійковій системі.
’t’	Відкриває файл у текстовому вигляді.
’+’	Відкриває файл із можливістю читання та дозапису.

Запис до файлу виконується за допомогою команди write.

Приклад 1.36.

```
Lst = [‘hi’, ‘hello’, ‘world’]
Fl.open(‘file.txt’, ‘w’)
for i in lst:
   Fl.write(i + ‘\n’)
Fl.close()
```

І на останньому кроці роботи є обов’язковим закриття робочого файлу. Для цього використовується метод close().

Функції

Python – це мова програмування, яка дозволяє розробку програм із застосовуванням різних парадигм програмування, зокрема і за допомогою функціонального підходу. Для нього характерним являється те, що будь-яка функціє є об’єктом, який примати та повертати значення.

Для опису користувацької функції використовується наступна конструкція.

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

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

У функцію можна передавати аргументи. Вони можуть бути представлені як фіксованою кількістю значень так і необмежені кількістю. Нижче наведено приклад опису функцій у Python 3.

Приклад 1.37.

```
def modify_list(l):
    i = len(l) - 1
    while i >= 0:
        if l[i] % 2 == 1:
            l.remove(l[i])
        i -= 1
    for i in range(len(l)):
        l[i] = l[i] // 2
##############
lst = [1, 3, 5, 7]
modify_list(lst)
print(lst)
```

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 1.38.

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

Результат виконання програми:

5

Модулі

Для Python 3 характерна велика кількість додаткових модулів, які можна підключити за допомогою інструкції import назва модуля. Кожен модуль має певний набір функцій, які дозволяють спростити написання тих чи інших частин коду при розробці програми. Вони мають різне призначення, починаючи від Web розробки та завершуючи робототехнікою, аналізом даних та штучним інтелектом. До них варто віднести такі: audioop, base64, calendar, cgi, configparser, csv, curses, datetime, decimal, difflib, email, gettext, gzip, zlib, hashlib, html, http, io, itertools, json, logging, match, os, pathlib, random, re, socket, sqlite, ssl, string, threading, time, tkinter, urllib і urllib2, xml тощо.

NumPy та SciPy

Розглядувана мова програмування має велику кількість математичних функцій, але виникають ситуації, коли їх все ж буває не достатньо для спрощення процесу програмування. Наприклад необхідно опрацювати великі масиви числової інформації, зокрема при реалізації наукових розрахунків. Оскільки розглядувана нами мова програмування є інтерпретованою, то це не завжди можна швидко реалізувати за допомогою стандартизованих функцій. У таких випадках використовують спеціалізовані модулі, які написані за допомогою мов програмування C та C++, що дозволяють вирішити поставлені завдання максимально оперативно.

До таких пакетів відносять NumPy. Його особливістю є підтримка обробки багатовимірних матриць за допомогою добре налагодженого

функціоналу. Представлена бібліотека досить широко використовується при математичних обчислення, аналізі та обробці даних, при створенні систем штучного інтелекту тощо.

Коротко опишемо найбільш використовувані методи та функції представленої бібліотеки. Для зручності написання коду всі числові масиви позначатимемо наступним чином – arr_a, arr_b, arr_c і так далі.

arr_a.ndim() – кількість вимірів у масиві.

arr_a.shape() – визначає розмірність масису arr_a.

arr_a.size() – загальна кількість елементів у масиві.

arr_a.dtype() – вказує на тип елементів масиву.

arr_a.itemsize() – повертає розмір у байтах кожного елементу масиву.

Створення масиву за допомогою інструментарію модуля NumPy виглядає наступним чином.

```
import numpy as np
arr_a = np.array([1, 2, 6])
```

У цьому випадку функція array перетворює список у масив NumPy. Якщо у цій функції задати аргумент dtype, то таким чином можливо задати тип даних для всього масиву явно.

```
arr_a = np.array([1, 2, 6], dtype = ‘int’)
```

Дозволяється створювати масиви із нулів (функція arr_a = np.zeroes((2, 3))) та одиниць (arr_a = np.ones((3, 2))). Одиничну квадратну матрицю визначають за допомогою команди eye(розмірність). Порожній масив задається функцією empty(розмірність), але при цьому необхідно враховувати, що у такому випадку всі елементи масиву будуть містити значення, які розміщуються у пам’яті на момент створення масиву.

Генерація масиву послідовності із заданим кроком виконується за допомогою функції arange (наприклад np.arange(10, 20, 2)) або linspace(np.linspace(1, 2, 8)).

Тепер варто розглянути операції із числовими масивами, створеними за допомогою модуля NumPy. Завчасно необхідно зауважити про тнаступне.

1. Розмірність масивів, над якими виконуються дії, мають бути однаковими.

2. Для двовимірних масивів їх множення є по елементним і не відповідає операції множення двох матриць.

Отже, враховуючи обмеження, маємо, що для масивів характерним є дії: додавання, віднімання, множення, ділення, піднесення до степеня. Такі дії стосуватимуться кожного елементу структури і вони записуються аналогічно до дій із числами.

Для того щоб виконати множення матриць необхідно скористатися операцією @ або ж методом dot (наприклад A.dot(B)).

До масивів даних, створених за допомогою модуля NumPy можуть бути використані і універсальні функції, зокрема такі як: sum(), mean(), median(), max(), min(), sort() тощо.

Для машинного навчання важлива правильна підготовка даних. Зокрема на певних етапах необхідно порівняти двох масивів чисел. Для цього використовують звичні знаки порівняння. Так у випадку порівняння двох структур однакової розмірності та кількості члені, буде повернуто масив значень True/False.

Приклад 1.39.

```
a = np.array([1, 3, 7])
b = np.array([2, 1, 2])
print(a > b)
```

Результат виконання програми:

```
array(False, True, True, dtype = bool)
```

Для опрацювання таких масивів можуть використовуватися функції any() та all(), які дозволяють перевірити наявність істинних елементів і чи всі елементи істинні відповідно.

Є також можливість визначення помилок при введенні даних у масив. Наприклад функція isnan() повертає масив бульових елементів, які покажуть у якій комірці розміщена помилка або відсутнє певне значення.

NumPy дозволяє модифікувати форму представлення масивів.

arr_a.ravel() – перетворює будь-який масив у одновимірний.

arr_a.shape() та arr_a.reshape() – дозволяє змінити форму представлення масиву, наприклад із одновимірного на двовимірний.

arr_a.transponse() – транспонування масиву.

Окрім перерахованих функцій модуля NumPy у зазначений пакет входять вбудовані методи для роботи із індексами елементів масиву, створення зрізу, ітерації, виконання об'єднання двох масивів тощо. Більш докладно можна ознайомитися із повним списком методів на офіційному сайті бібліотеки.

Ще однією перевагою розглядуваної бібліотеки є те, що реалізовано глибоку взаємодія із модулем SciPy та його функціоналом. Це потужний математичний комплекс, який напряму може працювати із масивами NumPy та виконувати складні математичні розрахунки: обчислення матриць та лінійна алгебра, методи оптимізації, кластеризація, математична статистика, опрацювання сигналів тощо.

Для опису цього та всіх його можливостей не достатньо одного розділу цього навчально-методичного посібника, а тому методи, які використовуватимемо у подальших розрахунках будемо вказувати. Лише зауважимо, підключення його у програму відбувається аналогічно до NumPy (import scipy as sp).

Обидва ці модулі входять до дистрибутиву Anaconda.

Лабораторна робота № 3.

Тема. Функції, зовнішні модулі та файли у Python.

Мета. Набути умінь та навичок розробки програм із використанням функцій, зовнішніх модулів та файлів у Python.

Програмне забезпечення. Програмний комплекс Anaconda для версії Python 3.7.

Контрольні запитання.

1. Опис функцій у Python. Рекурсія.
2. Поняття зовнішніх модулів та їх методів.
3. Файли та метотоди роботи із ними.

Завдання до лабораторного заняття

Рівень 1

1. Реалізувати підрахунок кількості унікальних символів у рядку за допомогою використання функції та словника.

2. Запишіть функцію, яка обчислює суму довільної кількості числових аргументів. При цьому обов'язковою є перевірка на коректність числових даних.

3. Реалізуйте функцію, яка дозволяє будувати таблицю (словник) значень для будь-якого числа значень і довільної функції.

4. Написати функцію, яка виконує циклічне зміщення елементів списку цілих чисел на вказану кількість кроків n раз. Результат виводиться після кожного перетворення. Зсув має бути кільцевим, тобто якщо елемент вийшов за межі списку він має з'явитися із іншого кінця.

5. Обчислити значення виразу $1+x+x^2+....+x^n$ та результат подати у вигляді $1+2+2^2+....+2^n$ = сума.

Рівень 2

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, яка завершується 0. Виведіть цю послідовність у зворотному порядку.

5. Реалізувати піднесення до цілого степеня, незалежно від того від'ємний чи додатній степінь

Рівень 3

1. Порахувати, скільки слів у текстовому файлі рядків, слів і символів.

2. Задано файл із розкладом на тиждень. Окрім дня тижня, у рядку подаються назви предметів. Біля назви предмету у дужках тип заняття (лекція, лабораторна, практична). У рядку може бути запис або дня, або предмету. Порахувати кількість різних предметів, кількість лекцій, практичних, лабораторних.

3. Написати програму, яка створює текстовий файл із 9 рядків, у якому в 1-рядку записано одна літера «а», 2 – дві ….

4. Задано текстовий файл, який містить принаймні один рядок. Порахувати кількість рядків, які завершуються заданим символом.

5. Задано текстовий файл, який містить принаймні один рядок. Вивести рядки, які завершують на вказаний символ та інвертувати їх.

ЧАСТИНА 2.
ВСТУП ДО МАШИННОГО НАВЧАННЯ

Поняття машинного навчання

В умовах бурхливого розвитку сучасних інформаційних технологій значного поширення отримали системи моделювання штучного інтелекту. Особливо затребуваними стали програмні засоби, які можуть навчатися, удосконалюватися, здобувати досвід у процесі вирішення певного класу задач. На даний час цей напрям відомий під терміном «машинне навчання».

Перш ніж розкрити значення вказаного поняття, звернемося до більш загальної дефініції – «штучний інтелект». Зауважимо, що, незважаючи на надзвичайно велику зацікавленість цією тематикою як прикладною наукою та бізнесом так і філософією, єдиного визначення цього поняття на даний час не існує. 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*[4]. У контексті цього трактування будемо розглядати і термін «машинне навчання».

Загалом ідея пов’язана із створенням машин, які вміють здобувати знання самостійно належить Алану Тюрінгу. Вперше він висловив таку думку в статті «Computing Machinery and Intelligence»[5] 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[6].

---

[4] Нікольський Ю. В. Системи штучного інтелекту: навчальний посібник / Ю. В. Нікольський, В. В. Пасічник, Ю. М. Щербина. – Львів: «Магнолія-2006», 2015 р. – 279 с. – С. 11.

[5] Turing A. Computing Machinery and Intelligence / A. Turing // Mind. – 1950. –Vol. 59. – P. 433 – 460.

[6] Samuel A. L. Some Studies in Machine Learning Using the Game of Checkers / A. L. Samuel // IBM Journal. – July 1959. – P. 210–229.

На даний час одним із найбільш точним означенням процесу здобуття знань обчислювальною системою представлено Томом 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*[7].

Звичайно, такий напрям як машинне навчання пов'язаний із багатьма іншими розділами науки, які займаються опрацювання великих масивів інформації. Зокрема наведемо у приклад таку важливу і прикладну частину сучасних дослідження як аналіз даних (англійською мовою Data mining). Вона об'єднує у собі сукупність методів опрацювання даних та виявлення серед них завчасно невідомої, нетипової, корисної інформації та її інтерпретація, з метою прийняття певних рішень та узагальнень у різних сферах людської діяльності. Фактично і машинне навчання, і аналіз даних використовують схожий інструментарій для отримання знань, проте їх мета використання відрізняється. Так у першому випадку «знання» необхідні для «навчання» системи (побудова ефективної моделі штучного інтелекту), тоді як для другого – важливим є сприйнятливість даних, можливість їх візуалізації, для того щоб людина могла сприйняти і зробити власні висновки.

Важливе значення для розуміння фахівцями принципів побудови систем машинного навчання є наявність необхідних знань, умінь та навичок із програмування та аналізу ефективності алгоритмів, статистики, комбінаторики, математичної логіки, теорії графів тощо.

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.

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

2. Навчання без керівника (unsupervised learning) – передбачає 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. Фактично така модель отримання досвіду машиною є підвидом навчання із керівником.

Серед охарактеризованих підходів детально розглянемо саме перший випадок – *навчання із керівником*, як найбільш поширений. Визначимо цей термін більш детально.

Нехай задано множину X – сукупність всіх параметрів об'єкта або суб'єкта та множина Y, яка є певним наслідком, зробленим на основі вибірки X. Таким чином навчання із керівником передбачає знаходження такої функції f: X → Y, яка дозволить за даними $x_i \in X$ *передбачити відповіді* $y_i \in Y$. *У цьому випадку функцію f називають моделлю або розв'язуючою функцією*[8].

Коротко наведемо приклади застосування систем машинного навчання для вирішення реальних задач. На даний час зазначена модель штучного інтелекту застосовується у медичній діагностиці, фільтрації спаму, рекомендаційних системах для реклами товарів та послуг, у програмах комп'ютерного зору, розпізнавані текстів, комп'ютерній лінгвістиці, машинному перекладі, розпізнаванні мови тощо.

Особливості підготовки даних для створення моделей машинного навчання

Незалежно від того як буде використовуватися масив даних (для побудови моделі машинного навчання, перевірки її результативності

[8] Нікольський Ю. В. Системи штучного інтелекту: навчальний посібник / Ю. В. Нікольський, В. В. Пасічник, Ю. М. Щербина. – Львів: «Магнолія-2006», 2015 р. – 279 с. – С. 119.

або ж для прогнозування) їх необхідно подати максимально адаптованими для роботи із системою.

Класичним і, напевно, найбільш зручним представленням інформації є форма таблиці, в рядках якої записані певні об’єкти, що аналізуються, а у стовпчиках подаються значення ознак, яка характеризує стан певного визначеного об’єкту.

Перш ніж продовжити розгляд теми, пов’язаної із підготовкою даних для систем машинного навчання варто окреслити найбільш характерні види ознак, за допомогою яких описуються об’єкти.

1. Кількісні або числові ознаки. Описують числові значення для опису певних явищ. Зокрема до них варто віднести визначення часу, довжин, маси об’єктів, вартості, товарів тощо.

2. Порядкові характеристики ознак дозволяють характеризувати їх за принципом «більше – менше». Всі об’єкти ранжуються у певній черговості від найменшого номера до найбільшого або ж навпаки.

3. Номінативна ознака об’єкта дозволяє лише відрізняти один об’єкт від іншого. Простим прикладом можна слугувати позначення числами певних товарів (картопля – 1, морква – 2, капуста – 3, …), які позначають на вагах самообслуговування у супермаркетах. Це не означає, що можемо купити одну картоплину, дві морквини і три капустини. Кожне із чисел позначає клас об’єктів і жодним чином не вказує на їх порядкові чи кількісні характеристики.

Поняття нормалізації даних та їх відновлення

Сучасні бази даних можуть містити ознаки, які описуються різними числовими значеннями. При цьому в рамках однотипних характеристик (наприклад числових) може міститися інформація у різних одиницях вимірювання, відрізнятися порядком, окрім того у самих даних можуть бути помилки, пропуски або ж інші некоректні значення, які заважатимуть якісному опрацюванні інформації. Отже маємо важливу проблему підготовки даних для створення моделей машинного навчання.

Загалом вирішити вказану проблемну ситуацію можна наступним чином.

1. Відновлення пропущених даних за допомогою основних статистичних величин.

2. Виконання нормалізації даних.

Тут 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. Нижче буде наведено методи, які дозволятимуть покращити якість даних для програми.

Спершу розглянемо процес відновлення пропущених даних. Як би парадоксальним це не було, але найпростішим способом, що дозволяє позбутися помилкових записів у таблиці є звичне видалення рядків (об'єктів), які містять помилки. Такий підхід дозволяється застосовувати до надзвичайно великих вибірок, коли видалення невеликого числа об'єктів не призведе до руйнування загальної картини даних і не впливатиме на якість формування функції рішення.

Але не завжди такий радикальний спосіб вирішення проблеми помилок у базі даних є ефективним. Наприклад таблиця має незначну кількість записів і відсоток помилок такий, що вилучення об'єктів призведе до спотворення інтерпретації даних, а отже формування некоректно працюючої моделі машинного навчання. Для таких випадків варто виконати заміну втрачених або ж помилкових даних на такі значення як: середнє значення властивості для певної вибірки, моду або ж медіану. Для розрахунку цих значень у Python передбачені такі функції як mean (list) у модулі NumPy, mode(list) та median(list) у пакеті statistics. Також у останній згаданій бібліотеці функцій передбачено ще два методи для розрахунку медіани: median_low(list) та median_high(list). Їх використовують для розрахунку медіани, якщо у списку міститься парна кількість об'єктів. Наприклад вибірка має 10 об'єктів. У позиції 5 розміщено значення 7, а елемент із індексом 6 має значення 8. Медіана цього списку буде рівна $\frac{7+8}{2}=7{,}5$. А вказані функції повернуть такі значення як 7 та 8 відповідно.

Ще одним методом відновлення даних є використання метрики[9].

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 $\rho(X,Y) = \sum_{i=1}^{n}|x_i - y_i|$.

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

Розглянемо алгоритм відновлення даних за допомогою метрики.

1. Визначаємо об'єкт, який має у одній із властивостей пошкоджену інформацію.

2. Виключаємо визначену властивість із розгляду.

3. Використовуючи значення інших властивостей встановлюємо об'єкта найбільш схожого до того, у якому відновлюємо дані. Найближчим буде той об'єкт, який матиме за певною визначеною метрикою найменше її значення.

4. Заміна відсутньої або пошкодженої інформації значенням найбільш близького об'єкту або використати формулу для обрахунку відповідного значення.

---

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

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

Ще однією важливою частиною підготовки даних є процедура нормалізації. Річ у тім, що кожна властивість об'єкт може відрізнятися одиницями вимірювання та порядком представленої числової інформації. Така різноманітність значно ускладнює як можливості для навчання моделі так і для інтерпретації результатів її застосування.

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.

Ще одним важливим напрямом використання метрики є створення рекомендаційних систем. Рекомендаційні системи використовуються для реклами певної продукції, яка потенційно може зацікавити покупця.

Кожен товар характеризується певним числовими властивостями (ціною, вагою і іншими числовими характеристиками). За допомогою метрики обраховують «відстань» між об'єктами та ранжують їх. Товари, які найбільш подібні до вибраного товару за визначеними

характеристиками, будуть мати менше значення «відстані» ніж інші та виводитимуться на екран.

Звичайно, наведений короткий опис алгоритму пошуку подібних товарів є певним спрощенням побудови рекомендаційної платформи, оскільки сучасні системи враховують і ключові слова, і час перегляду, і кількість повернень читача на сторінку товару, і багато інших важливих параметрів, але, наведений приклад наочно демонструє створення моделі машинного навчання на основі такого поняття як «метрика».

Лабораторна робота № 4.

Тема. Автоматизація підготовки даних для алгоритмів машинного навчання.

Мета. Набути умінь та навичок розробки програм для автоматизації підготовки даних для створення моделей машинного навчання за допомогою мови програмування Python.

Програмне забезпечення. Програмний комплекс Anaconda для версії Python 3.7.

Контрольні запитання.

1. Табличне представлення інформації. Зчитування текстового файлу у Python.
2. Ключові характеристики ознак (максимальні та мінімальні значення, середнє арифметичне, мода, медіана, середнє квадратичне відхилення).
3. Основні способи відновлення даних.
4. Нормалізація даних.

Завдання до лабораторного заняття

1. Відкрити текстовий файл та проаналізувати дані.
2. Замінити всі помилки на середнє арифметичне значення властивості та записати у файл TabSA.txt.
3. Замінити всі помилки на моду та записати у файл TabMODA.txt.
4. Замінити всі помилки на медіану та записати у файл TabMD.txt.

5. Виконати нормалізацію даних та записати у файл TabNorm.txt
6. Відновити дані за допомогою метрики Манхеттен.

Лабораторна робота № 5.

Тема. Побудова рекомендаційної системи відеопереглядів.

Мета. Набути умінь та навичок розробки рекомендаційних систем для порталу відопереглядів за допомогою метрики та використання мови програмування Python.

Програмне забезпечення. Програмний комплекс Anaconda для версії Python 3.7.

Контрольні запитання.

1. Поняття рекомендаційної системи.
2. Алгоритми нормалізації даних.
3. Види метрик (Евклідова, метрика Манхеттен, Max-метрика).

Завдання до лабораторного заняття

1. Відкрити текстову таблицю даних.
2. Перевірити дані на похибки і якщо вони там є та виконати виправлення за допомогою метрики.
3. Нормалізувати дані таблиці.
4. Створити рекомендаційну систему для відеохостингу, яка пропонує до перегляду 5 близьких за параметрами відеороликів. Система працює наступним чином — користувач вводить такі параметри як час перегляду, кількість позитивних та негативних оцінок, а система повертає список відео із вказаними статистичними даними.
5. Удосконалити програму із урахуванням використання ключових слів для опису відео.

Системи прогнозування певного числового значення. Регресія.

Найбільш широким спектром задач, якими займаються фахівці із машинного навчання, є системи для прогнозування. Загалом їх розділяють на два великих класи.

1. Передбачення числового значення певної ознаки Y (власне регресія).

2. Передбачення номінального показника.

Розглянемо стратегії формування моделі регресії для машинного навчання.

I. Спочатку необхідно проаналізувати дані вибірки, відновити пропущені та помилкові, позбавитися викидів.

II. Множину об'єктів необхідно розбити на дві важливі частини. 1 частина – тренувальна вибірка, 2 частина – тестова. На основі тренувальної вибірки необхідно «навчити» систему прогнозувати, а на основі тестової – перевірити ефективність моделі. На цьому етапі відбуватиметься формування розв'язуючої функції, яка визначатиметься певним переліком коефіцієнтів ω_0, ω_1, ω_2, …, ω_n.

$$y' = \omega_0 + \omega_1 x_1 + \omega_2 x_2 + \ldots + \omega_n x_n$$ [10]

Щоб розрахувати зазначені коефіцієнти, необхідно знайти мінімум для функції $L(\omega_0, \omega_1, \omega_2, \ldots, \omega_n) = \sum_{i=1}^{n} |y'_i - y_i|$, де y'_i – прогнозоване значення, а y_i – реальне значення у тренувальній вибірці.

III. Оцінка якості моделі. Для цього використовуються такі показники.

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - y_i^*|$$

n – об'єм вибірки.

y_i – значення тестової вибірки.

y^*_i – значення тестової вибірки.

$$MAPE = \frac{1}{n} \sum_{i=1}^{n} \frac{|y_i - y_i^*|}{|y'_i|} \cdot 100\%$$

При цьому варто обов'язково зауважити, що прогнозоване значення для тестової вибірки не повинно давати такі ж самі дані, які розміщуються у відповідних значеннях навчальної вибірки.

Серед недоліків такої системи машинного навчання варто назвати такі: чутливість до викидів, наявності великих чисел у описі

[10] Приклад наведений для випадку лінійної регресії.

ознак та високої кореляційної залежності між характеристиками, яка заважає побудові ефективної моделі.

Приклад створення регресивної моделі за допомогою мови Python представлено далі.

Приклад 2.1.

Побудуємо модель регресії для прогнозування середньої вартості будинку на основі визначених параметрів. Для цього використовуватимемо базу даних вартості житла у місті Бостон, яка входить до бібліотеки машинного навчання sklearn та знаходиться у відкритому доступі.

Для обрахунку ваг використовуватимемо методи цієї ж бібліотеки.

```
import numpy as np #Підключаємо модуль NumPy

from sklearn import datasets
boston = datasets.load_boston()#Завантажуємо базу даних
boston.keys()#Демонструємо ключові поля у консолі

boston_df = pd.DataFrame(boston.data, columns = boston.feature_names)
boston_df.head()
"""Підключаємо бібліотеку машинного навчання"""
from sklearn.linear_model import LinearRegression
linear_regression = LinearRegression()
"""Створюємо модель машинного навчання"""
model = linear_regression.fit(boston.data, boston.target)
"""Представляємо розраховані ваги"""
feature_weight_df = pd.DataFrame(list(
                zip(boston.feature_names,model.coef_)
))
feature_weight_df.columns = ['Feature', 'Weight']
print(feature_weight_df)
"""Розраховуємо прогнозований результат"""
arr_np_w = np.array(model.coef_)
arr_np_d = np.array(boston.data[15])
first_pradict = sum(arr_np_w*arr_np_d)+ model.intercept_
print(first_pradict)
```

Задачі кластеризації

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

Хоча кластеризацію це модель машинного навчання без керівника (unsupervised learning), проте її часто використовують для: спрощення подальшої обробки даних (у певній зменшеній групі подібних об'єктів набагато легше виявити залежності); зменшення об'єму даних (заміна групи ознак однією, еталонною характеристикою і тим самим зменшить розмірність); пошук викидів та аномалій.

На даний час найбільш популярним алгоритмами, які дозволяють виконати розбиття на кластери є:

1. Розбиття на задане число кластерів (алгоритм k-means).

2. Система самостійно визначає необхідну кількість кластерів (алгоритм FOREL).

Алгоритм k-means

Робота заданого алгоритму передбачає попереднє введення людиною кількості кластерів, на які необхідно розбити задану множину X певного простору R^m, де m – розмірність простору. Основною ідеєю є одночасний пошук всіх центрів кластерів.

Отже.

1 Крок. Задаємо кількість кластерів, на які потрібно розбити задану множину X.

2 Крок. Генеруємо k довільних точок простору R^m – первинних центрів.

3 Крок. Переглядаємо всі об'єкти і розраховуємо «відстані» за певною метрикою до кожного із центрів.

4 Крок. Порівнюємо «відстані». Чим ближче об'єкт розміщується до центра тим із більшою ймовірністю його можна віднести до певного кластера, який сформований певним центром.

5 Крок. Перерахунок центра мас для кожного із класу і повторення кроків 4 і 5.

Цикл повторюється до тих пір, поки *виділені групи* не стабілізується.

Головним недоліком такої системи є те, що ефективність алгоритму залежить від досвіду та кваліфікації фахівця, який *налаштовує роботу моделі*.

Приклади роботи алгоритму на мові Python та використання бібліотеки sklearn подано нижче.

Приклад 2.2.

У представленому прикладі автоматичного генеруються розподіли точок на площині для вказаних параметрів. Зокрема визначальним параметром є кількість точок групування (в даному випадку їх 5). Після чого, за допомогою методові вказаної бібліотеки sklearn, розраховуємо ці групу точок (кластерів) та вказуємо їх центр.

```
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()

from sklearn.datasets.samples_generator import make_blobs

X, y_true = make_blobs(n_samples = 200, centers = 5,
                       cluster_std=0.75, random_state = 0)
plt.scatter(X[:, 0],X[:, 1], s = 60)

from sklearn.cluster import KMeans

km = KMeans(n_clusters = 5)
km.fit(X)

y_km = km.predict(X)
plt.scatter(X[:, 0],X[:, 1], c = y_km, s = 50, cmap = 'viridis')

centers = km.cluster_centers_

plt.scatter(centers[:, 0], centers[:, 1], c = 'red', s = 200, alpha = 0.5)
```

Результат роботи алгоритму можна представити графічно (рис. 2.1.).

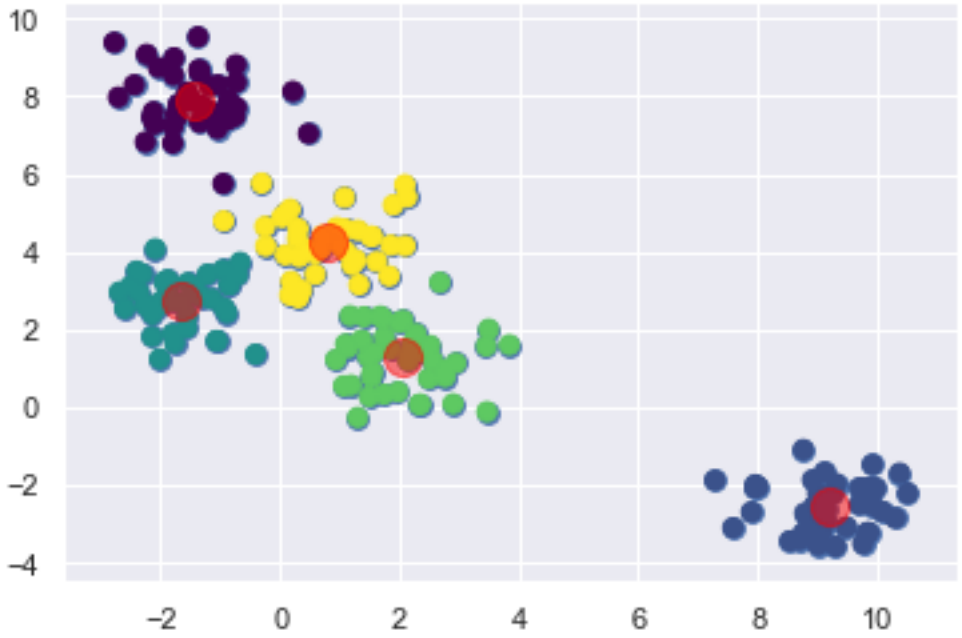

Рис. 2.1. Результат виконання кластеризації за допомогою методу k-means. Червоними точками позначено центри кластерів.

Алгоритм FOREL

«Вузьким місцем» схематично представленого попереднього алгоритму є те, що людина не може визначити однозначно кількість потрібних кластерів. Алгоритм FOREL має виділити точки скупчення і оголосити їх центрами кластерів.

На першому кроці розміщуємо довільний центр *F*. Другим – проводимо умовну сферу радіуса *R* та вибираємо всі об’єкти, які містяться у ній. Третім кроком розраховуємо центр тяжіння об’єктів (точка, координати якої є середніми значеннями виділених об’єктів) і переміщуємо туди точку *F*. Кроки 2 і 3 повторюються до тих пір поки множина вибраних точок не стабілізується, тобто не будуть додаватися нові об’єкти. У цьому випадку визначена множина оголошується кластером і видаляється із вибірки. А алгоритм розпочинається знову із кроку один.

Розрахунок відбувається до тих пір, поки не буде виділено всі кластери.

До недоліків алгоритму можна віднести те, що при його виконанні утворені кластери міститимуть невелику кількість об’єктів,

а самих груп буде дуже багато, що не відповідатиме реальному стану речей та спричинить труднощі для інтерпретації.

Лабораторна робота № 6.

Тема. Задачі кластеризації.

Мета. Набути умінь та навичок розробки програм для виконання розбиття бази даних на кластери за допомогою метода *k-means* за допомогою мови програмування Python.

Програмне забезпечення. Програмний комплекс Anaconda для версії Python 3.7.

Контрольні запитання.

1. Поняття кластеризації
2. Основні задачі кластеризації.
3. Алгоритм *k-means*.
4. Алгоритм *FOREL*.

Завдання до лабораторного заняття

1. Завантажити базу даних Iris, вбудовану в модуль *scikit-learn*.
2. Виконати нормалізацію даних.
3. Підключити бібліотеку *scikit-learn*
4. Виконати кластеризацію заданої множини (зокрема виділити центри кластерів).
5. Продемонструвати графічно результати роботи алгоритму.

6*. На прикладі бази даних Iris спробуйте реалізувати відшукання центрів кластерів за допомогою алгоритму *FOREL*, користуючись стандартними можливостями мови програмування Python.

Задачі класифікації

Поряд із задачами кластеризації, у машинному навчанні, важливе місце посідають моделі, які займаються класифікацією (розподілом по групах у відповідності до наперед вказаних ознак).

Тактику вирішення таких задач умовно можна розділити на дві частини. На першому етапі варто розділити вибірку на дві частини: тестову та тренувальну бази даних. Тренувальна вибірка дозволить навчити модель, а на основі тестової вибірки перевірити ефективність

системи. Звичайно такий підхід є традиційним для систем машинного навчання, проте дозволяє відслідковувати правильні і помилкові рішення моделі. Для цього використовують матрицю помилок або англійською мовою *confusion matrix*.

Матрицею помилок $M = \{m_{ij}\}_{i,j=1}^{C}$ *показує кількість об'єктів, які реально належить класу* ω_i, *проте помилково віднесені до класу* ω_j.

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

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 $TPR = \frac{TP}{TP + FN}$.

За їх допомогою будується так звана ROC крива.

ROC крива показує залежність ймовірності виявлення об’єкта класа 1 (TPR) від ймовірності помилкової класифікації (FPR) для різних об’єктів, а площа, яка обмежується цією кривою вказує на ефективність моделі класифікації.

Розглянемо один із найпоширеніших методів виконання класифікації або *kNN* – метод.

Нехай існує певний об’єкт $A(a_1, a_2, \ldots, a_n)$ із певними характеристиками $\{a\}_{i=1}^{n}$. Алгоритм, який дозволяє віднести його до певного класу будується на основі метрики і полягає у знаходженні *k* найбільш близьких сусідів, де *k* є вхідним параметром для моделі.

Вважається, що об’єкт $A(a_1, a_2, \ldots, a_n)$ відноситься до певного класу, якщо серед *k* його найближчих сусідів розміщено найбільше об’єктів певного класу.

На ефективність представленого алгоритму впливають такі параметри: вибір метрики, наявність викидів та велика кількість ознак.

Щоб мінімізувати негативні фактори необхідно підібрати вибрати необхідне значення *k*. Для цього використовують метод вибору оптимальних параметрів (крос-валідація). Суть методу полягає у наступному.

1. Розбиваємо вибірку на *k* рівних частин.

2. Виконуємо навчання моделі на $k - 1$ частинах, а перевіряємо її ефективність на k-тій частині. У результаті отримуємо $k - 1$ оцінок ефективності моделі.

3. Знаходимо середнє значення оцінок, та оцінюємо загальне значення моделі.

Нижче поданий алгоритм побудови моделі класифікації на мові Python.

Приклад 2.3.

Для реалізації методу класифікації kNN будемо використовувати ту ж саму базу даних квіток ірисів, яка міститься у бібліотеці scikit-learn. Окрім цього виведемо у консоль важливі значення ефективності запропонованого алгоритму.

```
import sklearn as sk

iris = sk.datasets.load_iris()
X_iris, Y_iris = iris.data, iris.target

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)

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

from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)
```

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Дерева рішень

Ще одним важливим напрямом розвитку систем машинного навчання є побудова дерев рішень. Часто їх використовують для розв'язування задач прогнозування. Загалом ця модель є ще однією реалізацією машинного навчання із керівником і за своєю суттю схожа до того як такі задачі вирішує людина.

У загальному випадку дерево рішень складається із кореня (найвищої вершини), вузлів (правила розв'язування або прийняття рішень) та листків (містить значення прогнозів). Графічний приклад такого дерева рішення наведено на рисунку.

Правилом розв'язування є деяка функція об'єкта, яка дозволяє визначити у яку із вершин необхідно помістити об'єкт. У листових вершинах можуть розміщуватися як значення певної цільової функції (задачі регресії) так і клас, до якого необхідно віднести об'єкт.

Для побудови необхідного дерева необхідно визначити оптимальну ознаку об'єкту, яка дозволить максимально ефективно створити модель.

Важливим також є визначення граничних значень ознаки, що дозволять максимально ефективно створити модель. Серед основних методів знаходження оптимальної статистичної ознаки використовують ентропію, невизначеність Gini та статистичну інформативність. Серед них більш докладно розглянемо невизначеність Gini, як один із самих популярних методів.

Формула, яка описує невизначеність Gini має наступний вигляд.

$$Gini(F') = Pr(F' = False)*Pr(Y = False \mid F' = False)*Pr(Y = True \mid F' = False) + Pr(F' = True)*Pr(Y = False \mid F' = True)*Pr(Y = True \mid F' = True)$$

Де Pr – це ймовірність тієї чи іншої події.

Ця формула дозволяє визначити розкид значень ознаки Y при фіксованому значенні ознаки F′. Для побудови розгалуження необхідно вибирати ознаки із мінімальним значенням Gini.

Приклад побудови класифікатора за допомогою бібліотеки *scikit-learn* наведено нижче.

Приклад 2.4.

Демонстрація ефективності класифікатора побудованого на дереві рішень на основі бази даних ірисів.

```
from sklearn.datasets import load_iris

iris = load_iris()

#print(iris)
X = iris['data']
Y = iris['target']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.20)
print(X_train)

from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Лінійний класифікатор

Якщо дерева рішень можуть вирішувати як задачі класифікації так і прогнозування, то лінійний класифікатор вирішує задачу класифікації. Основою алгоритму є побудова лінійної розділяючої поверхні (якщо маємо два класа) або кусково-лінійної (у випадку трьох і більше класів).

Нехай задано об'єкт $X(x_1, x_2, \ldots, x_n)$, де $x_1, x_2, \ldots, x_n$ – значення його властивостей. Об'єкт A належить до класу 1, якщо $\omega_0 + \omega_1 x_1 + \omega_2 x_2 + \ldots + \omega_n x_n > 0$ та відноситься до класу -1, коли $\omega_0 + \omega_1 x_1 + \omega_2 x_2 + \ldots + \omega_n x_n < 0$. $\omega_0, \omega_1, \omega_2, \ldots, \omega_n$ – значення ваг, які розраховуються на тренувальній вибірці.

При застосуванні лінійного класифікатора можуть виникнути такі помилки.

1. $\sum_{i=1}^{n} \omega_i x_i + \omega_0 > 0$, але розглядуваний об'єкт відноситься до класу – 1.

2. $\sum_{i=1}^{n} \omega_i x_i + \omega_0 < 0$, але розглядуваний об'єкт відноситься до класу 1.

Щоб уникнути таких ситуацій розраховують величину відступу M (від англійського слова margin).

Модель помиляється при аналізі об'єктів тренувальної вибірки , якщо M < 0, де y – значення певної цільової функції.

Отже умови, які дозволяють зменшити кількість помилок машинного навчання, є такі.

$$[M < 0] = \begin{cases} 1, \textit{ якщо } M > 1 \\ 0, \textit{ якщо } M < 0 \end{cases}$$

Таким чином необхідно мінімізувати функцію $\sum_{i=0}^{n} [M_i < 0]$, де M_i – значення відступу для певного об'єкта у тренувальній вибірці.

Наступним кроком є відшукання мінімум, проте представлена функція не є диференційованою. У такому випадку її заміняють на мажоруючу функцію, наприклад $(1 - M)^2$. Після цього обчислюються похідні, знаходяться значення $\omega_0, \omega_1, \omega_2, \ldots, \omega_n$, які відповідають мінімуму. Значення цих ваг дозволяють більш точно виконувати класифікацію об'єктів вибірки.

Складність полягає у тому, що не завжди можна отримати при диференціюванні систему, яку легко розв'язати класичними способами. У таких випадках використовуються градієнтні методи для розрахунку вказаних значень ваг, зокрема метод градієнтного спуску.

Приклад 2.5.

Ще один приклад використання бази даних ірисів для побудови класифікатора.

```
import sklearn as sk
```

```
iris = sk.datasets.load_iris()
X_iris, Y_iris = iris.data, iris.target

from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_iris, Y_iris, test_size=0.30,
random_state=33)

from sklearn.linear_model import SGDClassifier
clf = SGDClassifier()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print(y_pred)

from sklearn import metrics

print (metrics.classification_report(y_test, y_pred, target_names=iris.target_names))
```

Лабораторна робота № 7.

Тема. Задачі класифікації.

Мета. Набути умінь та навичок розробки програм для систем класифікації за допомогою мови програмування Python.

Програмне забезпечення. Програмний комплекс Anaconda для версії Python 3.7.

Контрольні запитання.

1. Задачі класифікації.
2. Метод класифікації kNN.
3. Використання дерева рішень для класифікації об’єктів.
4. Поняття лінійного класифікатора.

Завдання до лабораторного заняття

1. Завантажити базу даних Iris, вбудовану в модуль *scikit-learn*.
2. Виконати нормалізацію даних.
3. Розбити дані на тренувальну та тестову вибірку
4. Побудувати моделі машинного навчання для класифікації об’єктів за допомогою метода kNN, дерева рішень та лінійного класифікатора.

5. Вивести у консоль показники ефективності кожного представлених методів.
6. Порівняти їх, вказати переваги та недоліки для кожного з них.

ДОДАТОК 1
Перелік корисних джерел із штучного інтелекту, машинного навчання та аналізу даних

Література

1. Вандер П. Дж. Python для сложных задач: наука о данных и машинное обучение / П. Дж. Вандер. – СПб.: Питер, 2018 г. – 576 с.

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

3. Коэльо Л. В, Ричарт В. Построение снстем машишюго обучения на языке Python / Пер. с англ. Слинкии А. А. – М.: ДМК Пресс, 2016. – 302 с.

4. Нікольський Ю. В. Системи штучного інтелекту: навчальний посібник / Ю. В. Нікольський, В. В. Пасічник, Ю. М. Щербина. – Львів: «Магнолія-2006», 2015 р. – 279 с.

5. Ямпольський Л. С. Системи штучного інтелекту в плануванні, моделюванні та управлінні :підручник для студентів вищ. навч. закл. / Л. С. Ямпольський, Б П. Ткач, О. І. Лісовиченко. – К.:ДП Видавничий дім «Персонал», 2011. – 544 с.

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

7. Mitchell T. M. Machine Learning / T. M. Mitchell. – McGraw-Hill, 1997. – 432 p.

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

Корисні посилання

1. Все про нейромережі. – Електронний ресурс: https://neurohive.io.

2. Вступ до машинного навчання за допомогою Python і Scikit-Learn. – Електронний ресурс: https://habr.com/ru/company/mlclass/blog/247751/.

3. Вступ до машинного навчання. – Електронний ресурс: https://www.coursera.org/learn/vvedenie-mashinnoe-obuchenie#faqs

4. Машинне навчання – це легко. – Електронний ресурс: https://habr.com/ru/post/319288/.

5. Машинне навчання. – Електронний ресурс: https://courses.prometheus.org.ua/courses/IRF/ML101/2016_T3/about.

ДОДАТОК 2

База даних для виконання лабораторної роботи № 4

	Cost($)	product_1(kg)	product_2(g)	product_3(quantity)	product_4(quantity)
User 1	50	2	200	3	5
User 2	114,15	7	N/A	3	6
User 3	106,15	6	500	3	6
User 4	N/A	7	1500	3	N/A
User 5	25,65	1	200	2	1
User 6	109	3	300	15	5
User 7	90,25	1	3000	2	5
User 8	N/A	1	N/A	2	15
User 9	24,55	0,5	15	2	5
User 10	47	2	50	3	5
User 11	58,45	3	50	4	3
User 12	28,65	0,1	300	4	3
User 13	68,4	3,5	60	5	3
User 14	91,75	6	75	2	N/A
User 15	31,45	N/A	60	N/A	5
User 16	39,1	2	30	1	5
User 17	75,9	3	600	4	6
User 18	65,1	3	60	4	6
User 19	N/A	8	N/A	4	6
User 20	64,9	3	50	4	6
User 21	57,15	2	75	5	6
User 22	56,85	N/A	60	5	6
User 23	129,55	7	50	5	12
User 24	26,35	0,1	50	N/A	6
User 25	30,1	0,1	50	4	6
User 26	51,25	2	75	4	5
User 27	59,35	2	50	4	9
User 28	44,3	2	N/A	4	2
User 29	44,3	2	50	4	2
User 30	77,4	1,5	50	15	N/A

ДОДАТОК 3
База даних для виконання лабораторної роботи № 5

	Time_s	Positive_count	Negative_count
Video1	68	250	125
Video2	126	215	12
Video3	39	300	156
Video4	38	312	65
Video5	65	256	N/A
Video6	90	236	69
Video7	60	236	65
Video8	62	254	65
Video9	63	250	63
Video10	600	250	62
Video11	60	267	65
Video12	92	900	68
Video13	82	301	69
Video14	N/A	32	63
Video15	60	35	63
Video16	60	32	N/A
Video17	62	N/A	654
Video18	63	12	12
Video19	12	352	256
Video20	32	256	236
Video21	86	256	65
Video22	N/A	254	668
Video23	93	253	65
Video24	91	251	68
Video25	94	251	65
Video26	100	N/A	65
Video27	120	251	N/A
Video28	93	236	69
Video29	N/A	2364	697
Video30	85	254	65
Video31	90	214	68
Video32	85	258	69
Video33	93	259	N/A
Video34	90	257	69
Video35	93	259	65
Video36	94	257	63
Video37	N/A	256	64
Video38	93	254	67
Video39	93	254	63
Video40	93	254	67

ДОДАТОК 4

База даних для виконання лабораторних робіт № 6 та № 7

Ірриси Фішера (англ. Iris flower data set) – це важливий набір даних для задачі класифікації, на основі якого англійський статистик та біолог Рональд Фішер році показав роботу суть створеного ним методу дискримінантного аналізу. На даний час зазначений набір даних став класичним і дуже часто використовується у спеціалізованій літературі для демонстрації роботи різних алгоритмів машинного навчання. Також він є базовим набором для бібліотеки *scikit-learn.*

Завантажити цю базу даних можна за посиланням: http://archive.ics.uci.edu/ml/datasets/Iris

sepal.length	sepal.width	petal.length	petal.width	variety
5,1	3,5	1,4	0,2	Setosa
4,9	3	1,4	0,2	Setosa
4,7	3,2	1,3	0,2	Setosa
4,6	3,1	1,5	0,2	Setosa
5	3,6	1,4	0,2	Setosa
5,4	3,9	1,7	0,4	Setosa
4,6	3,4	1,4	0,3	Setosa
5	3,4	1,5	0,2	Setosa
4,4	2,9	1,4	0,2	Setosa
4,9	3,1	1,5	0,1	Setosa
5,4	3,7	1,5	0,2	Setosa
4,8	3,4	1,6	0,2	Setosa
4,8	3	1,4	0,1	Setosa
4,3	3	1,1	0,1	Setosa
5,8	4	1,2	0,2	Setosa
5,7	4,4	1,5	0,4	Setosa
5,4	3,9	1,3	0,4	Setosa
5,1	3,5	1,4	0,3	Setosa
5,7	3,8	1,7	0,3	Setosa
5,1	3,8	1,5	0,3	Setosa
5,4	3,4	1,7	0,2	Setosa
5,1	3,7	1,5	0,4	Setosa
4,6	3,6	1	0,2	Setosa
5,1	3,3	1,7	0,5	Setosa
4,8	3,4	1,9	0,2	Setosa
5	3	1,6	0,2	Setosa
5	3,4	1,6	0,4	Setosa
5,2	3,5	1,5	0,2	Setosa
5,2	3,4	1,4	0,2	Setosa
4,7	3,2	1,6	0,2	Setosa

sepal.length	sepal.width	petal.length	petal.width	variety
4,8	3,1	1,6	0,2	Setosa
5,4	3,4	1,5	0,4	Setosa
5,2	4,1	1,5	0,1	Setosa
5,5	4,2	1,4	0,2	Setosa
4,9	3,1	1,5	0,2	Setosa
5	3,2	1,2	0,2	Setosa
5,5	3,5	1,3	0,2	Setosa
4,9	3,6	1,4	0,1	Setosa
4,4	3	1,3	0,2	Setosa
5,1	3,4	1,5	0,2	Setosa
5	3,5	1,3	0,3	Setosa
4,5	2,3	1,3	0,3	Setosa
4,4	3,2	1,3	0,2	Setosa
5	3,5	1,6	0,6	Setosa
5,1	3,8	1,9	0,4	Setosa
4,8	3	1,4	0,3	Setosa
5,1	3,8	1,6	0,2	Setosa
4,6	3,2	1,4	0,2	Setosa
5,3	3,7	1,5	0,2	Setosa
5	3,3	1,4	0,2	Setosa
7	3,2	4,7	1,4	Versicolor
6,4	3,2	4,5	1,5	Versicolor
6,9	3,1	4,9	1,5	Versicolor
5,5	2,3	4	1,3	Versicolor
6,5	2,8	4,6	1,5	Versicolor
5,7	2,8	4,5	1,3	Versicolor
6,3	3,3	4,7	1,6	Versicolor
4,9	2,4	3,3	1	Versicolor
6,6	2,9	4,6	1,3	Versicolor
5,2	2,7	3,9	1,4	Versicolor
5	2	3,5	1	Versicolor
5,9	3	4,2	1,5	Versicolor
6	2,2	4	1	Versicolor
6,1	2,9	4,7	1,4	Versicolor
5,6	2,9	3,6	1,3	Versicolor
6,7	3,1	4,4	1,4	Versicolor
5,6	3	4,5	1,5	Versicolor
5,8	2,7	4,1	1	Versicolor
6,2	2,2	4,5	1,5	Versicolor
5,6	2,5	3,9	1,1	Versicolor
5,9	3,2	4,8	1,8	Versicolor
6,1	2,8	4	1,3	Versicolor
6,3	2,5	4,9	1,5	Versicolor
6,1	2,8	4,7	1,2	Versicolor
6,4	2,9	4,3	1,3	Versicolor

sepal.length	sepal.width	petal.length	petal.width	variety
6,6	3	4,4	1,4	Versicolor
6,8	2,8	4,8	1,4	Versicolor
6,7	3	5	1,7	Versicolor
6	2,9	4,5	1,5	Versicolor
5,7	2,6	3,5	1	Versicolor
5,5	2,4	3,8	1,1	Versicolor
5,5	2,4	3,7	1	Versicolor
5,8	2,7	3,9	1,2	Versicolor
6	2,7	5,1	1,6	Versicolor
5,4	3	4,5	1,5	Versicolor
6	3,4	4,5	1,6	Versicolor
6,7	3,1	4,7	1,5	Versicolor
6,3	2,3	4,4	1,3	Versicolor
5,6	3	4,1	1,3	Versicolor
5,5	2,5	4	1,3	Versicolor
5,5	2,6	4,4	1,2	Versicolor
6,1	3	4,6	1,4	Versicolor
5,8	2,6	4	1,2	Versicolor
5	2,3	3,3	1	Versicolor
5,6	2,7	4,2	1,3	Versicolor
5,7	3	4,2	1,2	Versicolor
5,7	2,9	4,2	1,3	Versicolor
6,2	2,9	4,3	1,3	Versicolor
5,1	2,5	3	1,1	Versicolor
5,7	2,8	4,1	1,3	Versicolor
6,3	3,3	6	2,5	Virginica
5,8	2,7	5,1	1,9	Virginica
7,1	3	5,9	2,1	Virginica
6,3	2,9	5,6	1,8	Virginica
6,5	3	5,8	2,2	Virginica
7,6	3	6,6	2,1	Virginica
4,9	2,5	4,5	1,7	Virginica
7,3	2,9	6,3	1,8	Virginica
6,7	2,5	5,8	1,8	Virginica
7,2	3,6	6,1	2,5	Virginica
6,5	3,2	5,1	2	Virginica
6,4	2,7	5,3	1,9	Virginica
6,8	3	5,5	2,1	Virginica
5,7	2,5	5	2	Virginica
5,8	2,8	5,1	2,4	Virginica
6,4	3,2	5,3	2,3	Virginica
6,5	3	5,5	1,8	Virginica
7,7	3,8	6,7	2,2	Virginica
7,7	2,6	6,9	2,3	Virginica
6	2,2	5	1,5	Virginica

sepal.length	sepal.width	petal.length	petal.width	variety
6,9	3,2	5,7	2,3	Virginica
5,6	2,8	4,9	2	Virginica
7,7	2,8	6,7	2	Virginica
6,3	2,7	4,9	1,8	Virginica
6,7	3,3	5,7	2,1	Virginica
7,2	3,2	6	1,8	Virginica
6,2	2,8	4,8	1,8	Virginica
6,1	3	4,9	1,8	Virginica
6,4	2,8	5,6	2,1	Virginica
7,2	3	5,8	1,6	Virginica
7,4	2,8	6,1	1,9	Virginica
7,9	3,8	6,4	2	Virginica
6,4	2,8	5,6	2,2	Virginica
6,3	2,8	5,1	1,5	Virginica
6,1	2,6	5,6	1,4	Virginica
7,7	3	6,1	2,3	Virginica
6,3	3,4	5,6	2,4	Virginica
6,4	3,1	5,5	1,8	Virginica
6	3	4,8	1,8	Virginica
6,9	3,1	5,4	2,1	Virginica
6,7	3,1	5,6	2,4	Virginica
6,9	3,1	5,1	2,3	Virginica
5,8	2,7	5,1	1,9	Virginica
6,8	3,2	5,9	2,3	Virginica
6,7	3,3	5,7	2,5	Virginica
6,7	3	5,2	2,3	Virginica
6,3	2,5	5	1,9	Virginica
6,5	3	5,2	2	Virginica
6,2	3,4	5,4	2,3	Virginica
5,9	3	5,1	1,8	Virginica