МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МІСЬКОГО ГОСПОДАРСТВА імені О. М. БЕКЕТОВА

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ

до проведення практичних занять та організації самостійної роботи
з навчальної дисципліни

«ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ»

(для здобувачів першого (бакалаврського) рівня вищої освіти
денної форми навчання зі спеціальностей 122, F3 – Комп'ютерні науки,
126, F6 – Інформаційні системи та технології)

Харків
ХНУМГ ім. О. М. Бекетова
2025

Методичні рекомендації до проведення практичних занять та організації самостійної роботи із навчальної дисципліни «Інтелектуальний аналіз даних» (для здобувачів першого (бакалаврського) рівня вищої освіти денної форми навчання зі спеціальностей 122, F3 – Комп’ютерні науки, 126, F6 – Інформаційні системи та технології) / Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова ; уклад. : В. М. Бредіхін, Ю. В. Левіков. – Харків : ХНУМГ ім. О. М. Бекетова, 2025. – 44 с.

Укладачі: канд. техн. наук, доц. В. М. Бредіхін,
ст. викл. Ю. В. Левіков

Рецензент

О. Б. Костенко, кандидат фізико-математичних наук, доцент кафедри комп’ютерних наук та інформаційних технологій Харківського національного університету міського господарства імені О. М. Бекетова

Рекомендовано кафедрою комп’ютерних наук та інформаційних технологій, протокол № 1 від 29 серпня 2021 р.

ЗМІСТ

ВСТУП

На сьогодні елементи штучного інтелекту активно впроваджуються у практичну діяльність. На відміну від традиційних систем штучного інтелекту технологія інтелектуального пошуку та аналізу даних або «добування даних» (Data Mining) не намагається моделювати природний інтелект, а посилює його можливості потужністю сучасних обчислювальних серверів, пошукових систем і сховищ даних. Технологія Data Mining є одним із етапів технології «виявлення знань у базах даних» – Knowledge Discovery in Databases.

Класичне визначення технології «добування даних» (Data Mining) – це виявлення у початкових даних раніше невідомих, нетривіальних, практично корисних та доступних інтерпретації знань, які необхідні для ухвалення рішень у різних сферах людської діяльності.

Усі завдання, які вирішуються методами Data Mining, можна умовно розділити на п’ять класів:

1. Класифікація (Classification) – встановлення функціональної залежності між вхідними та дискретними вихідними змінними. За допомогою класифікації вирішується задача приналежності об’єктів (спостережень, подій) до одного з наперед відомих класів. Це робиться за допомогою аналізу вже класифікованих об’єктів і формулювання певного набору правил. Класифікація використовується, якщо заздалегідь відомі класи приналежності об’єктів.

2. Кластеризація (Clustering) – це групування об’єктів (спостережень, подій) на основі даних (властивостей), що описують сутність об’єктів. Об’єкти всередині кластера повинні бути «схожими» один на одного і відрізнятися від об’єктів, які увійшли до інших кластерів. Чим більше схожі об’єкти всередині кластера і чим більше відмінностей між кластерами, тим точніше кластеризація. Часто щодо економічних завдань замість кластеризації вживають термін сегментація.

3. Регресія (Regression) – встановлення функціональної залежності між вхідними і неперервними вихідними змінними. Прогнозування найчастіше зводиться до вирішення задачі регресії. До цього ж типу завдань належить прогнозування часової (тимчасової) низки на основі історичних даних.

4. Асоціація (Associations) – виявлення закономірностей між пов’язаними подіями. Прикладом такої закономірності є правило, яке вказує, що з події X випливає подія Y. Такі правила називаються асоціативними. Вперше це завдання було запропоновано для знаходження типових шаблонів покупок, здійснених у супермаркетах, тому іноді її ще називають аналізом ринкового кошика (market basket analysis).

5. Послідовні шаблони (Sequence) – встановлення закономірностей між пов’язаними в часі подіями. Послідовні шаблони можуть бути використані під час планування продажів або надання послуг.

Практична робота № 1

Ознайомлення з Jupyter Notebook

Мета роботи – вивчення можливостей роботи Jupyter Notebook та встановлення бібліотек для проведення інтелектуального аналізу даних.

Завдання до виконання практичної роботи – ознайомитись з процесом запуску та інтерфейсом Jupyter Notebook.

1. Інтерфейс Jupyter Notebook. У **Jupyter Notebook** є два важливі терміни: **cells** (комірки) і **kernels** (ядра) що є ключем до розуміння Jupyter.

Kernel (ядро) – це «обчислювальний двигун», який виконує код, що міститься в документі Notebook.

Cell (комірка) – це контейнер для тексту, який відображатиметься в Notebook, або код, який виконуватиметься ядром записника.

2. Запуск та переривання виконання коду в Jupyter Notebook.

Якщо ваша програма зависла, можна перервати її виконання обравши на панелі меню пункт Kernel -> Interrupt (рис. 1.1).

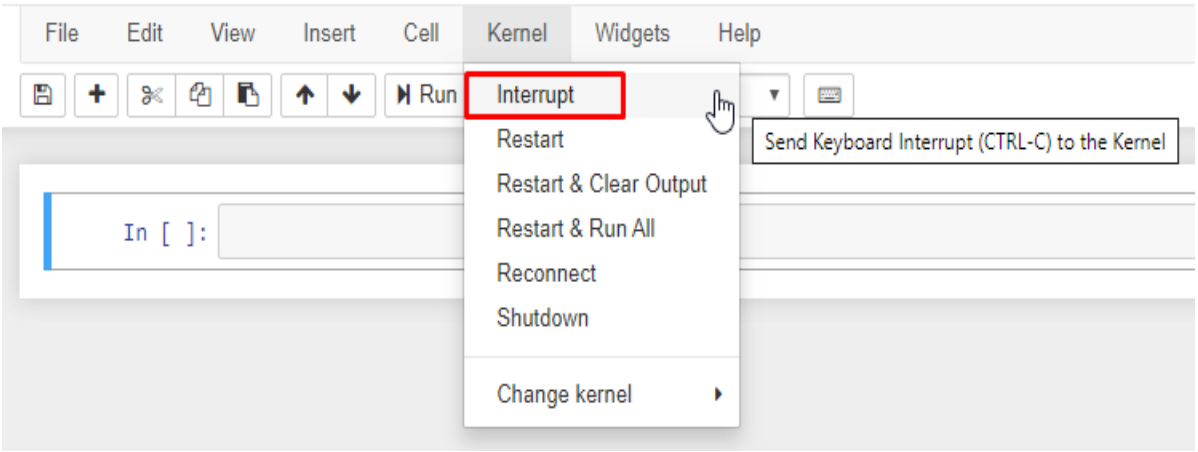

Рисунок 1.1 – Переривання виконання програми

Для додавання нового осередку використовуйте Insert -> Insert Cell Above та Insert -> Insert Cell Below (рис. 1.2).

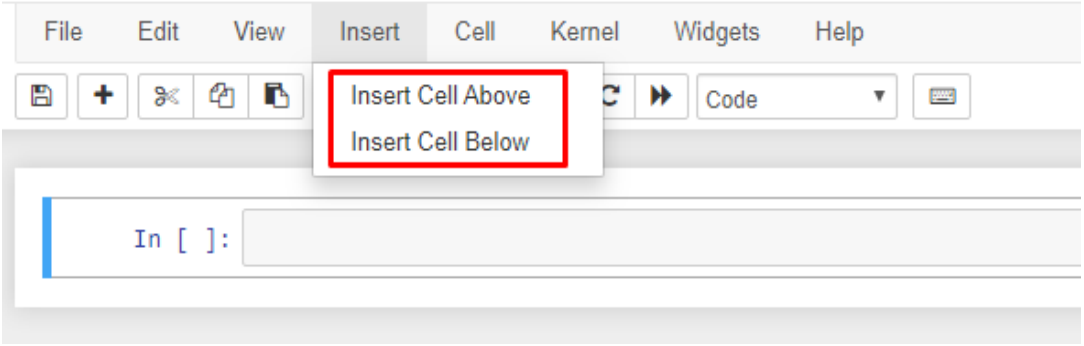

Рисунок 1.2 – Додавання нового осередку

Для запуску осередку використовуйте команди з меню **Cell** (рис. 1.3) або такі комбінації клавіш:

- **Ctrl + Enter –** виконати вміст комірки;
- **Shift + Enter –** виконати вміст комірки та перейти на комірку нижче;
- **Alt + Enter –** виконати вміст осередку і вставити новий осередок нижче.

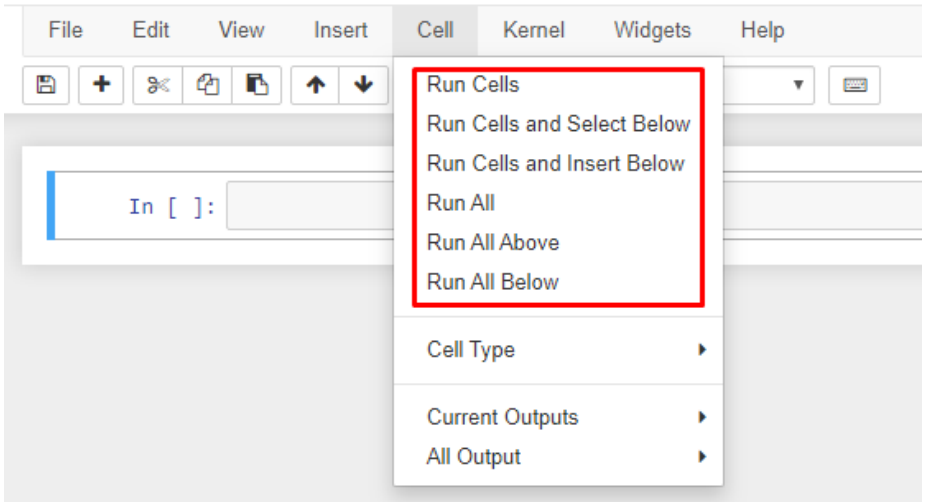

Рисунок 1.3 – Додавання нового осередку

3. Гарячі клавіші. Поєднання клавіш є дуже популярним аспектом середовища Jupyter, оскільки вони забезпечують швидкий робочий процес на

основі осередків. Багато з цих дій можна виконувати в активному осередку, коли вона знаходиться в командному режимі.

Нижче подано список деяких сполучень клавіш Jupyter.

Здійсніть перемикання між режимом редагування та командним режимом за допомогою **Esc** та **Enter** відповідно.

У командному режимі:

– прокрутіть свої осередки вгору та вниз за допомогою клавіш «**вгору**» та «**вниз**»;

– натисніть **A** або **B**, щоб вставити новий осередок вище або нижче активного осередку;

– **M** перетворить активну комірку на комірку Markdown;

– **Y** встановить активну комірку в кодову комірку;

– **D** + **D** (**D** двічі) видалить активний осередок;

– **Z** скасує видалення комірки.

Утримуйте **Shift** та натисніть **«вгору»** або **«вниз»**, щоб вибрати кілька осередків одночасно.

Із виділенням кількох осередків **Shift + M** об’єднає вибрані осередки. **Ctrl + Shift +** в режимі редагування розділить активну комірку курсора. Ви також можете натиснути та **Shift + клік** на полях зліва від ваших осередків, щоб вибрати їх.

4. Щоб потім не відволікатися, поставимо інші необхідні пакети. Відкриємо термінал і виконаємо:

– pip install numpy;

– pip install pandas;

– pip install matplotlib;

– pip install sklearn.

Практична робота № 2

Ознайомлення з бібліотекою NumPy

Мета роботи – вивчення можливостей роботи з бібліотекою NumPy для проведення інтелектуального аналізу даних.

Завдання до виконання практичної роботи – вивчити основи роботи з масивами, базові операції та індексацію даних за допомогою бібліотеки NumPy.

NumPy – відкрита бібліотека, що додає підтримку n-мірних масивів у Python та величезну кількість швидких зручних функцій для роботи з ними.

NumPy надає свій тип даних до роботи з багатовимірними масивами – ndarray. Вимірювання масиву називаються осями (axis). Наприклад, матриця 2 × 2 типу ndarray має дві осі (axis).

2.1 Створення масивів NumPy

Для отримання кількості осей у масиві (тут і далі під словом масив будемо на увазі масив NumPy), необхідно звернутися до властивості **ndim**:

```
In [1]: import numpy as np

In [6]: a1 = np.array([1,3,4,5,6])
        a1.ndim

Out[6]: 1
```

Досліджуємо властивість ndim. У наведеному коді створюємо масив **a1** зі списку чисел. Звертаємось до якості масиву ndim, отримуємо одновимірний масив.

Створимо двовимірний масив. Для цього як параметр методу array будемо використовувати не список, а список списків:

```
In [9]: a2 = np.array([[1,2,3,4,5], [5,4,3,2,1]])
        a2.ndim

Out[9]: 2
```

Щоб дізнатися кількість осей, а також кількість елементів, використовують властивість **shape**. Для наших списків воно виглядає так:

```
In [11]: a1.shape
Out[11]: (5,)

In [12]: a2.shape
Out[12]: (2, 5)
```

Досліджуємо властивість shape. Список a1 є однією віссю з п’яти елементів. Список a2 – одна вісь із двох елементів, тобто із двох рядків, і друга вісь з п’яти елементів, тобто з п’яти стовпців.

Існує чимало спеціальних масивів, що застосовуються у різних розрахунках: одиничні, нульові, порожні тощо. Щоб отримати масив, що складається строго з одних одиниць, використовується метод **on((n,m)): np.ones((5, 6)).**

Щоб отримати масив, що складається строго з одних нулів, використовується метод **zeros((n,m))**. Його синтаксис аналогічний до попереднього. Використання методів ones() та zeros():

```
In [2]: np.ones((5, 6))
Out[2]: array([[1., 1., 1., 1., 1., 1.],
               [1., 1., 1., 1., 1., 1.],
               [1., 1., 1., 1., 1., 1.],
               [1., 1., 1., 1., 1., 1.],
               [1., 1., 1., 1., 1., 1.]])

In [3]: np.zeros((5, 6))
Out[3]: array([[0., 0., 0., 0., 0., 0.],
               [0., 0., 0., 0., 0., 0.],
               [0., 0., 0., 0., 0., 0.],
               [0., 0., 0., 0., 0., 0.],
               [0., 0., 0., 0., 0., 0.]])
```

Квадратна матриця, що складається з одиниць на головній діагоналі та інших нулів, називається одиничною і будується за допомогою методу identity(n). Єдиний вхідний параметр – кількість рядків та стовпців: **np.identity(3).**

Порожня матриця створюється з допомогою методу **empty((n, m))**. Виклик цього методу повертає матрицю до такого вигляду:

```
In [4]: np.empty((3,4))
Out[4]: array([[1.09030547e-311, 2.47032823e-322, 0.00000000e+000,
                0.00000000e+000],
               [9.34609789e-307, 1.61590357e+184, 6.50334102e-091,
                1.31328938e-071],
               [3.57281157e-062, 6.88488155e-042, 3.99910963e+252,
                3.27738599e+179]])
```

Зверніть увагу, що матриця складається не з нулів, а дуже маленьких чисел типу float.

У NumPy є аналог стандартної функції range: np.arange (<start>, stop, <step>). Він повертає рівномірно розподілені значення заданому інтервалу і може обирати як цілі числа, і числа з плаваючою точкою. Наприклад:

– **np.arange(3, 22, 3);**

– **np.arange(1.5, 6.7, 0.7).**

Побудова діапазонів за допомогою методу arange() виглядає так:

```
In [5]: np.arange(3,22,3)
Out[5]: array([ 3,  6,  9, 12, 15, 18, 21])

In [6]: np.arange(1.5, 6.7, 0.7)
Out[6]: array([1.5, 2.2, 2.9, 3.6, 4.3, 5. , 5.7, 6.4])
```

Метод **reshape(a, newshape, order='C')** дозволяє переробити форму (кількість осей) існуючого масиву. Для наочності використовуватимемо такий масив і наведемо кілька прикладів:

1) **A = np.arange(8)** # піддослідний одновимірний масив;

2) **A.reshape((4,2))** # змінюємо його форму в матрицю з чотирьох рядків та двох стовпців;

3) **A.reshape((2, -1))** # робимо нову форму як матрицю із двох рядків.

Зміна форми масиву за допомогою reshape() виглядає так:

```
In [12]: A = np.arange(8)

In [13]: A
Out[13]: array([0, 1, 2, 3, 4, 5, 6, 7])

In [18]: A.reshape((4,2))
Out[18]: array([[0, 1],
                [2, 3],
                [4, 5],
                [6, 7]])

In [19]: A.reshape((2, -1))
Out[19]: array([[0, 1, 2, 3],
                [4, 5, 6, 7]])
```

NumPy дозволяє транспонувати матриці. Створимо собі матрицю C таким чином: **C = A.reshape((2, -1)).**

Щоб отримати транспоновану матрицю C, потрібно просто звернутися до властивості T у такий спосіб (транспонування матриць):

```
In [23]: C = A.reshape((2, -1))
         C

Out[23]: array([[0, 1, 2, 3],
                [4, 5, 6, 7]])

In [21]: C.T

Out[21]: array([[0, 4],
                [1, 5],
                [2, 6],
                [3, 7]])
```

Часто виникає потреба створити масив, що складається з повторів іншого масиву. Для цього існує метод **numpy.tile (A, reps)**, де A – якийсь масив, а resp – шаблон, за яким потрібно скопіювати масив A. Розглянемо кілька прикладів. Для цього створимо список A у такий спосіб: **A = np.arange(3):**

– виклик np.tile(A, (2,2)) побудує новий масив повторами масиву A двічі по вертикалі та двічі по горизонталі;

– виклик np.tile(A, (3,1)) побудує новий масив повторами масиву A три рази по вертикалі і один раз по горизонталі.

Створення матриць за допомогою методу tile() виглядає так:

```
In [24]: A = np.arange(3)
         A

Out[24]: array([0, 1, 2])

In [25]: np.tile(A, (2,2))

Out[25]: array([[0, 1, 2, 0, 1, 2],
                [0, 1, 2, 0, 1, 2]])

In [27]: np.tile(A, (3, 1))

Out[27]: array([[0, 1, 2],
                [0, 1, 2],
                [0, 1, 2]])
```

2.2 Операції з масивами NumPy

NumPy надає безліч поелементних арифметичних операцій над масивами: додавання, віднімання, множення, поділ, піднесення до степеня, тригонометричні функції, гіперболічні функції.

Поелементна операція впливає на кожен конкретний елемент. Додавання складає елементи, що знаходяться на однакових індексах. Аналогічно працюють віднімання, множення та розподіл.

Створимо матриці D та E розміру 3 x 3. Наприклад, складемо обидві матриці, перемножимо один з одним, перемножимо матрицю E з числом 10:

– **D = np.arange(9).reshape((3,3));**

– **E = np.arange(2, 11).reshape((3,3)).**

```
In [33]: D = np.arange(9).reshape((3,3))
         E = np.arange(2, 11).reshape((3,3))

In [34]: D
Out[34]: array([[0, 1, 2],
                [3, 4, 5],
                [6, 7, 8]])

In [35]: E
Out[35]: array([[ 2,  3,  4],
                [ 5,  6,  7],
                [ 8,  9, 10]])

In [47]: D+E
Out[47]: array([[ 2,  4,  6],
                [ 8, 10, 12],
                [14, 16, 18]])

In [48]: D*E
Out[48]: array([[ 0,  3,  8],
                [15, 24, 35],
                [48, 63, 80]])

In [49]: E*10
Out[49]: array([[ 20,  30,  40],
                [ 50,  60,  70],
                [ 80,  90, 100]])
```

2.3 Математичні операції над масивами NumPy

Варто окремо відзначити, що вибір знака для поелементного перемноження матриць виглядає дивним. Від нього, згідно з правилами лінійної алгебри, інтуїтивно очікується інша поведінка – перемноження матриць за такими правилами: кожний елемент результуючої матриці – це сума утворень кожного елемента відповідного рядка у першій матриці з відповідним елементом з другої колонки. Він дозволяє поелементно перемножити дві матриці.

Для класичного множення NumPy використовується метод **dot(a, b)**. Застосуємо його до матриць D та E: **np.dot(D,E):**

```
In [50]: np.dot(D,E)
Out[50]: array([[ 21,  24,  27],
                [ 66,  78,  90],
                [111, 132, 153]])
```

Щоб знайти максимум матриці можна використовувати метод **max(): E.max().** Максимум матриці виглядає так:

```
In [52]: E.max()
Out[52]: 10
```

Вхідним параметром цього методу є axis – повернення максимуму за відповідним виміром:

– **E.max(axis = 0)** # знаходить максимуми у стовпцях;

– **E.max(axis = 1)** # знаходить максимуми у рядках.

Пошуки максимуму в рядках та стовпцях матриці виглядають так:

```
In [57]: E
Out[57]: array([[ 2,  3,  4],
                [ 5,  6,  7],
                [ 8,  9, 10]])

In [56]: E.max(axis=0)
Out[56]: array([ 8,  9, 10])

In [54]: E.max(axis=1)
Out[54]: array([ 4,  7, 10])
```

Аналогічно працює метод **sum(). E.sum(axis = 0)** # знаходить суму кожного стовпця. **E.sum(axis = 1)** # знаходить суму кожного рядка. Пошук суми рядків та стовпців матриці виглядає так:

```
In [59]: E.sum(axis=0)
Out[59]: array([15, 18, 21])

In [60]: E.sum(axis=1)
Out[60]: array([ 9, 18, 27])
```

2.4 Індексація масивів NumPy

Пакет NumPy підтримує кілька різних способів індексації масивів. Щоб їх розглянути, створимо собі такий масив: **A = np.arange(10).**

Підтримуються класичні зрізи як одновимірних, так і двовимірних масивів. Для демонстрації зрізів двовимірного масиву, застосуємо до масиву A метод reshape, щоб перетворити його на матрицю з п’ятьма рядками та двома стовпцями:

```
In [64]: A = np.arange(10)
         A
Out[64]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [65]: A[3:7]
Out[65]: array([3, 4, 5, 6])

In [66]: A[1:8:2]
Out[66]: array([1, 3, 5, 7])
```

```
In [72]: A=A.reshape((5,-1))
         A
Out[72]: array([[0, 1],
                [2, 3],
                [4, 5],
                [6, 7],
                [8, 9]])

In [73]: A[2:4]
Out[73]: array([[4, 5],
                [6, 7]])
```

2.5 Класичні зрізи Python

Для сортування можна використати логічні операції. Виведемо парні елементи списку від нуля до дев’яти. Для цього замість індексів можна використати логічні вирази. Для зв’язування виразів використовуються функції **np.logical_and() та np.logical_or()**.

A[A % 2 == 0] # виведе всі парні елементи масиву A.

A [np.logical_and (A! = 5, A! = 0)] # виведе всі елементи масиву A, які не дорівнюють нулю та п’яти. Сортування елементів за допомогою логічних операцій відбувається так:

```
In [74]: A = np.arange(10)

In [75]: A[A % 2 == 0]
Out[75]: array([0, 2, 4, 6, 8])

In [77]: A[np.logical_or(A != 5, A != 0)]
Out[77]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

Практична робота № 3

Ознайомлення з бібліотекою Pandas

Мета роботи – вивчення можливостей роботи з бібліотекою Pandas для проведення інтелектуального аналізу даних.

Завдання до виконання практичної роботи – вивчити основи роботи маніпулювання з таблицями та тимчасовими рядами за допомогою бібліотеки Pandas.

Pandas – дуже популярна бібліотека для обробки даних на python. Працює на основі NumPy та надає спеціальні структури даних для маніпулювання з таблицями та тимчасовими рядами.

Як приклад використовуватимемо класичний датасет Titanic зі змагань kaggle. Датасет є таблицею, що постачається у форматі *.csv. Для його читання Pandas надає метод, який перетворює таблицю на формат даних Pandas – Dataframe.

Відкриємо його за допомогою Pandas: **data = pd.read_csv('шлях до файлу titanic.csv').**

Змінна data є датафреймом Pandas і, по суті, таблицею з даними.

Щоб оцінити таблицю, дізнатися про її структуру можна використовувати метод **head(n = 5)**, який повертає перші n рядків датафрейму: **data.head().**

Метод head() виглядає так:

```
In [5]: data.head()
```

Out[5]:

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1	1	Allen, Miss. Elisabeth Walton	female	29.00	0	0	24160	211.3375	B5	S	2	NaN	St Louis, MO
1	1	1	Allison, Master. Hudson Trevor	male	0.92	1	2	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON
2	1	0	Allison, Miss. Helen Loraine	female	2.00	1	2	113781	151.5500	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON
3	1	0	Allison, Mr. Hudson Joshua Creighton	male	30.00	1	2	113781	151.5500	C22 C26	S	NaN	135.0	Montreal, PQ / Chesterville, ON
4	1	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.00	1	2	113781	151.5500	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON

Для отримання імен колонок є властивість **columns: data.columns.**

Імена колонок датафрейму виглядають так:

```
In [6]: data.columns
Out[6]: Index(['pclass', 'survived', 'name', 'sex', 'age', 'sibsp', 'parch', 'ticket',
               'fare', 'cabin', 'embarked', 'boat', 'body', 'home.dest'],
              dtype='object')
```

Індексація датафреймів аналогічна стандартній індексації списків у python та індексації NumPy. Індексація датафреймів виглядає так:

```
In [7]: data[4:7]
```

Out[7]:

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
4	1	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0	1	2	113781	151.5500	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON
5	1	1	Anderson, Mr. Harry	male	48.0	0	0	19952	26.5500	E12	S	3	NaN	New York, NY
6	1	1	Andrews, Miss. Kornelia Theodosia	female	63.0	1	0	13502	77.9583	D7	S	10	NaN	Hudson, NY

Крім цього, Pandas надає два методи індексації:

- **iloc** для індексації за номерами стовпців та рядків;
- **loc** для індексації за рядковими значеннями рядків та назвами стовпців.

Наприклад, є завдання отримати стать пасажирів для рядків із першої до сьомої. Використовуємо loc у такий спосіб: **data.loc[1:7, 'sex'].**

Перший параметр – ідентифікатори рядків (у цьому випадку це числа), другий параметр – потрібний стовпець або стовпчики. На скріншоті, наприклад, наведено отримання не тільки статі, а й віку пасажирів з необхідними номерами. Використання loc здійснюється у такий спосіб:

```
In [19]: data.loc[1:7, ['sex', 'age']]
```

Out[19]:

	sex	age
1	male	0.92
2	female	2.00
3	male	30.00
4	female	25.00
5	male	48.00
6	female	63.00
7	male	39.00

Iloc працює аналогічно, але тільки з номерами рядків і стовпців, а не з їх іменами. Отримаємо для прикладу перші два стовпці перших десяти рядків: **data.iloc[:10, :2].** Використання iloc здійснюється у такий спосіб:

```
In [38]: data.iloc[:10, :2]
```

Out[38]:

	pclass	survived
0	1	1
1	1	1
2	1	0
3	1	0
4	1	0
5	1	1
6	1	1
7	1	0
8	1	1
9	1	0

Є можливість звертатися на ім’я до кожного конкретного стовпця. Отримаємо перші п’ять імен пасажирів нашої таблиці: **data['name'].head().** Перші п’ять імен датафрейму виглядають так:

```
In [42]: data['name'].head()
Out[42]: 0                     Allen, Miss. Elisabeth Walton
         1                    Allison, Master. Hudson Trevor
         2                      Allison, Miss. Helen Loraine
         3              Allison, Mr. Hudson Joshua Creighton
         4    Allison, Mrs. Hudson J C (Bessie Waldo Daniels)
         Name: name, dtype: object
```

Під час індексації можна будувати логічні запити даних датафрейму. Наприклад, наведемо п’ять перших жінок-пасажирів: **data[(data['sex'] == 'female')]['name'].head().** Перші п’ять жінок датафрейму виглядають так:

```
In [47]: data[(data['sex'] == 'female')]['name'].head()
Out[47]: 0                     Allen, Miss. Elisabeth Walton
         2                      Allison, Miss. Helen Loraine
         4    Allison, Mrs. Hudson J C (Bessie Waldo Daniels)
         6                  Andrews, Miss. Kornelia Theodosia
         8      Appleton, Mrs. Edward Dale (Charlotte Lamson)
         Name: name, dtype: object
```

Логіка роботи така: отримуємо рядки таблиці, у яких колонка sex = female. Вибираємо тільки стовпчик name і за допомогою методу head() отримуємо верхні п’ять рядків.

Ускладнимо приклад. Отримаємо датафрейм, що містить перші п’ять чоловіків або жінок старше 50-ти років: **data[(data['sex'] == 'women') &**

(data['age'] > 50) | (data['sex'] == 'male')].head(). Результати виконання запиту виглядають так:

```
In [12]: data[(data['sex'] == 'women') & (data['age'] > 50) | (data['sex'] == 'male')].head()
```

Out[12]:

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
1	1	1	Allison, Master. Hudson Trevor	male	0.92	1	2	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON
3	1	0	Allison, Mr. Hudson Joshua Creighton	male	30.00	1	2	113781	151.5500	C22 C26	S	NaN	135.0	Montreal, PQ / Chesterville, ON
5	1	1	Anderson, Mr. Harry	male	48.00	0	0	19952	26.5500	E12	S	3	NaN	New York, NY
7	1	0	Andrews, Mr. Thomas Jr	male	39.00	0	0	112050	0.0000	A36	S	NaN	NaN	Belfast, NI
9	1	0	Artagaveytia, Mr. Ramon	male	71.00	0	0	PC 17609	49.5042	NaN	C	NaN	22.0	Montevideo, Uruguay

Зв’язування умов відбувається за допомогою логічних операторів І (&) та АБО (|).

3.1 Зміна датафреймів Pandas

Нехай потрібно змінити назву колонки. Pandas дозволяє зробити це за допомогою методу rename (). Перейменуємо колонку name на Name. У параметр columns передаємо словник, в якому вказуються пари «попереднє ім’я» – «нове ім’я» колонок. Параметр inplace = True змушує Pandas змінювати поточний датафрейм, а не створювати новий: **data.rename(columns={'name':'Name'}, inplace=True).** Перейменування колонки відбувається так:

```
In [18]: data.rename(columns={'name':'Name'}, inplace=True)
In [17]: data.head()
```

Out[17]:

	pclass	survived	Name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1	1	Allen, Miss. Elisabeth Walton	female	29.00	0	0	24160	211.3375	B5	S	2	NaN	St Louis, MO
1	1	1	Allison, Master. Hudson Trevor	male	0.92	1	2	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON
2	1	0	Allison, Miss. Helen Loraine	female	2.00	1	2	113781	151.5500	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON
3	1	0	Allison, Mr. Hudson Joshua Creighton	male	30.00	1	2	113781	151.5500	C22 C26	S	NaN	135.0	Montreal, PQ / Chesterville, ON
4	1	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.00	1	2	113781	151.5500	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON

Можна застосувати довільну функцію користувача до кожного елемента вибраних стовпців (наприклад, ми можемо отримати прізвища всіх людей із колонки Name).

Функція отримання прізвища виглядає так: **def get_last_name(name): return name.split(',')[0].strip().** Функція виділення прізвища з повного імені та його тестування відбувається так:

```
In [21]: def get_last_name(name):
             return name.split(',')[0].strip()

         get_last_name('Allen, Miss. Elisabeth Walton')
Out[21]: 'Allen'
```

Тепер, щоб застосувати її до всіх елементів датафрейму, використовується метод apply(), за допомогою якого передається об'єкт функції. Наприклад: **last_names = data['Name'].apply(get_last_name).** Використання методу apply відбувається так:

```
In [22]: last_names = data['Name'].apply(get_last_name)

In [24]: last_names.head()
Out[24]: 0      Allen
         1    Allison
         2    Allison
         3    Allison
         4    Allison
         Name: Name, dtype: object
```

А тепер додаймо до нашого датафрейму стовпець прізвищ, які ми зберегли в змінній last_names. Для цього у квадратних дужках вказується ім'я нового стовпця. Додавання нового стовпця відбувається так:

```
In [25]: data['Last name'] = last_names

In [26]: data.head()
Out[26]:
```

	pclass	survived	Name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest	Last name
0	1	1	Allen, Miss. Elisabeth Walton	female	29.00	0	0	24160	211.3375	B5	S	2	NaN	St Louis, MO	Allen
1	1	1	Allison, Master. Hudson Trevor	male	0.92	1	2	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON	Allison
2	1	0	Allison, Miss. Helen Loraine	female	2.00	1	2	113781	151.5500	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON	Allison
3	1	0	Allison, Mr. Hudson Joshua Creighton	male	30.00	1	2	113781	151.5500	C22 C26	S	NaN	135.0	Montreal, PQ / Chesterville, ON	Allison
4	1	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.00	1	2	113781	151.5500	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON	Allison

Щоб видалити стовпець, використовується метод drop(). Видалимо щойно доданий стовпець прізвищ. Параметр axis необхідний, щоб змусити Pandas працювати зі стовпцями. Спробуйте змінити його на нуль і подивитися, що станеться: **data.drop('Last name', axis=1, inplace=True).** Видалення стовпця відбувається так:

```
In [28]: data.drop('Last name', axis=1, inplace=True)
```

```
In [29]: data.head()
```

Out[29]:

	pclass	survived	Name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1	1	Allen, Miss. Elisabeth Walton	female	29.00	0	0	24160	211.3375	B5	S	2	NaN	St Louis, MO
1	1	1	Allison, Master. Hudson Trevor	male	0.92	1	2	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON
2	1	0	Allison, Miss. Helen Loraine	female	2.00	1	2	113781	151.5500	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON
3	1	0	Allison, Mr. Hudson Joshua Creighton	male	30.00	1	2	113781	151.5500	C22 C26	S	NaN	135.0	Montreal, PQ / Chesterville, ON
4	1	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.00	1	2	113781	151.5500	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON

Дуже часто датасети постачаються неповними. Це означає, що у деяких осередках таблиці може не бути даних. У машинному навчанні намагаються або позбутися таких рядків даних, або якось їх заповнити. Для отримання порожніх рядків використовується метод **isnull()**, а отримання непорожніх рядків – метод **notnull()**. Це працює досить інтуїтивно:

– **data['boat'].isnull().head()** # – перші п’ять порожніх осередків стовпця;

– **data[data['boat'].notnull()].head()** # – перші п’ять непорожніх осередків стовпця.

Робота з методами isnull() та notnull() виглядає так:

```
In [37]: data['boat'].isnull().head()
```

```
Out[37]: 0    False
         1    False
         2     True
         3     True
         4     True
         Name: boat, dtype: bool
```

```
In [38]: data[data['boat'].notnull()].head()
```

Out[38]:

	pclass	survived	Name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1	1	Allen, Miss. Elisabeth Walton	female	29.00	0	0	24160	211.3375	B5	S	2	NaN	St Louis, MO
1	1	1	Allison, Master. Hudson Trevor	male	0.92	1	2	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON
5	1	1	Anderson, Mr. Harry	male	48.00	0	0	19952	26.5500	E12	S	3	NaN	New York, NY
6	1	1	Andrews, Miss. Kornelia Theodosia	female	63.00	1	0	13502	77.9583	D7	S	10	NaN	Hudson, NY
8	1	1	Appleton, Mrs. Edward Dale (Charlotte Lamson)	female	53.00	2	0	11769	51.4792	C101	S	D	NaN	Bayside, Queens, NY

Якщо ви працювали з SQL, то вам напевно знайомий метод GROUP BY. Він дозволяє групувати дані за якимось заданим критерієм. Pandas має такі ж можливості. І тому використовується метод groupby().

Згрупуємо пасажирів за статтю залежно від класу каюти та порахуємо їхню кількість: **data.groupby('sex')['pclass'].value_counts().** Кількість пасажирів залежно від класу каюти та статі:

```
In [42]: data.groupby('sex')['pclass'].value_counts()
Out[42]: sex     pclass
         female  3         216
                 1         144
                 2         106
         male    3         493
                 1         179
                 2         171
         Name: pclass, dtype: int64
```

Pandas містить набір статистичних методів, які дозволяють знайти середнє значення, максимум, мінімум, стандартне відхилення будь-якого набору даних із датафрейму. З іншого боку, для швидкої оцінки всіх популярних параметрів використовується метод describe().

Розрахуємо всі ці показники для вартості проїзду пасажирів за різними класами: **data.groupby('pclass')['fare'].describe().** Результат роботи методу describe() виглядає так:

```
In [43]: data.groupby('pclass')['fare'].describe()
```

Out[43]:

pclass	count	mean	std	min	25%	50%	75%	max
1	323.0	87.508992	80.447178	0.0	30.6958	60.0000	107.6625	512.3292
2	277.0	21.179196	13.607122	0.0	13.0000	15.0458	26.0000	73.5000
3	708.0	13.302889	11.494358	0.0	7.7500	8.0500	15.2458	69.5500

У результаті отримуємо таблицю, що містить кількість пасажирів, середню вартість, її стандартне відхилення та інші показники.

Ці функції доступні окремо. Наприклад, визначимо, у якому співвідношенні вижили на Титаніку чоловіки та жінки: **data.groupby('sex')['survived'].mean():**

```
In [49]: data.groupby('sex')['survived'].mean()
Out[49]: sex
         female    0.727468
         male      0.190985
         Name: survived, dtype: float64
```

Після закінчення роботи з датафреймом його можна зберегти у файлах різних форматів. Наприклад, для збереження формату csv використовується

метод to_csv(): **data.to_csv('titanic_2.csv', index=False).** Зберігаємо датафрейм у файл CSV з ім’ям titanic_2.

Практична робота № 4

Ознайомлення з бібліотекою Matplotlib

Мета роботи – вивчення можливостей роботи з бібліотекою Matplotlib для проведення інтелектуального аналізу даних.

Завдання до виконання практичної роботи – вивчити основи візуалізації даних за допомогою бібліотеки **Matplotlib.**

Jupyter Notebook підтримує виведення графіків «на місці». Для активації режиму використовується магічна команда % matplotlib inline.

Для побудови графіків використовується об’єкт plt, який був імпортований з пакета matplotlib.

Першим етапом побудови графіка є створення полотна. Робиться це так. Параметр **figsize** визначає розмір полотна у дюймах. Параметри передаються як кортеж. Числа можуть бути дрібними: **fig = plt.figure(figsize = (10, 6)).**

Після цього додаємо осі, будуємо криві та наносимо легенду на полотно:

– **fig = plt.figure(figsize = (10, 6));**

– **axes = fig.add_axes([0,0,1,1]) #** – створюємо прямокутник для побудови графіків у вигляді списку [left, bottom, width, height];

– **axes.plot(x, x**2, 'r') #** – додати першу криву на полотно червоного кольору;

– **axes.plot(x, x**3, 'b*--') #** – додати другу криву на полотно синього кольору з маркерами іншого типу;

– **axes.set_xlabel('x') #** – додати назву осі X;

– **axes.set_ylabel('y') #** – додати назву осі Y;

– **axes.set_title('Hello proglib') #** – додати назву всього графіка;

– **axes.legend([r'x^2', r'x^3'], loc=0) #** – додати легенду;

– **plt.show() #** – вивести графік на екран.

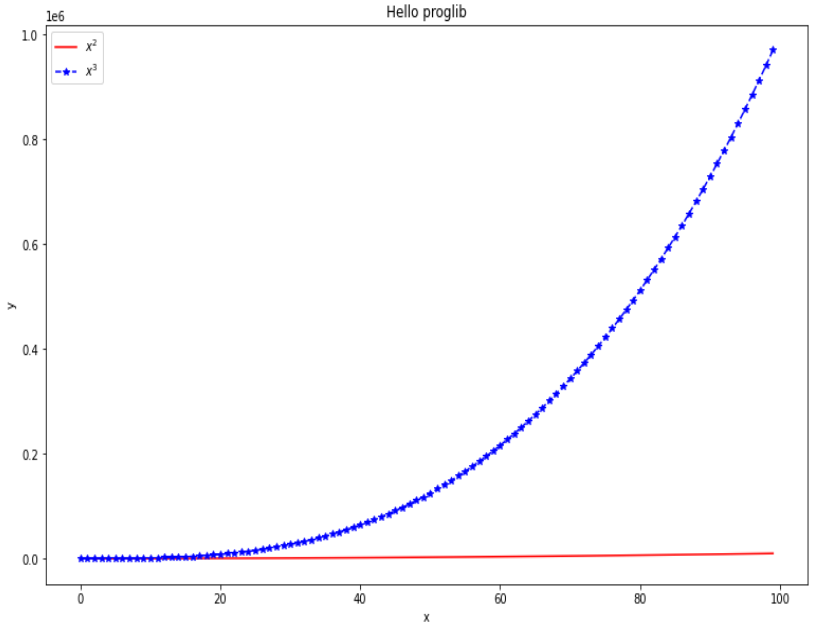

Рисунок 4.1 – Результат побудови двох графіків

Пакет дозволяє будувати сімейства графіків.

1. Графік у графіці. Схема побудови абсолютно така сама, як у попередньому прикладі:

– **fig = plt.figure();**

– **axes1 = fig.add_axes([0.1, 0.1, 0.8, 0.8])** #основний графік;

– **axes2 = fig.add_axes([0.2, 0.5, 0.4, 0.3])** #внутрішній графік (його розмір менший, цифри показують частку від figsize).

#Основний графік:

– axes1.plot(x, x2, 'r');**

– axes1.set_xlabel('x');

– axes1.set_ylabel('y');

– axes1.set_title(«Я зовнішній графік»).

 # Вкладений графік:

– axes2.plot(x2, x, 'b');**

– axes2.set_xlabel('y');

– axes2.set_ylabel('x');

– axes2.set_title('Я внутрішній графік');

– plt.show().

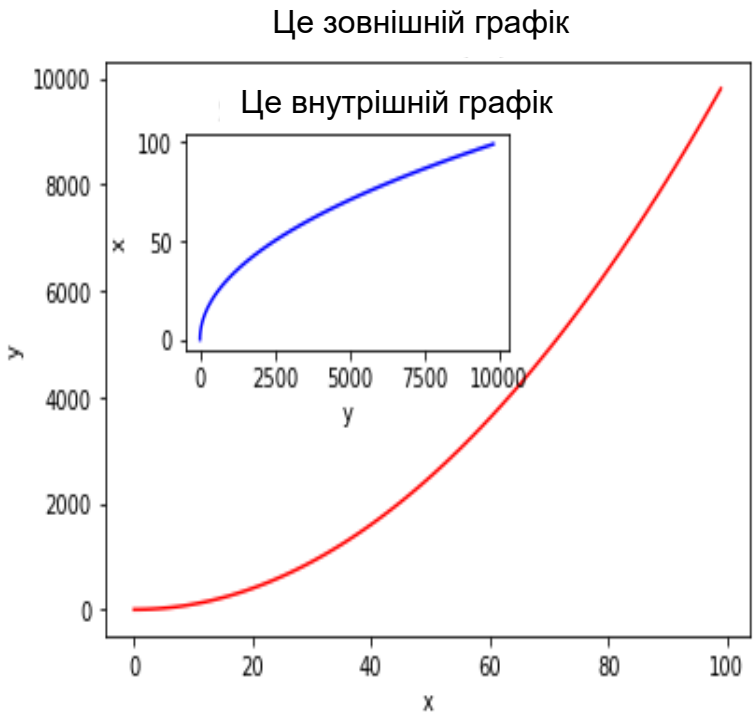

Рисунок 4.2 – Побудова графіка у графіку

2. Сімейство графіків. У цьому випадку, полотно створюється методом subplots(), параметром якого є кортеж чисел, що показують, скільки графіків по горизонталі і по вертикалі він містить. Побудуємо один рядок із трьох графіків: **fig, axes = plt.subplots(nrows = 1, ncols = 3, figsize = (16, 5)).**

For pow, ax in enumerate(axes):

– ax.plot(x, x(pow + 1), 'b');**

– **ax.set_xlabel('x');**

– **ax.set_ylabel('y');**

– **ax.set_title(f'$y = x^{pow + 1}$', fontsize=18);**

– **fig.tight_layout()** #автоматично вписує всі графіки у розмір полотна.

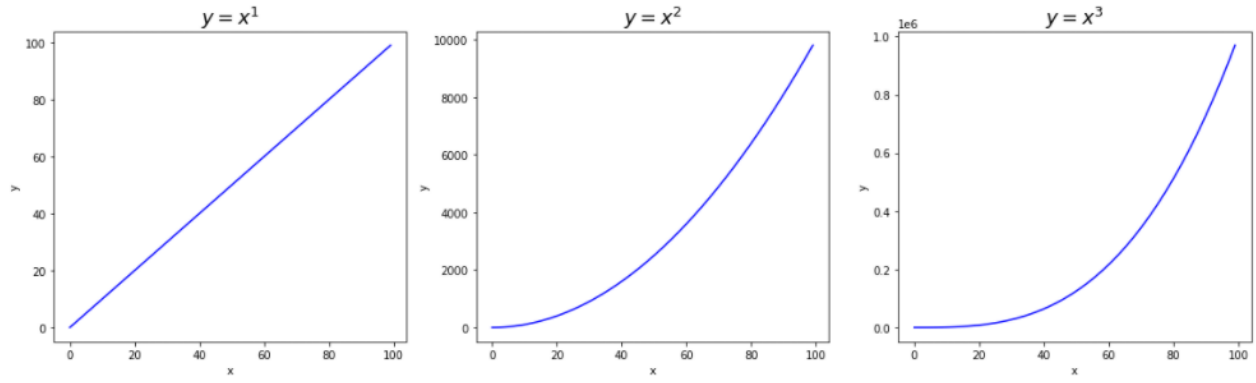

Рисунок 4.3 – Сімейство графіків

Matplotlib підходить для побудови статистичних графіків. Це величезна область, але побудуємо гістограму «розподіл пасажирів за віком для чоловіків та жінок». Як дані використовуємо той самий датасет titanic:

– **fig = plt.figure();**

– **axes = fig.add_axes([0.0, 0.0, 1.0, 1.0]);**

– **bins = 20** # кількість стовпців;

– **index = np.arange(bins)** # створюємо список від 0 до bins – 1;

– **axes.hist(data[data['sex'] == 'male']['age'].dropna(), bins = bins, alpha = 0,6, label = 'Чоловіки')** # додаємо на полотно гістограму розподілу віку серед чоловіків;

– **axes.hist(data[data['sex'] == 'female']['age'].dropna(), bins = bins, alpha = 0,6, label = 'Жінки')** # додаємо на полотно гістограму розподілу віку серед жінок;

– **axes.legend()** # будуємо легенду;

– **axes.set_xlabel('Вік', fontsize=18);**

– **axes.set_ylabel('Кількість', fontsize=18);**

– **axes.set_title('Розподіл віку по статі людини', fontsize=18);**

– **plt.show().**

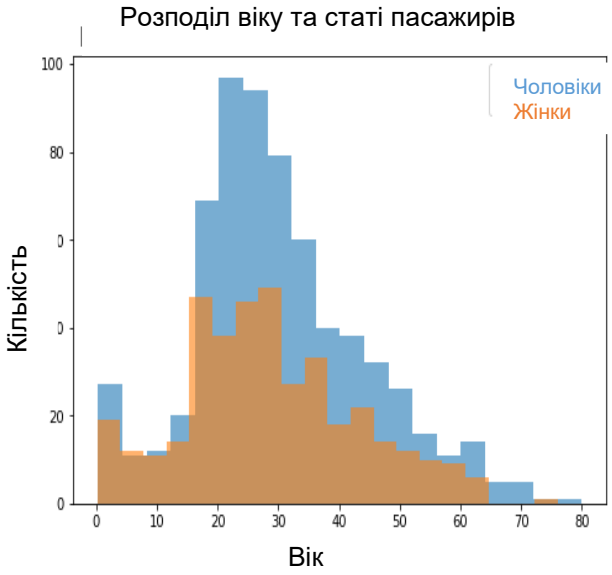

Рисунок 4.4 – Гістограма «Розподіл віку за статтю пасажирів»

Практична робота № 5

Лінійна регресія

Мета роботи – вивчення можливостей роботи з бібліотекою Matplotlib для проведення інтелектуального аналізу даних.

Завдання до виконання лабораторної роботи – вивчити основи візуалізації даних за допомогою бібліотеки Matplotlib.

Щоб швидко виконати лінійну регресію, необхідно використати бібліотеку scikit-learn. Встановіть її за допомогою pip: **pip install scikit-learn.**

Додаємо інші бібліотеки для роботи з даними.

– **import numpy as np;**

– **import matplotlib.pyplot as plt** #Для візуалізації даних;

– **import pandas as pd** #Для зчитування даних;

– **from sklearn.linear_model import LinearRegression.**

Для виконання завдання потрібний NumPy для обчислень, Pandas для імпорту набору даних у форматі csv у цьому випадку і matplotlib для візуалізації наших даних та лінії регресії. Необхідно використовувати клас LinearRegression для виконання лінійної регресії.

Знайдіть набір даних та код: URL: https://github.com/chasinginfinity/ml-from-scratch/tree/master/03%20Linear%20Regression%20in%202%20minutes

Тепер виконаємо регресію:

– **data = pd.read_csv('data.csv')** #Завантажує data set;

– **X = data.iloc[:, 0].values.reshape(-1, 1)** #перетворює дані на numpy масив;

– **Y = data.iloc[:, 1].values.reshape(-1, 1)** #-1 means that calculate dimension of rows, but have 1 column;

– **linear_regressor = LinearRegression()** #create object for the class;

– **linear_regressor.fit(X, Y)** #perform linear regression;

– **Y_pred = linear_regressor.predict(X)** #make predictions.

У нас є наші прогнози у Y_pred. Тепер давайте візуалізуємо набір даних та лінію регресії:

– **plt.scatter(X, Y);**

– **plt.plot(X, Y_pred, color='red');**

– **plt.show().**

Як результат, отримаємо зображення, де набір даних подано синім кольором а лінію регресії – червоним.

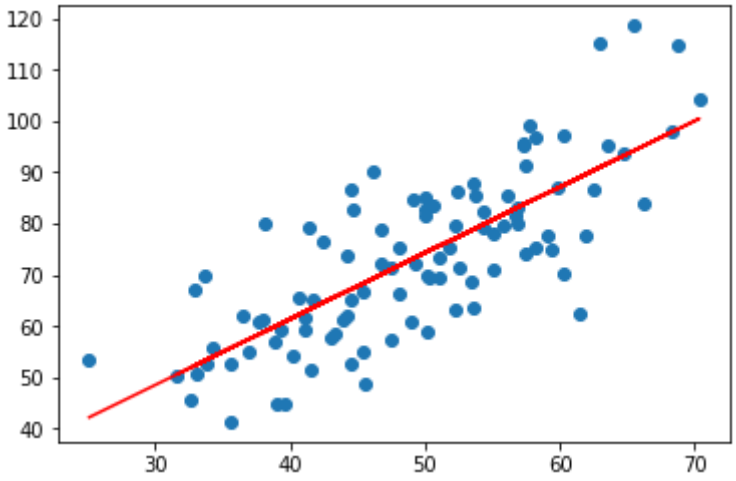

Рисунок 5.1 – Регресійний аналіз

Ви можете використовувати будь-який набір даних на ваш вибір і навіть виконати множинну лінійну регресію (більше однієї незалежної змінної), використовуючи клас LinearRegression в sklearn.linear_model. Також цей клас використовує звичайний метод найменших квадратів для виконання цієї регресії. Це зручний інструмент для того, щоб зробити кілька швидких прогнозів і отримати уявлення про наданий вам набір даних:

1. Крок 1: імпорт набору даних.
2. Крок 2: попередня обробка даних.
3. Крок 3: поширення тесту та наборів поїздів.
4. Крок 4: встановлення лінійної регресії; модель на тренувальний набір.
5. Крок 5: прогноз результатів тесту.
6. Крок 6: візуалізація результатів тесту.

Тепер, коли ми бачили кроки, давайте почнемо з кодування.

Реалізація моделі лінійної регресії у Python

Ми будемо використовувати набір заробітної плати. Наш набір даних матиме два стовпці, а саме – роки досвіду та зарплати.

Посилання на набір даних: URL: https://github.com/content-anu/dataset-simple-linear

Імпорт набору даних

Ми почнемо з імпорту набору даних, використовуючи Pandas, а також імпорту інших бібліотек, таких як NumPy та Matplotlib:

– import numpy as np;

– import pandas as pd;

– import matplotlib.pyplot as plt;

– dataset = pd.read_csv('Salary_Data.csv');

– dataset.head();

– dataset.head().

Показує перші кілька стовпців нашого набору даних. Вихід згаданого фрагмента полягає в такому:

2. Попередня обробка даних. Тепер, коли ми імпортували набір даних, ми будемо виконувати попередню обробку даних:

– x = dataset.iloc[:,:-1].values #independent variable array;

– y = dataset.iloc[:,1].values #dependent variable vector.

X є незалежним змінним масивом і залежним вектором змінної. Зверніть увагу на різницю між масивом та вектором. Залежна змінна має бути у векторі, а незалежна змінна – над масивом.

3. Поділ набору даних. Нам потрібно розмістити наш набір даних у тест та набір поїздів. Зазвичай ми дотримуємося політики 20–80 або політики 30–70 відповідно.

Чому потрібно виконати розщеплення? Це тому, що ми хочемо тренувати нашу модель відповідно до років і зарплати. Потім перевіримо нашу модель на тестовому наборі. Ми перевіряємо, чи зроблені передбачення моделлю на даних

тестових даних, що приведено в набір даних. Якщо це так, то модель точна і робить правильні прогнози:

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=1/3,random_state=0).

Не потрібно застосовувати масштабування функцій для лінійної регресії, оскільки бібліотеки подбають про це.

4. Встановлення лінійної регресії у навчальний набір. Із лінійної моделі модельної моделі Sklearn імпортують лінійний клас регресії. Створіть об’єкт для лінійного регресійного класу під назвою Regressor.

Щоб підписати регрес у тренувальний набір, ми використаємо метод FIT – функцію для відповіді регресам у навчальному наборі.

Потрібно відповідати X_Train (навчальним даним матриці функцій) у цільові значення Y_train. Таким чином, модель вивчає кореляцію та дізнається, як передбачити залежні змінні на основі незалежної змінної:

– **from sklearn.linear_model import LinearRegression;**

– **regressor = LinearRegression();**

– **regressor.fit(X_train,y_train)** #actually produces the linear eqn for the data.

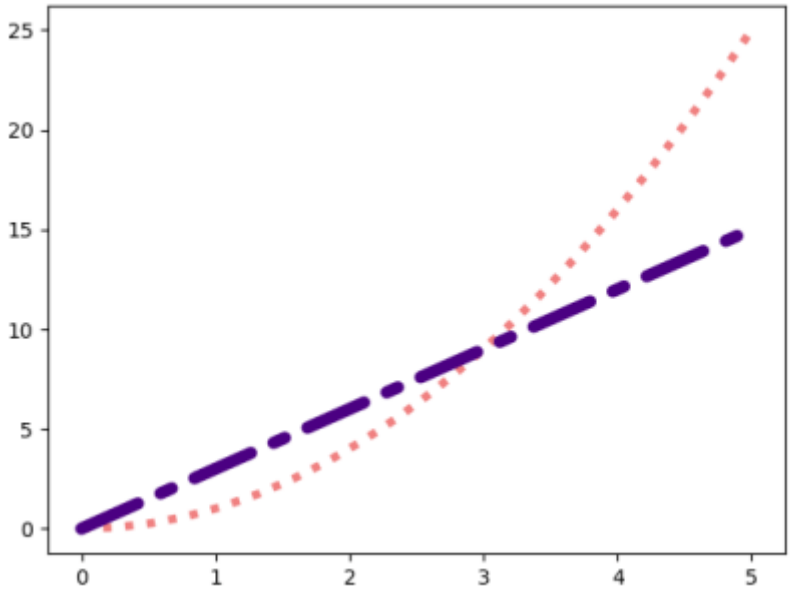

Рисунок 5.2 – Лінійна та нелінійна форма регресійної кривої

5. Прогнозування результатів випробувань. Ми створюємо вектор, що містить усі прогнози заробітної плати випробувань. Прогнозована зарплата потім вкладається у вектор, названий Y_PRED (містить прогноз для всіх спостережень у тестовому наборі).

Прогнозувальний метод робить прогнози для тестового набору. Отже, вхід – це тестовий набір. Параметр для прогнозування повинен бути масивом або матрицею, отже, вхід – X_TEST:

– **y_pred = regressor.predict(X_test):**

– **y_pred.**

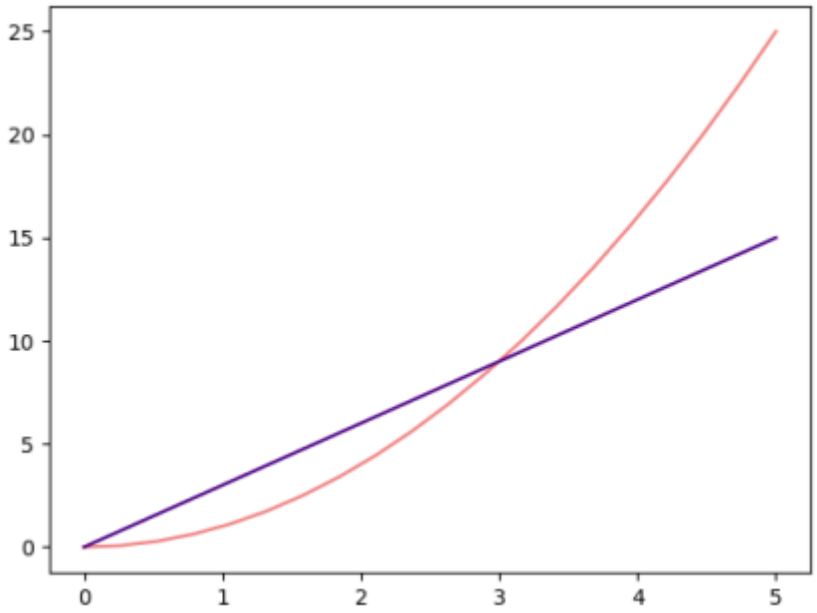

Рисунок 5.3 – Графіки прогнозування результатів випробувань

Y_test:

– Vector 1: [1, -1, 1];

– Vector 2: [4, 2, 3];

– Cross Product: [-5, 1, 6].

Y_test – реальна зарплата випробувального набору, Y_PRED – передбачені зарплати.

6. Візуалізація результатів. Подивимося, як виглядатимуть результати нашого коду, коли ми його візуалізуємо.

Побудова точок (спостереження) відбувається так: щоб візуалізувати дані, ми маємо втілити сюжет графіка за допомогою Matplotlib. Щоб побудувати реальні точки спостереження, тобто побудувати реальні цінності.

Вісь X міститиме багаторічний досвід, а вісь Y міститиме прогнозовані зарплати. Plt.scatter – графіки розсіювання графіка даних. Параметри містять:

– X – координата (X_train: кількість років);

– Y – координата (Y_Train: реальна зарплата працівників);

– колір (лінія регресії в червоній та лінія спостережень в синьому)

Побудова лінії регресії

Plt.plot мають такі параметри:

– X координати (X_TRAIN) – кількість років;

– Y координати (прогнозують на X_TRAIN) – прогноз X-поїзда (залежно від кількох років).

Примітка: y-координата не Y_PRED, тому що Y_PRED передбачена заробітна плата спостереження за набором тесту:

– #plot for the TRAIN;

– plt.scatter(X_train, y_train, color='red') # plotting the observation line;

– plt.plot(X_train, regressor.predict(X_train), color='blue') #plotting the regression line;

– plt.title(«Salary vs Experience (Training set)») #stating the title of the graph;

– plt.xlabel(«Years of experience») #adding the name of x-axis;

– plt.ylabel(«Salaries») #adding the name of y-axis;

– plt.show() #specifies end of graph.

Наведений вище код генерує графік для набору поїзда, наведений нижче:

– Vector 1: [1, -1];

– Vector 2: [4, 2, 3];

– Cross Product: [-3, -3, 6];

– #plot for the TEST;

– plt.scatter(X_test, y_test, color='red');

– plt.plot(X_train, regressor.predict(X_train), color='blue') #plotting the regression line;

– plt.title(«Salary vs Experience (Testing set)»);

– plt.xlabel(«Years of experience»);

– plt.ylabel(«Salaries»);

– plt.show().

Наведений вище фрагмент коду генерує графік, як показано нижче:

– Vector 1: [[1, 2, 3]; [4, 5, 6]];

– Vector 2: [[4, 5, 6]; [1, 2, 3]];

– Cross Product: [-3, -3, 6]; [[-3, 6, -3]; [3, -6, 3]].

Практична робота № 6

Кластерний аналіз

Мета роботи – вивчення можливостей будівництва моделі кластеризації на Python.

Завдання до виконання лабораторної роботи – вивчити основи будівництва моделі кластеризації за допомогою бібліотеки **Pandas** та **Sklearn.**

Ми хочемо створити природні угруповання для набору об’єктів даних, які не можуть бути зазначені в самих даних. Наш аналіз використовуватиме дані виверження Old Faithful, знаменитого гейзера в Єллоустонському парку. Барні Гован знайшod ці дані у репозиторії: URL: https://github.com/barneygovan/from-data-with-love

Він містить лише два атрибути: час очікування між виверженням (хвилини) та тривалість виверження (хвилини). За допомогою двох атрибутів можна легко створити просту модель кластера k-середніх.

Кластерна модель k-means

Модель кластера K-середніх працює так:

1. Почніть із набору з k випадково вибраних центроїдів (передбачуваних центрів k кластерів).

2. Визначте, яка точка спостереження знаходиться в якомусь кластері відповідно до найближчого центроїду (використовуючи квадрат евклідової відстані: Σpj = 1 (xij-xi'j) 2, де p – розмірність).

3. Перерахуйте центроїд кожного кластера, мінімізуючи квадрат евклідової відстані від кожного спостереження у кластері.

4. Повторюйте 2 і 3, доки елементи кластера (отже, положення центру мас) не перестануть змінюватися.

5. Якщо це все ще збиває з пантелику, відвідайте Jigsaw Academy: URL: https://youtu.be/aiJ8II94qck

Тепер давайте продовжимо використовувати цю техніку до нашого набору даних Old Faithful.

Перший крок: дослідницький аналіз даних

Вам необхідно встановити кілька модулів, зокрема один під назвою **Scikit Learn**. Новий модуль – це набір інструментів для машинного навчання та інтелектуального аналізу даних на Python. Cluster – це модуль **Scikit Learn**, він використовує алгоритм кластеризації для імпорту функцій, тому імпортуйте його зі **Scikit Learn**.

Спочатку давайте імпортуємо всі необхідні модулі до нашого iPython Notebook і зробимо деякий дослідницький аналіз даних: URL: https://en.wikipedia.org/wiki/Exploratory_data_analysis

In [18]:

– import pandas as pd;

– import numpy as np;

– import matplotlib;

– import matplotlib.pyplot as plt;

– import sklearn;

– from sklearn import cluster;

– % matplotlib inline;

– faithful = pd.read_csv('/Users/michaelrundell/Desktop/faithful.csv');

– faithful.head().

Out повністю [18]:

	прорив	чекаючий
0	3.600	79
1	1.800	54
2	3.333	74
3	2.283	62
4	4.533	85

Прочитайте старий вірний csv та імпортуйте всі необхідні значення. Необхідно прочитати csv із локального каталогу, який виявиться робочим столом комп’ютера і відобразить перші п’ять записів даних. В цьому наборі даних немає стовпців або стовпців з NaN, тому ми можемо пропустити частину очищення даних у цьому прикладі. Погляньмо на базову діаграму розкиду даних.

In [19]:

– faithful.columns = ['eruptions', 'waiting'];

– plt.scatter(faithful.eruptions, faithful.waiting);

– plt.title('Old Faithful Data Scatterplot');

– plt.xlabel('Length of eruption (minutes)');

– plt.ylabel('Time between eruptions (minutes)').

Out[19]: <matplotlib.text.Text at 0x12a29bba8>:

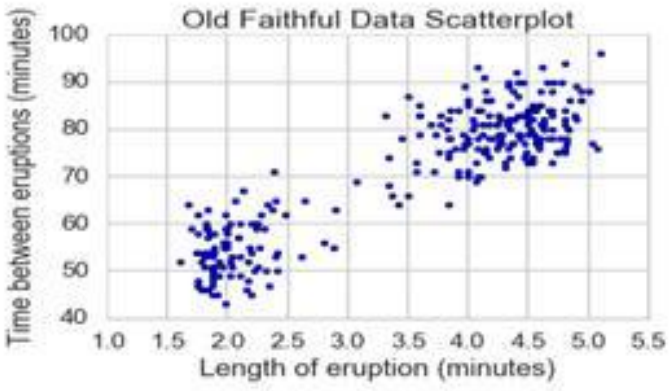

Рисунок 6.1 – Базова діаграма розкиду даних

Перейменуйте стовпці та використайте Matplotlib, щоб створити простий графік розсіювання.

Невеликі нотатки про процес: необхідно перейменувати стовпці так, щоб вони не відрізнялися неозброєним оком, але у стовпці «чекання» має бути додатковий простір перед словом. Щоб уникнути плутанини з подальшим

аналізом, потрібно змінити його і переконатися, що ми нічого не забудемо і не зробимо помилок у дорозі.

Другий крок: створення моделі кластера

Те, що ми бачимо, є діаграмою розсіювання з двома легко очевидними кластерами, але набір даних не відзначає будь-які спостереження як такі, що належать до будь-якої групи. Наступні кілька кроків охоплять процес візуального розрізнення двох груп. У наведеному нижче коді створено кілька важливих змінних та змінив формат даних.

In [20]:

– faith = np.array(faithful);

– k = 2;

– kmeans = cluster.KMeans(n_clusters=k);

– kmeans.fit(faith);

– labels=kmeans.labels;

– centroids=kmeans.cluster_centers.

Форматування та створення функцій:

1. Прочитайте достовірний кадр даних як масив NumPy, щоб Sci-kit міг читати дані.

2. Виберіть K = 2 як кількість кластерів, тому що ми намагаємося створити дві чіткі групи.

3. Змінна «kmeans» визначається виведенням виклику кластерного модуля у Sci-kit. Ми взяли K кластерів та підігнали дані до масиву «вірність».

Тепер, коли ми встановили змінні, які використовуються для створення моделі кластера, створимо візуалізацію. Наступний код малює діаграму розсіювання за кольором кластера та дає остаточне розташування центроїду. Опис конкретного рядка коду можна знайти нижче.

In [21]: for i in range(k):

– # select only data observations with cluster label == i;

– ds = faith[np.where(labels==i)];

– # plot the data observations;

– plt.plot(ds[:,0],ds[:,1],'o', markersize=7);

– # plot the centroids;

– lines = plt.plot(centroids[i,0],centroids[i,1],'kx');

– # make the centroid x's bigger;

– plt.setp(lines,ms=15.0);

– plt.setp(lines,mew=4.0);

– plt.show():

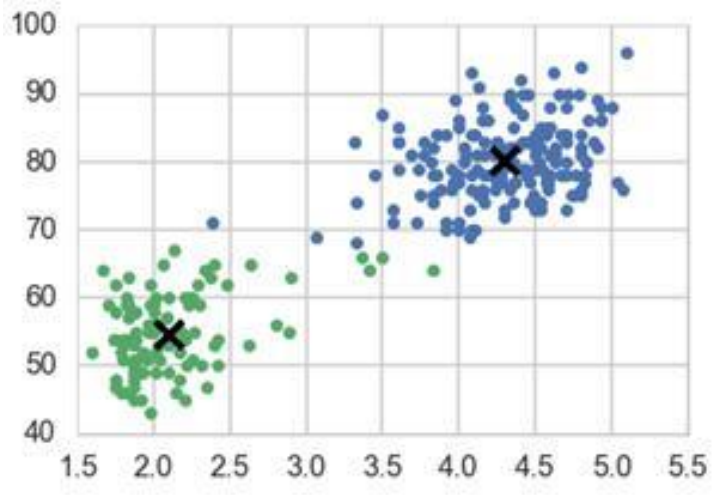

Рисунок 6.2 – Центри кластерів

Створіть візуалізацію кластерної моделі. Швидко розділіть наведений вище код:

1. Вся робота з групування даних у дві групи виконується у попередньому фрагменті коду, для цього використовуємо команду kmeans.fit (віра). Ця частина коду створює діаграму, яка це показує.

2. Змінна ds – це просто вихідні дані, але переформатовані, щоб увімкнути нову колірну мітку на основі кількості груп – цілого числа в k.

3. Plt.plot викликає дані x, дані y, форму об’єкта та розмір кола.

4. Решта коду відображає кінцевий центроїд процесу кластеризації k-середніх і керує розміром та товщиною мітки центроїду.

Ось вона – проста кластерна модель. Цей код підходить для кластерів, що містять різні числа, але і для цієї проблеми має сенс утримувати лише два кластери. Тепер, коли ми маємо ці кластери, які здаються добре визначеними, ми можемо зробити висновок про значення цих двох кластерів. Це зелене скупчення, що здебільшого складається з виверженнь вулканів. Короткий час очікування між виверженнями вулканів можна визначити як «слабкий чи швидкий вогонь», а блакитне скупчення Марса можна назвати «виверженням вогневої потужності».

Варто зазначити, що цей метод застосовується не до всіх наборів даних: URL: http://varianceexplained.org/r/kmeans-free-lunch/

Це означає, що кластеризація – це «не безоплатний обід». Якщо ваші дані мають нерівномірну можливість кластеризації, припущення K-середніх не будуть виконані (вони не мають приблизно однакової кількості спостережень у кожному кластері) або будуть мати несферичні кластери.

СПИСОК РЕКОМЕНДОВАНИХ ДЖЕРЕЛ

1. Болюбаш Н. М. Інтелектуальний аналіз даних : навч. посіб. / Н. М. Болюбаш. – Миколаїв : Вид-во ЧНУ ім. Петра Могили, 2023. – 320 с.

2. Інтелектуальний аналіз даних Data Mining [Електрон. ресурс] : навч.-метод. посіб. – Електрон. текст. дані. – Кропивницький : ФОП Піскова М. А., 2022. – 112 с. – Режим доступу: https://dspace.cusu.edu.ua/server/api/core/bitstreams/f8bbe456-1b5e-47b6-a80f-0a20b5d17264/content, вільний (дата звернення: 05.04.2025). – Назва з екрана.

3. Методичні вказівки до виконання контрольних робіт з дисципліни «Інтелектуальний аналіз даних» для студентів заочної форми навчання спеціальності 122 – «Комп'ютерні науки» [Електрон. ресурс] / В. І. Месюра, Я. В. Іванчук, О. К. Колесницький. – Електрон. текст. дані. – Вінниця : ВНТУ, 2021. – 42 с. – Режим доступу: https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/34293 /90097.pdf, вільний (дата звернення: 05.04.2025). – Назва з екрана.

4. Підручник з інтелектуального аналізу даних: що таке інтелектуальний аналіз даних? [Електрон ресурс] /Девід Картер. – Електрон. текст. дані. – Режим доступу : https://www.guru99.com/uk/data-mining-tutorial.html, вільний (дата звернення: 05.04.2025). – Назва з екрана.

5. Інтелектуальний аналіз даних [Електрон. ресурс]. – Електрон. текст. дані. – Режим доступу : https://science.lpnu.ua/sites/default/files/journal-paper/2017/nov/6688/1049-58.pdf, вільний (дата звернення: 05.04.2025). – Назва з екрана.

6. Інтелектуальний аналіз даних [Електрон. ресурс] : конспект лекцій / А. О. Дашкевич ; Нац. техн. ун-т «Харків. політехн. ін-т». – Електрон. текст. дані. – Харків : НТУ «ХПІ», 2024. – 96 с. – Режим доступу: https://repository.kpi.kharkov.ua/handle/KhPI-Press/85503, вільний (дата звернення: 03.03.2025). – Назва з екрана.

7. Бредіхін В. М. Фокус на штучний інтелект для прогнозування відтоку клієнтів з сайтів on-line освіти / В. М. Бредіхін, Т. С. Сенчук, К. С. Стужук // Комунальне господарство міст. – 2021. – Т. 6. – Вип. 166. – С. 3–7. – DOI: https://khg.kname.edu.ua/index.php/khg/article/view/5858/5776.

Електронне навчальне видання

Методичні рекомендації
до проведення практичних занять та організації самостійної роботи
з навчальної дисципліни

«ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ»

(для здобувачів першого (бакалаврського) рівня вищої освіти денної форми навчання зі спеціальностей 122, F3 – Комп'ютерні науки, 126, F6 – Інформаційні системи та технології)

Укладачі: **БРЕДІХІН** Володимир Михайлович,
ЛЕВІКОВ Юрій Володимирович

Відповідальний за випуск *М. В. Булаєнко*
Редактор *Б. О. Хільська*
Комп'ютерне верстання *В. М. Бредіхін*

План 2022 , поз. 239М

Підп. до друку 11.11.2025. Формат 60 × 84/16
Ум. друк. арк. 2,7.

Видавець і виготовлювач:
Харківський національний університет
міського господарства імені О. М. Бекетова,
вул. Чорноглазівська, 17, Харків, 61002
Електронна адреса: office@kname.edu.ua
Свідоцтво суб'єкта видавничої справи:
ДК № 8386 від 14.07.2025