УДК 004.62

АВТОМАТИЗАЦІЯ ОБРОБКИ ДАНИХ З ВИКОРИСТАННЯМ МОВИ ПРОГРАМУВАННЯ PYTHON

Б.М. Козловський[1], *С.В. Коваленко*[2]

[1] *магістрант кафедри САІТ, НТУ «ХПІ», Харків, Україна*
[2] *професор кафедри САІТ, кандидат техн. наук, НТУ «ХПІ», Харків, Україна*
Bohdan.Kozlovskyi@cs.khpi.edu.ua

В умовах стрімкого розвитку цифрових технологій та експоненціального зростання обсягів даних, автоматизація їх обробки стає критичним фактором успіху для бізнесу та наукових досліджень. За прогнозами компанії International Data Corporation до 2025 року глобальний обсяг даних досягне 175 зеттабайт, що робить ручну обробку практично неможливою та економічно недоцільною. Обробка великих даних є ключовим викликом для сучасних інформаційних систем, оскільки традиційні методи, що включають ручне очищення та підготовку даних, не справляються з великими обсягами інформації та потребують значних ресурсів [1].

Метою цієї роботи є дослідження можливостей використання автоматизованих рішень на основі мови програмування Python та необхідність розглянути основні етапи обробки великих обсягів даних, які включають в себе збір, очищення, нормалізацію та підготовку інформації для подальшого аналізу.

Мова програмування Python завдяки своїй простоті та широкому інструментарію стає одним із найбільш зручних рішень для створення ефективних, масштабованих систем обробки інформації. Успішна обробка великих даних значною мірою забезпечується за допомогою популярних бібліотек Python, таких як NumPy, Pandas, Dask та PySpark, кожна з яких має свої переваги та недоліки для розв’язків необхідних задач [2]. Важливу роль у екосистемі Python для обробки даних відіграє бібліотека NumPy, яка забезпечує потужну підтримку багатовимірних масивів та математичних операцій. NumPy є фундаментом для багатьох інших інструментів аналізу даних, таких як Pandas, забезпечуючи високу ефективність роботи з векторами та матрицями.

```
In [4]: # Визначаємо діапазон для x1
        x1_vals = np.linspace(-10, 20, 400)

        # Обчислюємо x2 для кожного рівняння
        x2_1 = -8.3/8.2 * x1_vals + 134/8.2
        x2_2 = 10.2/15.2 * x1_vals + 102/15.2
        x2_3 = 14.5/7.3 * x1_vals - 141/7.3

        # Функція для знаходження перетину двох прямих
        def find_intersection(A, B):
            return np.linalg.solve(A, B)

        # Обчислюємо точки перетину
        A1 = np.array([[8.3, 8.2], [-10.2, 15.2]])
        B1 = np.array([134, 102])
        intersect_12 = find_intersection(A1, B1)
```

Рис. 1 – Приклад використання бібліотеки NumPy

Бібліотека Pandas є зручним інструментом для роботи зі структурованими даними, такими як таблиці та бази даних. Вона забезпечує широкий набір функцій для обробки, фільтрації та аналізу даних, що робить її ефективною для задач невеликих і середніх масштабів. Однак, із зростанням обсягів даних виникає потреба в більш потужних інструментах для паралельної обробки та оптимізації продуктивності. Бібліотека Dask розширює можливості Pandas, дозволяючи виконувати обробку великих масивів даних шляхом розподілу завдань на кілька ядер процесора. Це забезпечує гнучкість та масштабованість для роботи з більшими наборами даних, при цьому Dask залишається зручною та інтегрованою бібліотекою із екосистемою Python.

Ще один потужний інструмент для обробки великих даних – PySpark, частина платформи Apache Spark. PySpark є оптимальним рішенням для задач, що потребують обробки великих масивів інформації в кластерному середовищі. Вона дозволяє паралелізувати обчислення та виконувати розподілену обробку даних, що є ефективним підходом для важких операцій у реальному часі та забезпечує високу швидкість обробки. PySpark також підтримує інтеграцію з хмарними платформами, такими як Amazon Web Services та Google Cloud Platform, що дозволяє масштабувати обчислювальні ресурси в залежності від потреб обробки.

Важливим аспектом автоматизації обробки даних також є їх візуалізація [3]. Python пропонує потужні бібліотеки для створення візуалізацій, такі як Matplotlib, Seaborn та Plotly. Ці інструменти дозволяють створювати інтерактивні графіки, діаграми та дашборди, що значно полегшує аналіз та презентацію результатів обробки даних.

```
In [7]: # Загальний графік для всіх трьох ліній
        plt.figure(figsize=(8, 6), dpi=100)
        plt.plot(x1_vals, x2_1, label=r'$f[0]$', color="#4357AD", linestyle="dashed")
        plt.plot(x1_vals, x2_2, label=r'$f[1]$', color="#48A9A6", linestyle="dotted")
        plt.plot(x1_vals, x2_3, label=r'$f[2]$', color="#AF125A", linestyle="dashdot")
        plt.plot(*intersect_12, 'ro')
        plt.plot(*intersect_13, 'ro')
        plt.plot(*intersect_23, 'ro')
        # Підписи для кожної лінії
        plt.annotate(r'$8.3x_1 + 8.2x_2 = 134$', xy=(-5, 14), xytext=(10, 20),
                     textcoords='offset points', fontsize=10, color='#4357AD', rotation=-18)
        plt.annotate(r'$-10.2x_1 + 15.2x_2 = 102$', xy=(-5, -4), xytext=(10, 20),
                     textcoords='offset points', fontsize=10, color='#48A9A6', rotation=12)
        plt.annotate(r'$14.5x_1 - 7.3x_2 = 141$', xy=(-5, -26), xytext=(10, 20),
                     textcoords='offset points', fontsize=10, color='#AF125A', rotation=30)
        # Підписи точок перетину
        plt.annotate(f'Точка перетину f[0] та f[1]', xy=intersect_12, xytext=(-60, 10),
                     textcoords='offset points', fontsize=9, color='red')
        plt.annotate(f'Точка перетину f[0] та f[2]', xy=intersect_13, xytext=(-60, -15),
                     textcoords='offset points', fontsize=9, color='red')
        plt.annotate(f'Точка перетину f[1] та f[2]', xy=intersect_23, xytext=(-60, -20),
                     textcoords='offset points', fontsize=9, color='red')
        # Заповнення трикутника
        plt.fill([intersect_12[0], intersect_13[0], intersect_23[0]], [intersect_12[1], intersect_13[1], intersect_23[1]],
                 color='gray', alpha=0.3)
        plt.xlabel(r'$x_1$')
        plt.ylabel(r'$x_2$')
        plt.title('Перетин трьох ліній')
        plt.axhline(0, color='black', linewidth=1)
        plt.axvline(0, color='black', linewidth=1)
        plt.grid(color='gray', linestyle='--', linewidth=0.5)
        plt.legend()
        plt.show()
```

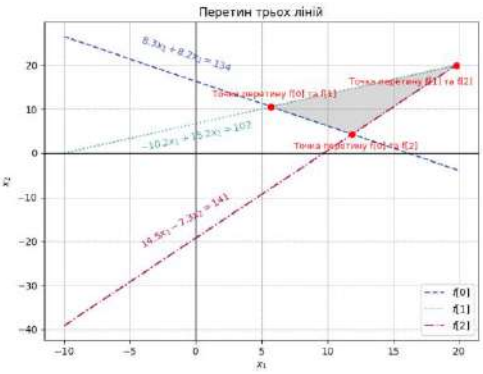

Рис. 2 – Приклад використання бібліотеки Matplotlib

Використання Python для автоматизації обробки великих даних відкриває широкі можливості для організацій, які працюють із великими обсягами інформації. Інтеграція з Pandas, Dask та PySpark дозволяє значно підвищити ефективність аналітики та знизити витрати на обробку даних за рахунок автоматизації та паралельного виконання завдань. Крім того, впровадження хмарних обчислень у поєднанні з Python додає гнучкості, дозволяючи компаніям адаптувати свої системи під потреби ринку та обробляти великі масиви даних швидше і ефективніше.

Список літератури:

1. *Kovalenko S. M., Kutsenko O. S., Kovalenko S. V., Kovalenko A. S.* Approach to the automatic creation of an annotated dataset for the detection, localization and classification of blood cells in an image. Radio Electronics, Computer Science, Control, (1). 2024. 128. https://doi.org/10.15588/1607-3274-2024-1-12.

2. *McKinney, W*. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. O'Reilly Media. 2017. p. 470.

3. *Коваленко, А. С.* Застосунок для автоматичного створення анотованого набору даних зображень / *А. С. Коваленко* // Сучасні інформаційні технології та системи в управлінні [Електронний ресурс] : Зб. матеріалів V Міжнар. наук.-практ. конф. молодих вчених, аспірантів і студентів – Київ: КНЕУ, 2024 – С. 273 – 275.