DOI 10.34132/mspc2025.01.06.16

Євген Дарнапук

ВИКОРИСТАННЯ МОЖЛИВОСТЕЙ БІБЛІОТЕК PANDAS, NUMPY ТА RUPTURE ДЛЯ ОБРОБКИ ТА АНАЛІЗУ МАСИВІВ ДАНИХ АМБУЛАТОРНОГО ДОБОВОГО МОНІТОРИНГУ ТИСКУ КРОВІ

Тези представляють застосування бібліотек Pandas, Numpy та Ruptures для ефективної обробки, аналізу й автоматичного виявлення структурних змін у масивах даних амбулаторного добового моніторингу артеріального тиску.

Ключові слова: *амбулаторний добовий моніторинг крові, python, numpy, data science, pandas, rupture.*

Амбулаторний добовий моніторинг тиску крові (англ. Ambulatory Blood Pressure Monitoring, *ABPM*) – це один із найефективніших неінвазивних методів тривалого контролю стану серцево-судинної системи [1]. В першу чергу, він є найкращим методом для діагностування гіпертензії виконуючи вимірювання протягом типової добової активності досліджуваної особи. Такий моніторинг дозволяє реєструвати систолічний (*SYS*), діастолічний (*DIA*) тиск і частоту серцевих скорочень (*HR*) з інтервалом у кілька хвилин протягом доби або більш тривалого періоду. Результатом є масив неінвазивних, але складних для ручної інтерпретації даних, що вимагає використання сучасних математичних методів аналізу і цифрової обробки сигналів.

Ефективність аналізу великих масивів таких даних залежить від здатності сучасних методів виявляти приховані закономірності та фазові переходи у фізіологічних процесах, а також знижувати вплив шуму. Впровадження інструментів Python – pandas (для маніпуляцій із даними), numpy (для масивних розрахунків), ruptures (для виявлення змін у структурі сигналів), pywt (вейвлет-фільтрація) – дозволяє здійснювати комплексний автоматизований аналіз та отримувати нові знання із великих масивів *ABPM*.

У дослідженні було використано відкритий та деперсоніфікований датасет амбулаторного добового моніторингу крові, що містить результати добового моніторингу із пристрою Oscar 2 Ambulatory Blood Pressure Monitor. Дані містили час вимірювання, систолічний, діастолічний тиск та частоту пульсу. Для зручності аналізу їх було приведено до табличного вигляду формату Excel. Попередню обробку виконано використовуючи бібліотеку pandas (рис. 1) [2].

```
df = pd.read_excel("dataset_with_headers.xlsx")
df["Time"] = pd.to_datetime(df["Time"].astype(str), errors="coerce")
df_clean = df.dropna(subset=["Time", "SYS", "DIA", "HR"]).reset_index(drop=True)
```

Рис. 1. Попередня обробка даних амбулаторного моніторингу тиску крові

Одним із перших етапів аналізу масивів даних амбулаторного добового моніторингу тиску крові є візуалізація часових рядів основних фізіологічних показників: систолічного тиску (*SYS*), діастолічного тиску (*DIA*) та частоти серцевих скорочень (*HR*). Цей крок дозволяє досліднику отримати цілісне уявлення про динаміку показників протягом доби, виявити загальні тренди, коливання, екстремальні значення та аномалії, які можуть свідчити про зміни фізіологічного стану пацієнта. Для детального аналізу будується комбінований графік часових рядів всіх трьох показників. Блок коду з застосуванням бібліотеки matplotlib наведено на рис. 2. На рис. 3 показано графік динаміки систолічного, діастолічного тиску та серцевого ритму протягом доби.

```
# === Побудова графіка з реальною віссю часу ===
plt.figure(figsize=(14, 6))

# Побудова трьох ліній
plt.plot(df_clean["Time"], df_clean["SYS"], label="SYS (систолічний)", marker='o')
plt.plot(df_clean["Time"], df_clean["DIA"], label="DIA (діастолічний)", marker='s')
plt.plot(df_clean["Time"], df_clean["HR"], label="HR (пульс)", marker='^')

# Оформлення осей
plt.xlabel("Час вимірювання")
plt.ylabel("Значення (мм рт. ст. / уд. за хв.)")
plt.title("Динаміка систолічного, діастолічного тиску та пульсу протягом доби")
plt.xticks(rotation=45)
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

Рис. 2. Побудова комбінованого графіку часових рядів трьох показників

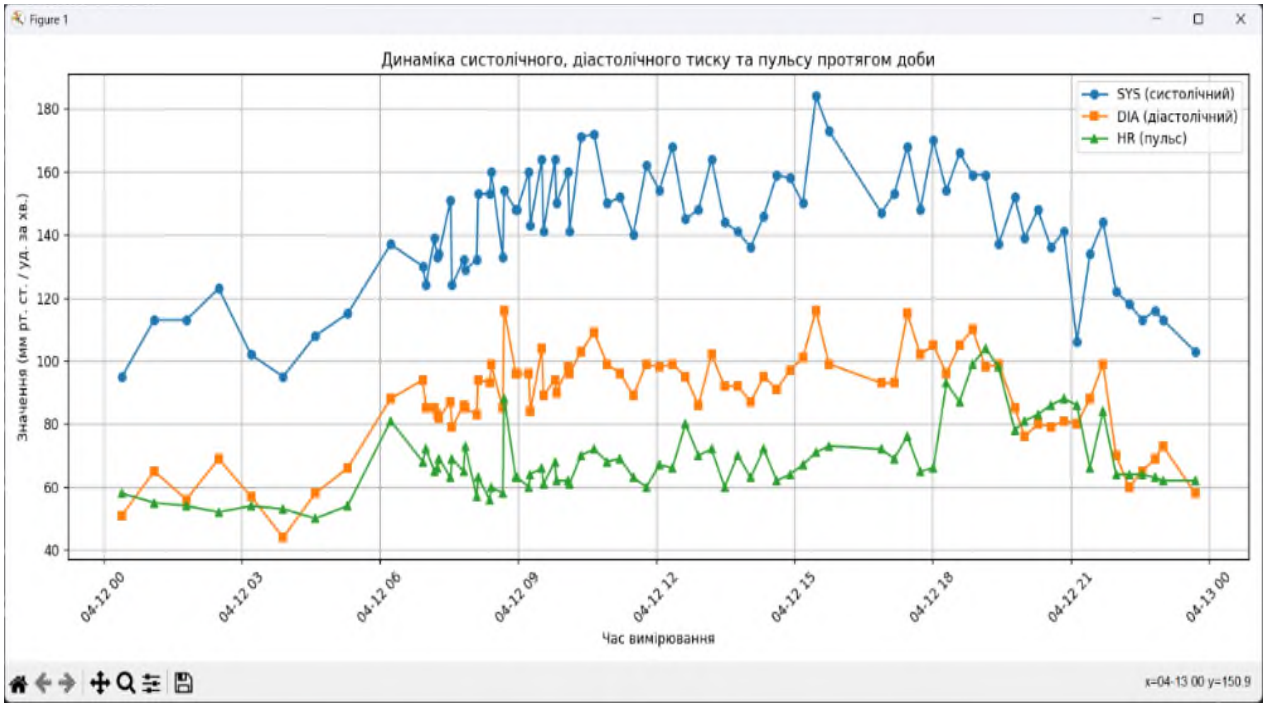


Рис. 3. Динаміка систолічного, діастолічного тиску та пульсу протягом добового моніторингу пацієнта

Для оцінки складності й структурної організації часових рядів фізіологічних сигналів використовуються:

1) коефіцієнт повторення (recurrence ratio, *RR*) – показник, який відображає повторюваність патернів сигналу у двовимірній гістограмі переходів між послідовними значеннями;

2) ентропія Шеннона (Shannon entropy, *H*) – міра невпорядкованості або хаотичності сигналу, яка розраховується на основі ймовірностей переходів між станами.

Для визначення коефіцієнта повторення використовується формула (1):

$$RR = \frac{B_{nonempty}}{B^2}, \quad (1)$$

де B – кількість бінів та $B_{nonempty}$ – кількість бінів з непорожнім вмістом.

Для визначення ентропії Шеннона використовується формула (2):

$$H = -\sum_i \; p_i \times (p_i) \; , \quad (2)$$

де p_i – нормалізовані значення з двовимірної гістограми, що не дорівнюють нулю. Приклад реалізації методів визначення рекурентного коефіцієнту та ентропії Шеннона з використанням бібліотеки numpy наведено на рис. 4 та рис. 5.

```python
def shannon_entropy(signal, bins=9):
    x_n = signal[:-1]
    x_np1 = signal[1:]
    hist2d, _, _ = np.histogram2d(x_n, x_np1, bins=bins)
    p = hist2d.flatten()
    p = p[p > 0] / np.sum(p)
    return -np.sum(p * np.log2(p))

def recurrence_ratio(signal, bins=9):
    x_n = signal[:-1]
    x_np1 = signal[1:]
    hist2d, _, _ = np.histogram2d(x_n, x_np1, bins=bins)
    return np.sum(hist2d > 0) / (bins * bins)
```

Рис. 4. Визначення показників ентропії Шеннона та коефіцієнта повторюваності з застосуванням бібліотеки *numpy*

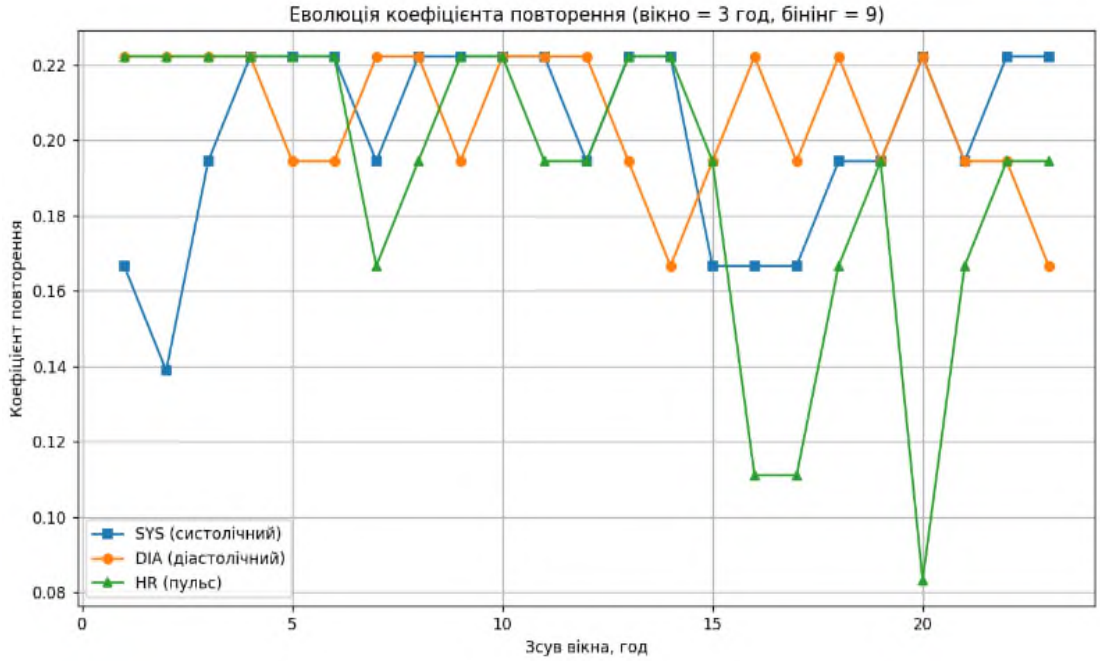
Для аналізу еволюції *RR* та *H* використовується ковзаюче вікно фіксованого розміру (наприклад, 3 години) із заданим кроком (наприклад, 1 година):

```python
# Параметри вікна та бінінгу
sampling_interval_min = np.median(np.diff(df_clean["Time"].astype(np.int64)) / 60e9)
samples_per_hour = int(60 / sampling_interval_min)
window_hours = 3
step_hours = 1
bins = 9
window_size = samples_per_hour * window_hours
step_size = samples_per_hour * step_hours

# Обчислення
rr_hr, entropy_hr, window_shift_hours = [], [], []
for start in range(0, len(df_clean) - window_size, step_size):
    end = start + window_size
    segment = df_clean["HR"].iloc[start:end].to_numpy()
    rr = recurrence_ratio(segment, bins=bins)
    h = shannon_entropy(segment, bins=bins)
    rr_hr.append(rr)
    entropy_hr.append(h)
    window_shift_hours.append(len(window_shift_hours) + 1)
```

Рис. 5. Розрахунок динаміки показників у ковзаючих вікнах

Використовуючи бібліотеку matplotlib, побудуємо графіки еволюції коефіцієнта повторення та еволюції ентропії Шеннона, що наведено на рис. 6.



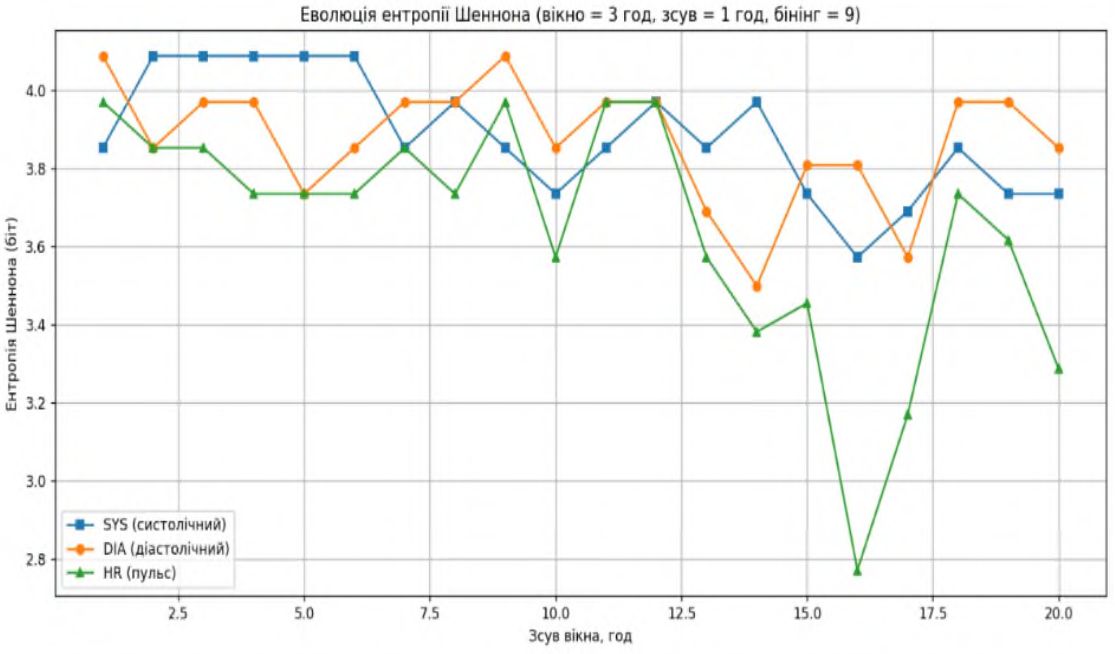


Рис. 6. Графіки еволюції коефіцієнта повторення та еволюції ентропії Шеннона

Для кожного показника (RR та ентропії Шеннона) обчислюється перша числова похідна:

$$\frac{\partial RR}{\partial t} \approx RR_{i+1} - RR_i, \frac{\partial H}{\partial t} \approx H_{i+1} - H_i. \quad (3)$$

Фазовий перехід вважається виявленим, якщо відповідає наступній вимозі (4):

$$\left|\frac{\partial RR}{\partial t}\right| > \mu_{RR} + \sigma_{rr}, \text{або } \left|\frac{\partial H}{\partial t}\right| > \mu_H + \sigma_H, \quad (4)$$

де μ – середнє значення похідної, σ – стандартне відхилення. Реалізація обчислень похідних наведено на рис. 7. Графіки похідних наведені на рис. 8.

```
# Похідні
rr_hr = np.array(rr_hr)
entropy_hr = np.array(entropy_hr)
grad_rr = np.gradient(rr_hr)
grad_H = np.gradient(entropy_hr)
```

Рис. 7. Визначення перших похідних до коефіцієнтів повторення та ентропії Шеннона

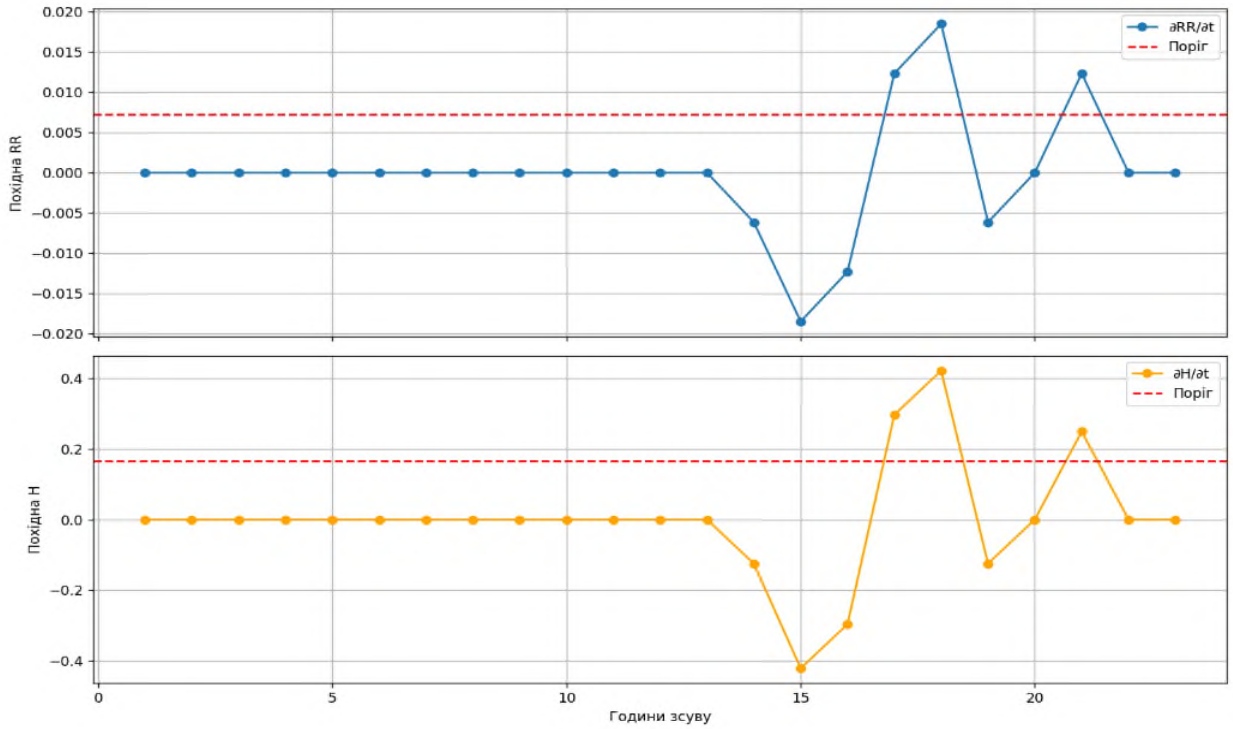


Рис. 8. Графік похідних коефіцієнта повторення (згори) та ентропії Шеннона (знизу)

Однією з ключових задач у аналізі часових рядів медичних сигналів є автоматичне виявлення моментів, коли структура сигналу змінюється – так званих фазових переходів (change points). Це можуть бути раптові зсуви середнього значення, дисперсії, частоти або зміна закономірностей у динаміці параметра. Для цієї задачі у Python-екосистемі широко використовується спеціалізована бібліотека ruptures.

Бібліотека ruptures реалізує сучасні алгоритми сегментації часових рядів (change point detection), серед яких:

- Binary Segmentation (Binseg) – послідовне бінарне поділення сигналу для пошуку точок змін;
- Pelt, Window-based, Bottom-Up та інші алгоритми.

У роботі було обрано алгоритм Binseg, як простий та швидкий для задачі з невеликим числом очікуваних змін. Реалізація пошуку фазових переходів наведена на рис. 9 [3].

```
# ruptures: фазові переходи
X = np.column_stack((rr_hr, entropy_hr))
algo = rpt.Binseg(model="l2").fit(X)
breakpoints = algo.predict(n_bkps=3)
predicted_hours = [window_shift_hours[i - 1] for i in breakpoints[:-1]]
```

Рис. 9. Пошук фазових переходів за допомогою засобів бібліотеки *rupture*

Еталонні значення фазових переходів є [9; 16; 20] і вони визначені вручну. Розраховані значення переходів за допомогою числових похідних є наступними [10; 16; 20]. Відповідно, точність визначення фазових переходів наближається до 100 % при аналізі цього датасету.

References

1. McGrath B. P. Ambulatory blood pressure monitoring. *Medical Journal of Australia*. 2002. Vol. 176, no. 12. P. 588–592. DOI: 10.5694/j.1326-5377.2002.tb04590.x.
2. Enache M. C. Data Analysis with Pandas. *Annals of Dunarea de Jos University of Galati. Fascicle I. Economics and Applied Informatics*. 2019. Vol. 25, no. 2. P. 69–74. DOI: 10.35219/eai1584040933.
3. Binary segmentation - ruptures. *GitHub Pages*. Retrieved from: https://centre-borelli.github.io/ruptures-docs/user-guide/detection/binseg/ (Last accessed: 17.05.2025).

UTILISING THE CAPABILITIES OF THE PANDAS, NUMPY, AND RUPTURES LIBRARIES FOR THE PROCESSING AND ANALYSIS OF AMBULATORY BLOOD PRESSURE MONITORING DATA ARRAYS

The theses present the application of the Pandas, Numpy, and Ruptures libraries for efficient processing, analysis, and automatic detection of structural changes in ambulatory blood pressure monitoring data arrays.

Key words: *ambulatory blood pressure monitoring, python, numpy, data science, pandas, rupture.*

Відомості про автора / Information about the Author

Євген Дарнапук, старший викладач кафедри комп'ютерної інженерії Чорноморського національного університету імені Петра Могили, м. Миколаїв, Україна. E-mail: yevhen.darnapuk@chmnu.edu.ua, orcid: https://orcid.org/0000-0002-7099-5344.

Yevhen Darnapuk, Senior Lecturer of the Computer Engineering Department at Petro Mohyla Black Sea National University, Mykolaiv, Ukraine. E-mail: yevhen.darnapuk@chmnu.edu.ua, orcid: https://orcid.org/0000-0002-7099-5344

Стаття отримана редакцією 20.05.2025.