Вінницький національний технічний університет
Факультет інформаційних електронних систем
Кафедра інфокомунікаційних систем і технологій

Бакалаврська дипломна робота на тему:

ПРОГРАМНІ ЗАСОБИ PYTHON МОДЕЛЮВАННЯ ТА СИСТЕМНОГО АНАЛІЗУ КІБЕРЗАХИСТУ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ СИСТЕМ

Виконав: студент 4-го курсу, групи ПЗТ-22мс
спеціальності 172 – Телекомунікації та радіотехніка
_____ Тригубенко Я.Ю.

Керівник: к.т.н., доц. каф. ІКСТ
_____ Бортник С. Г.
«12» 06 2024 р.

Рецензент: д.т.н., проф., професор каф. ІРТС
_____ Семенов А.О
«12» 06 2024 р.

Допущено до захисту
Завідувач кафедри ІКСТ
_____ д.т.н., проф. Кичак В.М.
«12» 06 2024 р.

Вінниця ВНТУ - 2024 рік

Вінницький національний технічний університет
Факультет інформаційних електронних систем
Кафедра інфокомунікаційних систем та технологій
Рівень вищої освіти перший (бакалаврський)
Галузь знань 17 – Електроніка та телекомунікації
(шифр і назва)
Спеціальність 172 – Телекомунікації та радіотехніка
(шифр і назва)
Освітня програма – Програмне забезпечення телекомунікаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри ІКСТ
д.т.н., професор В.М. Кичак
«06» 05 2024 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Тригубенку Ярославу Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи: «Програмні засоби Python моделювання та системного аналізу кіберзахисту інформаційно-комунікаційних систем»
керівник роботи Бортник Сергій Геннадійович, к.т.н, доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом ВНТУ від "11" березня 2024 року № 80
2. Строк подання студентом роботи 07 червня 2024 року

3. Вихідні дані до роботи: Тип стандарту зв'язку – IEEE 802.11; Сумарна швидкість інформаційних потоків - Ethernet 100 Мбіт/с згідно технічного завдання; Імовірність помилки в радіотракті - 10^{-6}; Тип програмного забезпечення - віртуальні машини; Метод доступу до програмного середовища – дистанційний;

4. Зміст розрахунково-пояснювальної записки роботи (перелік питань, які потрібно розробити): вступ, часові ряди – інструмент моделювання динамічних систем, аналіз часових рядів в задачах моніторингу шифрованого мережевого трафіку, програмні засоби python та модулів розширення для роботи з часом та датами, засоби python для використання в задачах описової статистики, охорона праці.

5. Перелік графічного матеріалу: Приклади аномалій в даних моніторингу мережевого трафіку; Прояви аномалій в часових рядах; Узагальнюючий фреймворк часових рядів; Алгоритм CPD автокореляції; Основні елементи описової статистики, що використовуються для аналізу варіативногсті даних.

6. Консультанти розділів роботи

Розділ	ище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Спеціальна частина	Бортник С.Г., доцент кафедри ІКСТ	06.05.2024	12.06.2024
Охорона праці	Дембіцька С.В. професор кафедри БЖДПБ.	06.05.2024	12.06.2024

7. Дата видачі завдання 06 травня 2024 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Формування та затвердження теми та індивідуального завдання бакалаврської дипломної роботи (БДР)	06.05.2024р.	
2	Виконання спеціальної частини БДР. Перший рубіжний контроль виконання БДР	20.05.2024р.	
3	Виконання спеціальної частини БДР. Другий рубіжний контроль виконання БДР	31.05.2024р.	
4	Виконання розділу «Охорона праці»	04.06.2024р.	
5	Нормоконтроль БДР	05.06.2024р.	
6	Попередній захист БДР	06.06.2024р.	
7	Рецензування БДР	07.06.2024р.	
8	Захист БДР	12.06.2024р.	

Студент _____ Тригубенко Я.Ю.
(підпис)

Керівник роботи _____ Бортник С.Г.
(підпис)

АНОТАЦІЯ

УДК 621.396

Тригубенко Я.Ю. – бакалаврська дипломна робота студента спеціальності 172 – Телекомунікації та радіотехніка – Вінниця: ВНТУ 2024 р. 67 стор., 6 – рис., 5 – табл. 22 -джерел.

У бакалаврській дипломній роботі Проаналізовано сучасний стан використання часових рядів як інструменту моделювання динамічних систем.
Вивчено програмні засоби Python та модулі розширення для роботи з часом та датами, включаючи модулі time, datetime, бібліотеки numpy та pandas. Описано особливості використання цих інструментів для аналізу та обробки часових рядів. Розглянуто засоби Python для використання в задачах описової статистики, зокрема для обчислення основних характеристик розподілів, таких як середнє значення та медіана вибірки. Показано приклади обробки даних часового ряду, отриманого при моніторингу мережевого трафіку.
На кінцевому етапі роботи проводиться дослідження питань охорони праці.

Ключові слова: Python, мова програмування, бібліотеки, моделювання.

ANNOTATION

Trigubenko Y.Y.- bachelor thesis of a student of specialty 172 - Telecommunications and radio engineering - Vinnytsia: VNTU 2024. 67 pages, 6 - fig., 5 – table, 22 - sources.

The bachelor thesis analyzed the current state of using time series as a tool for modeling dynamic systems.
Learned Python tools and extension modules for working with time and dates, including the time, datetime, numpy, and pandas libraries. Features of using these tools for time series analysis and processing are described. Explores Python tools for use in descriptive statistics problems, particularly for computing basic characteristics of distributions, such as the sample mean and median.
At the final stage of the work, a study of occupational health and safety issues is conducted.

Keywords: Python, programming language, libraries, modeling.

ЗМІСТ

ВСТУП

Сучасний стан розвитку інформаційних технологій дозволяє ефективно використовувати часові ряди для моделювання динамічних систем. Часові ряди являють собою послідовність даних, відсортованих у хронологічному порядку, що дозволяє аналізувати та прогнозувати поведінку систем у часі. Це особливо актуально в задачах моніторингу шифрованого мережевого трафіку, де необхідно вчасно виявляти аномалії та забезпечувати безпеку даних.

Для роботи з часовими рядами в мові програмування Python існує багато інструментів і бібліотек, які значно спрощують обробку та аналіз даних. Наприклад, модулі time та datetime забезпечують базові операції з датами і часом, тоді як бібліотеки numpy та pandas пропонують більш розширені можливості для маніпуляцій з часовими рядами та їх аналізу. Використання цих інструментів дозволяє ефективно обробляти великі обсяги даних і отримувати необхідні статистичні характеристики.

Окрім цього, засоби Python широко використовуються для задач описової статистики, що є важливою частиною аналізу часових рядів. Основні характеристики розподілів, такі як середнє значення та медіана вибірки, допомагають зрозуміти структуру та поведінку даних. Зокрема, бібліотеки numpy та pandas забезпечують зручні функції для обчислення цих характеристик, що робить їх незамінними інструментами для дослідників.

У роботі також розглядаються приклади обробки даних часового ряду, отриманого при моніторингу мережевого трафіку. Ці приклади демонструють, як можна використовувати бібліотеки Python для аналізу реальних даних та виявлення важливих закономірностей. Зокрема, показано, як за допомогою функцій numpy та pandas можна аналізувати часові ряди і отримувати корисну інформацію для прийняття рішень.

Особливу увагу приділено питанням охорони праці під час виконання роботи. Технічні рішення щодо безпечного виконання роботи, гігієни праці та виробничої санітарії забезпечують безпечні умови праці для співробітників і мінімізують ризики під час виконання завдань.

Метою даної роботи є аналіз та обробка даних часових рядів для моніторингу шифрованого мережевого трафіку з використанням сучасних інструментів Python. Завданнями дослідження є аналіз стану галузі, порівняльний аналіз існуючих методів, розробка алгоритмів та структури програмного продукту, вибір та обґрунтування інструментів для реалізації, а також тестування та впровадження розробленого рішення.

Протягом розробки було використано методи аналізу часових рядів, побудови алгоритмів, розробки програмного забезпечення та оптимізації його роботи. Основні результати та ідеї роботи були представлені та обговорені на науковій конференції, що підтверджує їх актуальність та практичну цінність.

1. ЧАСОВІ РЯДИ – ІНСТРУМЕНТ МОДЕЛЮВАННЯ ДИНАМІЧНИХ СИСТЕМ

Часовий ряд у найширшому розумінні – це послідовність числових показників, упорядкованих у часі, що характеризують стан і зміни об'єкта, процесу або явища, які досліджуються в конкретний момент часу. Приклади часових рядів зустрічаються всюди: погодинна кількість переглядів веб-сторінок, добова температура повітря в місті, рівень води у водоймі, щомісячна кількість звернень до служби підтримки, квартальний прибуток компанії, річні демографічні дані, концентрація речовини під час хімічного процесу тощо. Кожен часовий ряд включає два основні елементи: час і відповідне значення показника або показників.

Час в часовому ряді може визначатися як конкретний момент. Наприклад, у технології IoT (Internet of Things) часові ряди використовуються для представлення даних з датчиків (тиск, температура, напруга тощо). У таких випадках використовується термін "мітка часу", а сам часовий ряд називають "моментним". В інших випадках часовий ряд може бути інтервальним, тобто дані в ньому представляють результат, накопичений за певний інтервал часу. Наприклад, щомісячні обсяги продукції підприємства або кількість відпрацьованих людино-днів. Моментний ряд можна легко перетворити на інтервальний шляхом об’єднання даних за певний проміжок часу. Це часто роблять для спрощення аналізу. Наприклад, інтернет-трафік, який складається з пакетів інформації, що надходять у довільні моменти часу, можна перетворити на інтервальний ряд, сумуючи обсяги всіх пакетів за певний проміжок часу (мілісекунду, секунду, годину).

У векторі u(t) кожне значення розглядається як випадкова змінна. Вимірювання в часовому ряді завжди (якщо не зазначено інше) впорядковані хронологічно. Ці вимірювання також називають значеннями, рівнями, спостереженнями або точками ряду. Більшість часових рядів складаються з даних, зібраних у рівновіддалені моменти часу (еквідистантні ряди) або інтервали однакової величини, наприклад, погодинні вимірювання температури або щоденні підрахунки відвідувань веб-сайтів. Однак часові ряди можуть бути і нерегулярними або спорадичними, як-от дані з мітками часу в журналі подій комп’ютерної системи або історія екстрених викликів.

Таким чином, часові ряди – це модель, яка допомагає отримати знання про стан і поведінку об'єкта. Дослідження часового ряду – це формалізована процедура, що спрямована на розуміння природи об'єкта, параметри якого формують часовий ряд, та вираження цього розуміння у вигляді моделі, здатної описати його поведінку. Різноманіття прикладних задач, що зводяться до аналізу часових рядів, вражає. Найбільш захоплюючим є те, що вивчивши методи аналізу та програмні інструменти, перед дослідником відкривається великий простір для застосування цих знань.

Задачі, що розглядаються при дослідженні часових рядів, можна поділити на дві групи залежно від часу: аналіз минулих даних і прогнозування майбутніх.

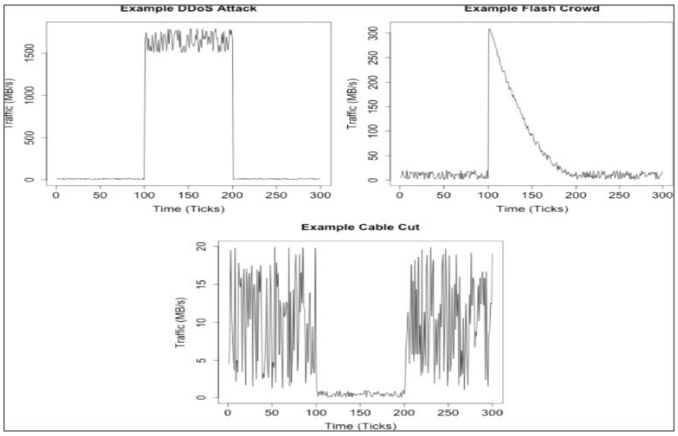

Рис. 1.1 – Приклади аномалій в даних моніторингу мережевого трафіку.

Перший тип задач – аналітика. Вона полягає у вивченні даних з минулого для пошуку відповідей на питання "що сталося?" і "чому це сталося?". Цей тип задач часто називають "виявленням аномалій", оскільки нас цікавлять моменти, коли об'єкт суттєво змінює свою поведінку. Виявлення аномалій є важливим завданням у багатьох прикладних областях, таких як виявлення мережевих атак, шахрайства (зокрема з кредитними картками), аномалій у медицині, деградації обладнання тощо. На рисунку 1.1 представлені зразки часових рядів, що відображають стан інтернет-мережі. Зміни параметрів можуть бути викликані як зловмисним втручанням, так і відмовою обладнання.

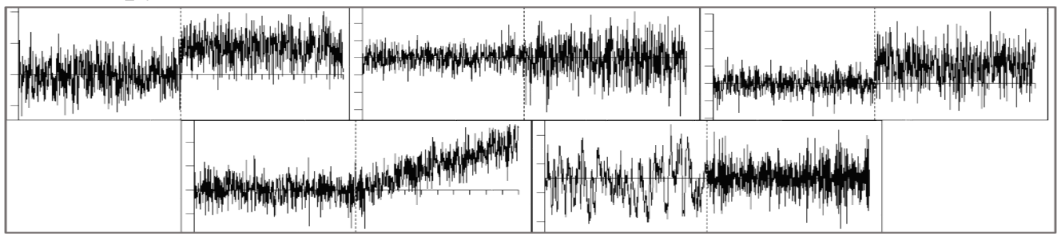

Рис. 1.2 – Деякі можливі прояви аномалій в часових рядах

Формальний аналіз часового ряду може виявити моменти, в яких відбуваються значні зміни параметрів, що, у свою чергу, свідчить про зміну поведінки досліджуваного об’єкта. Проте такі зміни не завжди легко виявити. На Рис. 1.2 наведено приклади, де визначити зміни та точно вказати момент їх початку складніше, ніж у попередніх випадках.

На Рис. 1.3 представлені ще декілька прикладів аномалій. Людина здатна

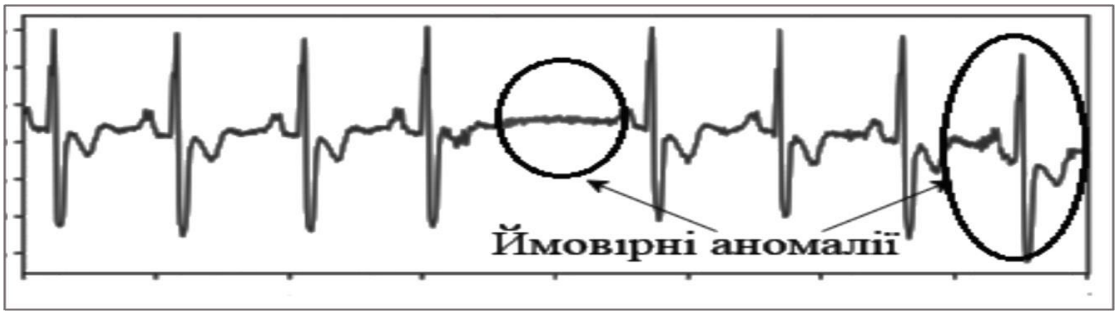

Рис. 1.3 – Деякі можливі прояви аномалій в часових рядах

Зазвичай візуальний аналіз графічного відображення часового ряду дозволяє виявити аномальні ситуації. Але автоматизація такого розпізнавання є складним завданням, особливо коли неможливо заздалегідь описати ні нормальну поведінку ряду, ні всі можливі аномалії.

Основна задача другого типу – це прогнозування майбутніх значень часового ряду на основі минулих даних. Це найвідоміше і найефектніше застосування часових рядів. Прогнозування охоплює різноманітні приклади: передбачення того, що станеться на фінансовому ринку завтра, обсягу продажів товарів на наступний тиждень, кількості поїздок Uber за найближчі три години, об’єму інтернет-трафіку у провайдера та кількості кібератак на сайт у наступний місяць, кількості захворювань на Covid у наступному тижні тощо.

Аналіз і прогнозування часових рядів тісно пов'язані. Як правило, результати одного типу дослідження використовуються для іншого. Обидва типи задач спираються на побудову моделі поведінки об'єкта. Існує багато різноманітних моделей часових рядів, які застосовуються як у наукових дослідженнях, так і для вирішення практичних задач. Серед них можна виділити прості моделі ковзаючого та експоненціального згладжування, моделі з урахуванням сезонних складових, лінійні та нелінійні регресійні моделі,

різноманітні авторегресійні моделі (ARMA, ARIMA, SARIMA тощо), моделі на основі ланцюгів Маркова, генетичних алгоритмів та нейронних мереж. Вивчення та застосування цих моделей розглядаються у відповідних навчальних курсах.

Побудові будь-якої моделі часового ряду (Рис. 1.4) передує дослідження властивостей ряду. Це включає визначення закону розподілу значень ряду, оцінку параметрів цього розподілу, визначення статистичних характеристик вибірки даних, перевірку стаціонарності ряду, сезонності тощо. Ці дослідження мають універсальний характер і не залежать від типу моделі, яка буде будуватися, ані від мети дослідження – пошуку аномалій чи прогнозування поведінки об'єкта в майбутньому, ані від прикладної галузі, в якій отримано дані.

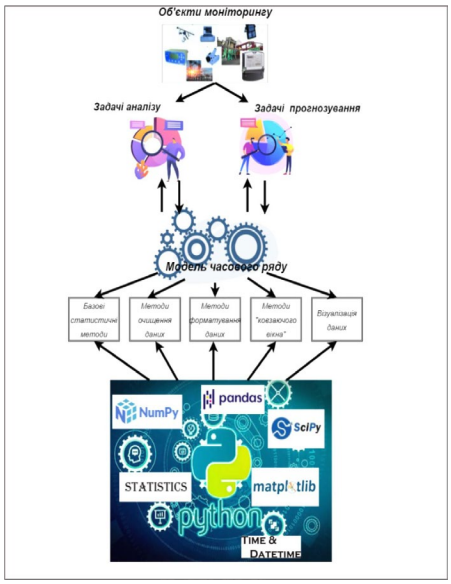

Рис. 1.4 – Узагальнюючий фреймворк часових рядів

Окремою, але не менш важливою задачею є первинна обробка даних: очищення, структурування, перетворення сирих даних у формат, необхідний для подальшого аналізу. Працюючи з часовими рядами, такі методи, як індексування на основі часу, повторна вибірка та ковзаючі вікна, можуть допомогти моделювати, аналізувати та прогнозувати значення в часі. Крім того, під час всіх етапів дослідження дуже важливо забезпечити можливість візуального відтворення інформації в різних форматах. Тобто, як на початковому («розвідувальному») етапі роботи, так і під час подальшого детального дослідження, дослідник повинен вільно володіти відповідними програмними засобами. Опис таких універсальних інструментів, які використовуються як на

початкових етапах аналізу часових рядів, так і в подальших складніших алгоритмах, є основою для подальших досліджень.

Набір програмних засобів та інструментів, які розглядаються, відносяться до так званої «екосистеми Python», тобто спеціалізованих модулів (бібліотек), що мають програмний інтерфейс, за допомогою якого програміст отримує можливість звертання до них з програм написаних на Python.

Нагадаємо, що Python – це високорівнева інтерпретована мова програмування, яка дозволяє швидко та легко створювати програмний код. Програміст, який використовує Python, може дотримуватися як процедурного, так і об'єктно-орієнтованого підходу до програмування. Наявність великої кількості бібліотек значно спрощує реалізацію складних процедур. Серед найпоширеніших модулів для дослідження часових рядів слід виділити такі:

Time та Datetime – це дві бібліотеки, які спеціалізуються на представленні, форматуванні, обробці та перетворенні даних, що безпосередньо пов'язані з часом. Вони забезпечують ефективну роботу з датами та часом, включаючи створення часових міток, обчислення різниці між датами, а також конвертацію між різними форматами представлення часу.

Statistics – це бібліотека, яка входить до стандартної бібліотеки Python і призначена для виконання задач, пов'язаних зі статистичною обробкою та аналізом даних. Вона включає в себе інструменти для обчислення основних статистичних показників, таких як середнє, медіана, мода, дисперсія та стандартне відхилення, що дозволяє швидко та ефективно аналізувати великі набори даних.

Numpy – це бібліотека, яка використовується для наукових обчислень. Інструменти цієї бібліотеки працюють з об'єктами типу «N-вимірний масив» і забезпечують основні математичні функції, а також набір сервісних операцій, таких як визначення розміру та розмірності масиву, обчислення основних статистичних параметрів, мінімальних та максимальних значень елементів у масивах та їх складових. Швидкість виконання арифметичних операцій робить numpy незамінною складовою будь-якого наукового, інженерного або технічного застосунку. Крім того, цей модуль часто використовується як основа для побудови інших модулів у екосистемі Python.

Використання цих бібліотек дозволяє програмістам ефективно працювати з часовими рядами, виконувати складні математичні та статистичні обчислення, а також легко маніпулювати та аналізувати дані. Завдяки їм Python стає

потужним інструментом для дослідження та обробки даних, що робить його популярним серед науковців, інженерів та аналітиків даних.

Pandas – ця бібліотека забезпечує високоефективні та водночас прості у використанні та представленні структури даних, такі як Series та DataFrame. Бібліотека розширює функціональність Python від простого збору та підготовки даних до аналізу даних. Водночас бібліотека забезпечує зручну та швидку роботу з даними, що представляють різні аспекти роботи з часом, що є основою аналізу часових рядів. Широке коло інструментів pandas можуть однаково добре застосовуються до аналізу будь-якого типу часових рядів, а гнучкі структури даних pandas можна застосовувати до даних часових рядів у будь-якій приклажній сфері, включаючи бізнес, науку, техніку, охорону здоров'я, кібербезпеку та багато інших.

Scipy – це бібліотека, яка використовується для наукових і технічних обчислень. Вона забезпечує функції оптимізації, обробки сигналів і зображень, інтеграції, інтерполяції та лінійної алгебри. Ця бібліотека використовується при реалізацію задач, пов'язаних з машинним навчанням. Бібліотека дуже потужна та розгалужена. В роботі буде розглянуто лише ту її частину, яка безпосередньо використовується для реалізації задач первинного аналізу часових рядів.

Matplotlib – на сьогоднішній день ця бібліотека – безумовний лідер по використанню в якості засобу візуалізації даних у різних форматах, таких як графік, розкид, стовпчикова діаграма, гістограма тощо. Вона містить усі функціональні можливості, що стосуються графіків, необхідні від побудови різноманітних дизайнерських особливостей їх представлення.

Як вже зазначалося, в роботі буде надана початкова інформація щодо використання можливостей перерахованих бібліотек. Головною метою є навіть не перерахування методів, що входять до складу бібліотек та їх аргументів, а ознайомлення з базовими ідеями, які були закладені розробниками при їх створенні. Це дасть можливість подальшого самостійного вдосконалення читача в відповідності до шляху його ознайомлення з теоретичними методами дослідження та моделювання динамічних систем.

Бакалаврська дипломна робота з аналізу та прогнозування часових рядів містить основні концепції та приклади застосування модулів Python, таких як time, datetime, statistics, numpy, pandas, scipy.stats, і matplotlib. У роботі також наведені приклади коду та графічні ілюстрації для кращого розуміння матеріалу. Різні типи задач, такі як виявлення аномалій і прогнозування майбутніх значень,

тісно пов'язані і часто використовують спільні моделі, серед яких найпоширенішими є моделі ковзаючого та експоненціального згладжування, авторегресійні моделі та нейронні мережі. Важливим є також дослідження властивостей ряду перед побудовою моделі. Робота надає студентам базу для подальшого поглиблення знань та навичок програмування, а також відкриває широкий простір для застосування отриманих знань у різних прикладних областях.

2 АНАЛІЗ ЧАСОВИХ РЯДІВ В ЗАДАЧАХ МОНІТОРИНГУ ШИФРОВАНОГО МЕРЕЖЕВОГО ТРАФІКУ

Інтернет став невід'ємною частиною сучасної цивілізації, забезпечуючи значні системні, соціальні та інфраструктурні зміни. Однак поширення інтернету збільшує залежність інституцій від його безперебійної, коректної та безпечної роботи, породжуючи нові проблеми, зокрема зростання кібератак. Комерційні, державні та некомерційні організації стикаються з програмами-вимагачами, шахрайством, крадіжкою облікових даних та інтелектуальної власності. Кібератаки спрямовані на отримання доступу, зміну або пошкодження комп'ютерних систем та мереж.

Технологія глибокого аналізу пакетів (Deep Packet Inspection – DPI) довгий час залишалася основною для забезпечення мережевої кібербезпеки. DPI-системи аналізують не лише заголовки пакетів, а й їхнє корисне навантаження, дозволяючи виявляти та блокувати віруси, фільтрувати інформацію та ідентифікувати підозрілий трафік. Методи перевірки мережі на основі сигнатур ефективні за умови, що атаки відомі розробникам програмного забезпечення заздалегідь. Однак зростання обсягу трафіку та пропускних можливостей каналів зв'язку, різноманітність середовищ передачі даних, зокрема мобільних мереж, IoT, космічних каналів, та збільшення частки шифрованого трафіку роблять традиційні методи менш ефективними.

Шифрування трафіку, що стало необхідністю для захисту від прослуховування та аналізу хакерами, до 2014 року не було поширеним. Більшість трафіку передавалася у відкритому вигляді, що дозволяло використовувати DPI для аналізу даних. Проте, з розвитком WiFi-мереж, провайдери могли відстежувати активність користувачів, що викликало необхідність шифрування даних. За даними Google, обсяг шифрованого трафіку зріс з 70% у 2016 році до 95% сьогодні, і до 2025 року майже весь трафік буде зашифрованим.

Зловмисники також почали використовувати шифрування для приховування атак. Це ускладнює завдання розробників систем захисту, оскільки необхідно знайти баланс між безпекою та конфіденційністю користувачів і можливістю виявлення загроз. Хоча самі дані в зашифрованому трафіку не можуть бути проаналізовані за допомогою DPI, інформація про заголовки пакетів, IP-адреси,

порти, інтервали затримки між пакетами, розмір пакетів та кількість пакетів за протоколами залишаються доступними для дослідження.

Мережевий трафік є складним динамічним процесом, суперпозицією багатьох потоків з множинними взаємопов'язаними характеристиками, які генеруються різними протоколами, програмами або операційними системами. Логічно припустити, що будь-яка мережева атака змінить деякі характеристики трафіку. Тому задача зводиться до вміння аналітично зафіксувати ці зміни за допомогою математичних та статистичних методів і автоматизувати процес з використанням програмних інструментів.

Процедура аналізу шифрованого трафіку може використовуватися не лише для детектування шкідливого трафіку, а й для підвищення якості обслуговування мереж, блокування заборонених ресурсів тощо. Однак ці задачі відрізняються. При підвищенні якості обслуговування мета полягає у пошуку ознак трафіку та прийнятті рішень на основі математичних процедур. Виявлення шкідливого трафіку складніше, особливо при новітніх атаках нульового дня, коли попередня інформація щодо поведінки ознак відсутня. Завдання полягає у виявленні аномалій у трафіку, які можуть свідчити про кіберзагрозу або інші зміни в мережі.

Це задача виявлення аномалій (anomaly detection) або визначення точки зміни (Change Point Detection - CPD) у поведінці об'єкта спостереження. Для виконання CPD необхідно обрати ознаки та алгоритм. Дослідники пропонують різні набори ознак, які характеризують трафік: частота появи вхідних та вихідних пакетів, співвідношення їхніх розмірів, тривалість сесій тощо. Оскільки задача ставиться у часі, і дані прив'язані до певного моменту, часовий ряд є найбільш точною аналітичною моделлю для задач аналізу поведінки мережі.

Алгоритми CPD (Рис. 2.1) ґрунтуються на концепції ковзаючого вікна, яке полягає у порівнянні двох наборів значень параметра на двох інтервалах часу. Інтервал, що починається раніше, називають «базовим вікном», а інше – «поточним вікном». Розміри вікон можуть бути однаковими або різними, а моменти початку збору даних відрізнятися на одиницю або більше. Алгоритм CPD є нескінченним циклом моніторингу. На першому кроці отримують дані з базового вікна та обчислюють їх характеристики. На другому кроці отримують дані поточного вікна і обчислюють їх характеристики. На третьому кроці порівнюють характеристики базового та поточного вікон. Якщо відмінностей не виявлено, процедура продовжується з другого кроку. Якщо зміни значимі,

генерується сигнал для сповіщення відповідальної особи. Поточне вікно стає новим базовим, і алгоритм продовжує роботу.

Характеристики, що отримуються з «сирих» даних базового та поточного вікон, можуть бути точковими оцінками основних статистичних параметрів: середнього, дисперсії, медіани, рідше моментів третього і четвертого порядку, коефіцієнтів асиметрії та ексцесу. Ці статистичні параметри використовуються для порівняння вікон і виявлення змін у поведінці мережевого трафіку. Аналіз часових рядів є важливим інструментом у задачах моніторингу шифрованого мережевого трафіку, дозволяючи виявляти аномалії та забезпечувати кібербезпеку в умовах зростаючої складності мережевих середовищ.

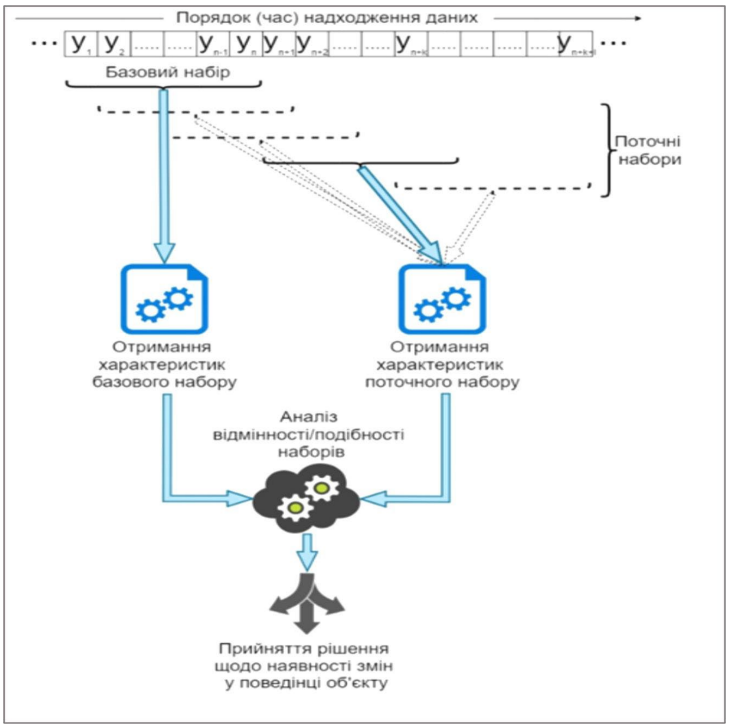

Рис. 2.1 – Алгоритм CPD автокореляції

Популярність підходу CPD (Change Point Detection) автокореляції обумовлена тим, що обчислені точкові оцінки можна легко і зрозуміло відобразити на інформаційних панелях для візуальної перевірки. Це дозволяє користувачам швидко оцінити зміни в поведінці об'єкта, базуючись на зібраних даних. Крім того, для більшості з цих оцінок існують методи визначення їхнього p-value – статистичного параметра, що показує, наскільки отримані оцінки сумісні з гіпотезою про незмінність поведінки об'єкта. Інакше кажучи, p-value

вказує на ймовірність помилково згенерувати сигнал зміни, якщо фактично жодної зміни не було. Ця помилка відома як помилка першого роду або "помилкове спрацювання".

Найбільш поширені статистичні тести, що використовуються для визначення відмінностей даних між двома вікнами спостереження, включають t-тест Стьюдента та його непараметричний аналог – тест Вілкоксона-Манна-Уїтні. Ці тести допомагають визначити, чи є зміни в середньому значенні або медіані наборів даних. Для тестування рівності або відмінності дисперсій у двох наборах даних використовуються F-критерій Фішера або більш стійкий критерій Левена. Для виявлення трендів (зростаючих або спадаючих) в даних застосовується критерій Аббе-Лінника, а для виявлення трендів у дисперсіях – критерій Фостера-Стюарта.

Більш складні та точні алгоритми, хоча й більш ресурсоємні, використовують інші ознаки, отримані з даних базового та поточного вікон. До них належать різноманітні методи порівняння двох відповідних гістограм, алгоритми, що використовують поняття ентропії даних, алгоритми фрактального аналізу, а також моделі часових рядів різної складності, від найпростіших регресійних до моделей типу SARIMA. Вони можуть порівнювати отримані моделі або аналізувати відхилення даних поточного вікна від моделі, побудованої на даних базового вікна.

Детальний опис конкретних алгоритмів CPD, їх порівняння, виявлення сильних і слабких сторін, швидкість виявлення змін щодо моменту їх виникнення та загальна ефективність виходять за рамки даної роботи. У наступних розділах будуть розглянуті інструменти різних модулів екосистеми Python, які разом дозволять читачу перейти до вивчення відповідних реалізацій алгоритмів CPD. Це допоможе вирішувати завдання, що виникають при аналізі часових рядів у різних прикладних сферах: науці, техніці, інженерії, соціальних науках, медицині, фінансах тощо.

3 ПРОГРАМНІ ЗАСОБИ PYTHON ТА МОДУЛІВ РОЗШИРЕННЯ ДЛЯ РОБОТИ З ЧАСОМ ТА ДАТАМИ

Часовий ряд – це спосіб представлення даних, де основне значення має інформація про момент або інтервал часу, коли кожне значення вибірки було отримано. Тому при аналізі часових рядів досліднику необхідні програмні інструменти, що дозволяють зручно опрацьовувати інформацію, пов'язану з часом або датою. В базовому Python, а також у пакетах numpy та pandas, є кілька таких інструментів для роботи з часом і датами.

3.1. Модуль time

Модуль time зі стандартної бібліотеки Python містить корисні методи для роботи з часом. Він дозволяє отримувати інформацію про поточну дату і час з точністю до мілісекунди, виводити ці дані у необхідному форматі, а також керувати виконанням програми, додаючи затримки по таймеру. Найпростіший виклик функції для визначення поточного часу виглядає так:

```
import time
time.time()
>>>>
Out[1]: 1668789408.0911138
```

Цей час представлений у вигляді "системного часу", що відповідає кількості секунд від початку "епохи" до поточного моменту. "Епоха" зазвичай починається 1 січня 1970 року, і для цього моменту лічильник секунд дорівнює нулю. Системний час представлений у форматі float. Отримане значення можна перевести у години, дні, місяці та роки для зручного відображення часу.

Варто зазначити, що деякі системи можуть мати іншу відправну точку "епохи".

Наприклад, функція time.gmtime(0) повертає час, який відображає початок епохи Unix – 1 січня 1970 року, 00:00:00 UTC. Це є очікуваним результатом:або отримати той самий результат, вказавши

```
print(time.gmtime(0))
```

```
# time.struct_time(tm_year=1970, tm_mon=1, tm_mday=1, tm_hour=0, tm_min=0, tm_sec=0, tm_wday=3, tm_yday=1, tm_isdst=0)
```

Ця структура часу містить деталі дати та часу, включаючи день тижня та день року.

Функція time.ctime() конвертує час у зручний для сприйняття строковий формат. Наприклад:

```
print(time.ctime(1668789408.0911138))
# Fri Nov 18 18:36:48 2024
```

Отриманий результат показує час у строковому форматі, що є зручним для відображення і перевірки.

Для отримання локального часу використовується функція time.localtime(), яка повертає поточний системний час у вигляді об'єкта struct_time:

```
loc_time = time.localtime()
print(loc_time)
# time.struct_time(tm_year=2024, tm_mon=5, tm_mday=18, tm_hour=22, tm_min=12, tm_sec=5, tm_wday=4, tm_yday=322, tm_isdst=0)
```

Ці засоби дозволяють, за необхідності, налаштувати результат так, щоб він включав, наприклад, лише значення місяця та дня. Для цього достатньо записати у наведеній вище функції як значення першого аргументу літерал "%m/%d". Або, при необхідності, навпаки: "%d/%m".

Якщо потрібно працювати з українськими назвами місяців та днів тижня, необхідно попередньо встановити відповідну локалізацію, звернувшись до функцій іншого допоміжного стандартного модуля Python:

```
s = time.time()
loc_time = time.localtime(s)
print(loc_time.tm_year)
```

Це дозволяє отримати значення конкретного атрибуту з об'єкта struct_time, наприклад, року.

Python надає багато корисних функцій для роботи з часом і датами. Однією з таких є функція time.strptime(), яка є зворотною до функції strftime(). Ця функція приймає два аргументи: строку, яку потрібно перетворити на дату і час, та опис формату, який визначає, як інтерпретувати строку. Результатом є об'єкт struct_time, що представляє дату і час.

```
st_time1 = time.strptime("18-травня 2024.", "%d-%B %Y.")
st_time2 = time.strptime("22:17:30", "%H:%M:%S")
print(st_time1)
print(st_time2)
```

```
time.struct_time(tm_year=2024, tm_mon=5, tm_mday=18, tm_hour=0, tm_min=0, tm_sec=0, tm_wday=4, tm_yday=139, tm_isdst=-1)
time.struct_time(tm_year=1900, tm_mon=1, tm_mday=1, tm_hour=22, tm_min=17, tm_sec=30, tm_wday=0, tm_yday=1, tm_isdst=-1)
```

Як видно, якщо у вхідному рядку відсутня інформація про деякі елементи дати і часу, вони заповнюються значеннями за замовчуванням.

Функція time.sleep() дозволяє призупинити виконання скрипту на певну кількість секунд. Це може бути корисно у різних ситуаціях, наприклад, для очікування завершення запису файлу на диск або для паузи між запитами до веб-сервера, щоб не перевантажувати його.

Розглянемо приклад використання time.sleep(), де скрипт призупиняється на 10 секунд:

```
pause = 10
print("Почали... і не хвилюйтеся, ми встановили затримку в 10 секунд.")
time.sleep(pause)
print(f"{pause} секунд пройшло. Продовжуємо роботу.")
```

Цей код спочатку виводить повідомлення, що скрипт почав роботу і встановлено затримку. Потім виконується функція time.sleep(), яка зупиняє виконання на 10 секунд. Після паузи виводиться повідомлення про продовження роботи.

Python також має більш потужну бібліотеку для роботи з датами і часом – datetime. Вона надає зручні методи для створення, маніпулювання та форматування дат і часу.

```
from datetime import datetime, timedelta

now = datetime.now()
print("Поточна дата і час: ", now)

Додавання або віднімання часу можна виконати за допомогою timedelta:
new_date = now + timedelta(days=5)
print("Дата через 5 днів: ", new_date)
```

Додавання або віднімання часу можна виконати за допомогою timedelta:
new_date = now + timedelta(days=5)
print("Дата через 5 днів: ", new_date)

```
Форматування дати і часу у строку здійснюється за допомогою strftime():
formatted_date = now.strftime("%Y-%m-%d %H:%M:%S")
print("Відформатована дата: ", formatted_date)

Аналогічно, можна парсити строку у об'єкт datetime за допомогою strptime():
date_str = "2024-05-18 22:17:30"
parsed_date = datetime.strptime(date_str, "%Y-%m-%d %H:%M:%S")
print("Парсинг дати: ", parsed_date)
```

3.2. Модуль datetime

На відміну від модуля time, який використовується для визначення поточного часу, існує модуль datetime, призначений для маніпуляцій, логічних і арифметичних операцій, порівнянь та обробки даних, що зберігають значення часу і дати. Модуль datetime в Python має 5 основних підмодулів, кожен з яких реалізує відповідний клас об’єктів:

- date – для роботи з об’єктами-датами;
- time – для маніпулювання об’єктами, які представляють час;

- datetime – комбінація дати та часу;
- timedelta – для роботи з поняттям «тривалість часу» або «інтервал»;
- tzinfo – для роботи з часовими поясами.

Ці підмодулі дозволяють представлення інформації щодо дати та часу в різних форматах, наприклад, «July 4», «March 8, 2022 22:00», «31 December 2022, 23:59:59», «07/23/2022», «23/07/2023» тощо. Такі дані називаються «часовою міткою» або «міткою часу». Інші дані представляють інтервали між моментами часу, як-от «1 year 3 month 5 days», «24 hours, 17 minutes, 56 seconds, 268 milliseconds», «3 weeks, 3 days, 3 hours, 3 minutes».

При обробці інформації можна додати інтервал до дати (мітки часу) і отримати нову дату або скласти інтервали, отримавши більший інтервал. Однак складання двох міток часу не має сенсу. Модуль datetime забезпечує коректну обробку даних обох типів у виразах, де вони можуть бути присутні одночасно.

Імпортуємо модуль datetime і створимо об’єкт дати (екземпляр об’єкту класу date):

import datetime as dt st_date=dt.date(year=2022,
month=5, day=10) print(st_date)
2024-5-10

Отримане значення представлене у стандартному для модуля datetime форматі і є незмінним об'єктом (у термінах опису об'єктів у Python). Спроба ввести некоректну дату, наприклад, dt.date(2022, 26, 3), призведе до виключення типу ValueError.

Створимо об'єкт класу time (важливо не плутати об'єкти класу time модуля datetime з об'єктами, що використовуються функціями модуля time). Цікаво поглянути у менеджер змінних IDE, щоб побачити, наскільки різняться представлення створених раніше об'єктів st_date та st_time.

st_time= dt.time(hour=12, minute=45, second=16, microsecond=257) print(st_time)
12:45:16.000257

Якщо потрібно представити мітку часу з точністю до дати, години, хвилини, секунди та мілісекунди, використовують об'єкти класу datetime модуля datetime, який є наступником класів date та time.

Цікаво поглянути у менеджер змінних IDE, щоб побачити різницю у представленні об'єктів st_date та st_time. Об'єкти класу datetime модуля datetime, який є наступником класів date та time, використовуються для представлення міток часу з точністю до дати, години, хвилини, секунди та мілісекунди.

Наприклад, створимо об'єкт, що представляє конкретний момент часу:

```
import datetime as dt

st_moment = dt.datetime(year=2022, month=11, day=10, hour=13, minute=0, second=55)
print(st_moment)
#  2022-11-10 13:00:55
```

Важливо зазначити, що параметри можна виключати тільки послідовно, починаючи з останнього. Наприклад, пропустити параметр year або month не можна – це викличе помилку типу TypeError.

Використання datetime.combine()

Для створення об'єкта datetime, що комбінує дату та час з різних об'єктів, можна скористатися функцією datetime.combine(). Наприклад:

```
st_date = dt.date(year=2022, month=11, day=10)
st_time = dt.time(hour=13, minute=0, second=55)
st_moment2 = dt.datetime.combine(st_date, st_time)
print(st_moment2)
#2022-11-10 13:00:55
```

Об'єкти datetime забезпечують зручний доступ до окремих компонентів часу. Наприклад:

```
print('Рік:', st_moment1.year)
print('Місяць:', st_moment1.month)
print('День:', st_moment1.day)
```

Модуль datetime також дозволяє отримувати додаткову інформацію про дату та час, наприклад, номер дня в році чи номер дня в тижні:

```
print('Номер дня в році:', st_moment1.timetuple().tm_yday)
print('Номер дня в тижні:', st_moment1.weekday())
print('Номер дня в тижні для дати:', st_date.weekday())
```

При визначенні номера дня у тижні приймається, що понеділку відповідає значення 0, вівторку – 1 і т.д. Для отримання поточного часу в модулі datetime використовуються функції now() або today():

```
now = dt.datetime.now()
print(now)
```

Різниця між datetime.now() та datetime.today() полягає в тому, що перша дозволяє працювати не тільки з локальним часом, але й з часом за Грінвічем. Метод strftime() також реалізований для об'єктів модуля datetime:

```
tod = dt.datetime.today()
print(tod.strftime("%d.%m.%Y"))
# >>> 19.5.2024
print(tod.strftime("%H:%M:%S"))
# >>> 19:42:45
```

Для зміни певних компонентів дати та часу, представлених у форматі datetime, використовується метод datetime.replace(). Розглянемо приклад:

```
time_mk = dt.datetime(2015, 3, 21)
print(time_mk)
# >>> 2015-03-21 00:00:00
time_mk2 = time_mk.replace(year=2022)
print(time_mk2)
# >>> 2024-03-21 00:00:00
time_mk2 = time_mk2.replace(month=time_mk2.month + 5)
print(time_mk2)
# >>> 2024-08-21 00:00:00
```

```
time_mk2 = time_mk2.replace(hour=3, second=54)
print(time_mk2)
# >>> 2024-08-21 03:00:54
```

У прикладі показано, як можна обчислити нове значення параметра на основі старого його значення (time_mk2.month + 5), але це потребує уваги програміста. Оскільки діапазони атрибутів обмежені природними кордонами (наприклад, значення параметра month має лежати в діапазоні від 1 до 12, hour – від 0 до 23 і т.д.), відповідальність за коректність змін покладається на програміста.

Метод timestamp() дозволяє конвертувати об'єкт datetime у його представлення у вигляді кількості секунд з початку епохи (1 січня 1970 року). Наприклад, розглянемо об'єкт datetime для конкретного моменту часу:

```
import datetime as dt

st_moment1 = dt.datetime(year=2024, month=5, day=10, hour=13, second=55)
print(st_moment1)
sys_time = st_moment1.timestamp()
print(sys_time)
# 2024-05-10 13:00:55
# 1715368855.0
```

Цей метод є дуже корисним для зберігання або передачі часу в форматі, сумісному з багатьма іншими системами.

Одна з найбільш поширених задач при роботі з датами та часом – визначення інтервалу між двома подіями. За допомогою класу datetime це можна зробити дуже просто. Наступний приклад демонструє, як обчислити різницю між поточною датою та певною датою у минулому:

```
time_mk1 = dt.datetime.now()
time_mk2 = dt.datetime(2015, 3, 21)
time_int = time_mk1 - time_mk2
print(time_int)
# 3350 days, 13:00:55.123456
```

Результатом є об'єкт timedelta, який зберігає інтервал як поєднання днів, секунд і мікросекунд. Клас datetime.timedelta дозволяє зручно працювати з інтервалами часу, перетворюючи передані параметри у відповідні одиниці:

- millisecond перетворюється на 1000 мікросекунд.
- minute перетворюється на 60 секунд.
- hour перетворюється на 3600 секунд.
- week перетворюється на 7 днів.

Для наочності створимо об'єкт класу timedelta з різними параметрами:

```
delta = dt.timedelta(
   days=50, seconds=27, microseconds=10, milliseconds=29000,
   minutes=5, hours=8, weeks=2
)
print(delta)
# 64 days, 8:05:27.010010
```

Створений об’єкт time_int є об’єктом класу timedelta, що реалізований засобами модуля datetime. Екземпляр об’єкта datetime.timedelta зберігає значення інтервалу як поєднання days, seconds і microseconds, а решта переданих параметрів конвертуються в ці одиниці:

- millisecond перетворюється на 1000 microseconds.
- minute перетворюється на 60 seconds.
- hour перетворюється на 3600 seconds.
- week перетворюється на 7 days.

Для наочності створимо об'єкт класу timedelta з різними параметрами:

```
delta = dt.timedelta(
   days=50, seconds=27, microseconds=10, milliseconds=29000,
   minutes=5, hours=8, weeks=2
)
print(delta)
# >>> 64 days, 8:05:27.010010
```

Об'єкти timedelta можна використовувати для додавання або віднімання часу від об'єктів datetime. Наприклад:

```
new_date = dt.datetime.now() + delta
print(new_date)
```

>>> Поточна дата та час + 64 дні, 8 годин, 5 хвилин, 27 секунд, 10 мікросекунд

Зверніть увагу, що кількість днів в інтервалі була перерахована (50 днів + 2 тижні = 50 + 2 * 7 = 64 дні). Хоча функція print() для таких об'єктів робить відповідний перерахунок, насправді об'єкт зберігає лише три атрибути:

datetime.timedelta(days=64, seconds=29156, microseconds=10)

Отримані таким чином компоненти інтервалу часу представлені у вигляді змінних типу int і можуть використовуватися за звичайними правилами роботи з даними цього типу в Python. Також об'єкти-інтервали можна додавати один до одного, а також додавати або віднімати їх від об'єктів часових міток, утворюючи нові об'єкти. У наступному прикладі показано, як отримати нову дату, маючи початкову дату та інтервал до іншої дати.

```
date1 = dt.datetime(2016, 12, 5)
delta1 = dt.timedelta(weeks=8, days=8, hours=2, minutes=5, seconds=17)
print(date1)
date2 = date1 + delta1
print(date2)
```

Зверніть увагу, що роки та місяці коректно перераховані. Навіть якщо враховуються високосні роки або різна кількість днів у місяцях, модуль datetime автоматично забезпечує правильність обчислень.

Крім цього, з об'єктами класу timedelta можна виконувати різні операції: множення на ціле число (що збільшує інтервал у відповідну кількість разів) або навіть на дійсне число (що призводить до округлення результату), ділення інтервалу на інтервал, унарну операцію «мінус» (зміну знаків значень компонентів на протилежний), логічні операції порівняння тощо. Якщо компоненти значень об'єкта від'ємні, це означає, що при додаванні такого інтервалу до певної дати результуюча дата буде меншою за початкову, тобто буде в минулому.

Метод total_seconds() дозволяє отримати значення інтервалу в секундах:

```
delta1 = dt.timedelta(weeks=8, days=8, hours=2, minutes=5, seconds=17)
delta_sec = delta1.total_seconds()
print(delta_sec)
# >>> 5537117.0
```

3.3. Робота з часом і датами в бібліотеці numpy

Основний недолік описаних вище об’єктів та засобів роботи з ними полягає в їх відносній повільності, що стає особливо помітною при роботі з великими об’ємами даних. Для цього в numpy використовується формат datetime64, який кодує дати як 64-бітові цілі числа і дозволяє скористатися всіма перевагами цього пакету.

На відміну від об’єктів модулів time та datetime, що підтримують роки в діапазоні від 1 до 9999 н.е., datetime64 дозволяє використовувати дати до нашої ери; роки до н.е. дотримуються астрономічної нумерації років, тобто рік 2 до н.е. має номер -1, рік 1 до н.е. – номер 0 і так далі.

Створити дані у форматі datetime64 можна різними способами:

```
n_dt1 = np.datetime64('2023-02-25')
print(n_dt1)
# >>> 2023-02-25

n_dt2 = np.datetime64('2023-02')
print(n_dt2)
# >>> 2023-02
```

Якщо потрібно задати лише рік і місяць, але з точністю до дня, можна використовувати другий параметр:

```
n_dt3 = np.datetime64('2023-02', 'D')
print(n_dt3)
# >>> 2023-02-01
```

Для створення мітки часу з точністю до хвилин:

```
n_dt4 = np.datetime64('2017-12-03T03:30')
print(n_dt4)
# >>> 2017-12-03T03:30
```

А для точності до мілісекунд:

```
n_dt5 = np.datetime64('2025', 'ms')
print(n_dt5)
```

```
# >>> 2025-01-01T00:00:00.000
```

Щоб визначити поточну мітку часу з максимальною точністю:

```
n_dt = np.datetime64(dt.datetime.now())
print(n_dt)
# >>> 2022-11-20T16:56:33.822641
```

Основні переваги numpy виявляються при роботі з масивами даних, зокрема з часовими мітками:

```
n_arr1 = np.array([np.datetime64('2020-10-22'), np.datetime64('2020-10-23'), np.datetime64('2020-10-24')], dtype='datetime64[s]')
print(n_arr1)
# >>> ['2020-10-22T00:00:00' '2020-10-23T00:00:00' '2020-10-24T00:00:00']
```

Для отримання або зміни значення окремого елемента масиву використовується звичайний для numpy спосіб:

```
elt = n_arr1[1]
print(elt)
# >>> 2020-10-23T00:00:00

n_arr1[1] = np.datetime64('2022-12-25')
print(n_arr1)
# >>> ['2020-10-22T00:00:00' '2022-12-25T00:00:00' '2020-10-24T00:00:00']
```

Зручним є використання функції numpy.arange() для створення масивів послідовних дат, що часто використовуються при дослідженні часових рядів:

```
n_arr2 = np.arange('2022-10-22', '2022-11-07', dtype='datetime64[D]')
print(n_arr2)
# >>> ['2022-10-22' '2022-10-23' '2022-10-24' '2022-10-25' '2022-10-26'
#     '2022-10-27' '2022-10-28' '2022-10-29' '2022-10-30' '2022-10-31'
#     '2022-11-01' '2022-11-02' '2022-11-03' '2022-11-04' '2022-11-05'
#     '2022-11-06']
```

Інший спосіб створення діапазону дат:

```
n_arr3=n_dt + np.arange(12) print(n_arr3)
>>>>
```

['2022-11-20T16:56:33.822641''2022-11-20T16:56:33.822642''2022-11-20T16:56:33.822643''2022-11-20T16:56:33.822644''2022-11-20T16:56:33.822645''2022-11-20T16:56:33.822646''2022-11-20T16:56:33.822647''2022-11-20T16:56:33.822648''2022-11-20T16:56:33.822649''2022-11-20T16:56:33.822650''2022-11-20T16:56:33.822651' '2022-11-20T16:56:33.822652']

Об'єкт datetime64 представляє окремий момент часу. Навіть якщо дві мітки часу мають різні одиниці вимірювання, вони можуть представляти той самий момент часу, що видно з наступного прикладу:

```
np.datetime64('2022') == np.datetime64('2022-01-01')
# >>> True
```

Тут значення зліва від оператора порівняння приводиться до точності, визначеної правим значенням, і лише потім виконується операція порівняння. Відповідно, у наступному випадку отримуємо негативний результат:

```
np.datetime64('2022') == np.datetime64('2022-10')
# >>> False
```

Дані у форматі datetime64 можуть порівнюватися між собою за допомогою звичайних логічних операторів. Більшим вважається той момент часу, який вказує на пізнішу дату:

```
np.datetime64('2022-09') >= np.datetime64('2022-10')
# >>> False

np.datetime64('2022-09') <= np.datetime64('2022-10')
# >>> True
```

Для роботи з часовими інтервалами в numpy використовується тип даних timedelta64. Аргументами для timedelta64 є число, яке представляє кількість одиниць, і одиниця дати/часу, наприклад, (D) день, (M) місяць, (Y) рік, (h) година, (m) хвилина або (s) секунда:

```
n_dl1 = np.timedelta64(1, 'D')
print(n_dl1)
# >>> 1 days
```

```
n_dl2 = np.timedelta64(4056, 's')
print(n_dl2)
# >>> 4056 seconds
```

Дані у форматах datetime64 та timedelta64 можуть використовуватися разом, комбінуючись у арифметичні вирази та дозволяючи зручні обчислення дати й часу:

```
n_dl3 = np.datetime64('2019-01-01') - np.datetime64('2018-01-01')
print(n_dl3)
# >>> 365 days

n_dl4 = np.datetime64('2019-01-01', 's') - np.datetime64('2018-01-01')
print(n_dl4)
# >>> 31536000 seconds

n_dt6 = np.datetime64('2021-06-15T10:46') + np.timedelta64(12, 'h')
print(n_dt6)
# >>> 2021-06-15T22:46

res = np.timedelta64(3, 'W') / np.timedelta64(1, 'D')
print(res)
# >>> 21.0
```

Таким чином, numpy забезпечує ефективну роботу з часовими даними, дозволяючи легко виконувати складні обчислення і маніпуляції з датами та інтервалами часу.

Одиниці timedelta64 «Y» (роки) та «M» (місяці) обробляються спеціальним чином через неможливість точно визначити кількість днів у році або місяці без урахування високосних років або конкретного місяця. Наприклад:

```
year_dl = np.timedelta64(1, 'Y')
month_dl = np.timedelta64(year_dl, 'M')
```

```
print(month_dl)
# >>> 12 months
```

Спроба виконати подібний оператор, як

```
month_dl = np.timedelta64(year_dl, 'D')
```

призведе до винятку типу «TypeError».

Функція numpy.busday_offset() також корисна для врахування робочих днів, включаючи святкові:

```
print(np.busday_offset('2021-02-17', 2))
print(np.busday_offset('2021-02-19', 2))
# >>> 2021-02-19
# >>> 2021-02-23
```

У першому випадку функція показує дату, віддалену від вказаної на 2 робочих дні, а в другому - на 4 робочих дні, оскільки два дні припали на суботу та неділю, які вважаються неробочими.

Також функція busday_count() підраховує кількість робочих днів в проміжку часу:

```
print(np.busday_count(np.datetime64('2021-01-01'), np.datetime64('2022-01-01')))
# >>> 261
```

Отже, у 2021 році більше 100 календарних днів припало на вихідні та святкові дні

3.4. Робота з часом та датами в pandas

Бібліотека pandas надає широкі можливості по роботі з даними часових рядів, використовуючи об'єкти Timestamp і Period.

Об'єкт Timestamp можна створити таким чином:

```
time_stamp = pd.Timestamp(dt.datetime(2022, 12, 13))
print(time_stamp)
# >>> 2022-12-13 00:00:00
```

Timestamp має багато атрибутів, які дозволяють отримати окремі компоненти часової мітки:

```
print(time_stamp.year)
print(time_stamp.day)
# >>> 2022
# >>> 13
```

Для відображення часових інтервалів використовується об'єкт Period, який має атрибут freq для зберігання інформації про частоту:

```
period = pd.Period('2023-01')
print(period)
# >>> Period('2023-01', 'M')
period1 = pd.Period('2023-01', freq='D')
print(period1)
# >>> Period('2023-01-01', 'D')
```

Ці об'єкти дозволяють зручно працювати з даними часових рядів у pandas.

Бібліотека pandas надає багато зручних інструментів для роботи з часовими рядами, використовуючи об'єкти Timestamp і Period.

Для створення окремих часових міток (Timestamp) в pandas можна використовувати такий синтаксис:

```
time_stamp = pd.Timestamp(dt.datetime(2022, 12, 13))
print(time_stamp)
# >>> 2022-12-13 00:00:00
```

Об'єкт Timestamp має багато атрибутів, що дозволяють отримувати окремі компоненти часової мітки:

```
print(time_stamp.year)
print(time_stamp.day)
# >>> 2022
# >>> 13
```

Для відображення часових інтервалів використовується об'єкт Period, який має атрибут freq для визначення частоти:

```
period = pd.Period('2023-01')
print(period)
# >>> Period('2023-01', 'M')

period1 = pd.Period('2023-01', freq='D')
print(period1)
# >>> Period('2023-01-01', 'D')
```

При необхідності можна конвертувати частоту для існуючого об'єкту за допомогою методу asfreq():

```
period2.asfreq('H')
```

Об'єкти pd.Timestamp і pd.Period можуть конвертуватися один в один:

```
period2.to_timestamp().to_period('M')
```

При роботі з часовими рядами зазвичай потрібно створити послідовність дат. Для цього можна використовувати об'єкти типу Series:

```
values = [25, 50, 15, 67, 70, 9, 28, 30, 32, 12]
start = dt.datetime(2022, 3, 31)
dates = [start - dt.timedelta(days=x) for x in range(0, len(values))]
ts = pd.Series(values, index=dates)
print(ts)
```

Ці функції та об'єкти дозволяють ефективно працювати з даними часових рядів в бібліотеці pandas.

```
values = [25, 50, 15, 67, 70, 9, 28, 30, 32, 12]
start = dt.datetime(2022, 3, 31)
dates = [start – dt.timedelta(days=x) for x in range(0, len(values))] ts = pd.Series(values, index=dates)
print(ts)
>>>>
2022-03-31 25
```

```
2022-03-30 50
2022-03-29 15
2022-03-28 67
2022-03-27 70
2022-03-26 9
2022-03-25 28
2022-03-24 30
2022-03-23 32
2022-03-22 12
dtype: int64
```

Для роботи з індексом, що містить часові мітки, використовується тип даних DatetimeIndex в pandas.

Щоб переглянути індекс серії у вказаному прикладі, можна використати метод index:

```
ts.index
```

```
DatetimeIndex(['2022-03-31', '2022-03-30', '2022-03-29', '2022-03-28', '2022-03-27', '2022-03-26', '2022-03-25', '2022-03-24', '2022-03-23', '2022-03-22'], dtype='datetime64[ns]', freq=None)
```

Також можна створити інший об'єкт типу Series з використанням тих самих часових міток:

```
values2 = [32, 54, 18, 61, 72, 19, 21, 33, 29, 17]
ts2 = pd.Series(values2, index=dates)
```

Об'єкти Series можна обробляти арифметичні операції, які застосовуються до відповідних елементів:

```
print((ts + ts2)/2)
```

Якщо індекси не збігаються, результатом можуть бути значення NaN:

```
print(ts + ts3)
```

Для створення послідовності дат можна скористатися методом pd.date_range():

```
ind_ts = pd.date_range('30/10/2022', '11/08/2022')
```

Також можна задати лише одну дату і кількість періодів для створення послідовності:

ind_ts4 = pd.date_range(end='30/10/2022', periods=4)

Pandas - це популярний бібліотека Python для роботи з даними, яка має потужні можливості для роботи з часовими рядами. Одним з ключових компонентів роботи з часовими рядами в Pandas є DatetimeIndex - спеціальний тип даних, який представляє індекс часових міток.

DatetimeIndex може бути створений з різних джерел, таких як списки дат, строки, або за допомогою функції pd.date_range(). Ця функція дозволяє гнучко генерувати послідовність часових міток, вказуючи початкову та кінцеву дати, а також частоту (наприклад, щодня, щомісяця, щорічно).

Операції над часовими рядами, представленими у вигляді Series в Pandas, ґрунтуються на відповідних операціях над DatetimeIndex. Наприклад, додавання двох часових рядів з однаковими індексами призведе до покомпонентного додавання значень, а віднімання - до покомпонентного віднімання.

Важливою особливістю роботи з часовими рядами в Pandas є те, що індекси можуть не збігатися. В такому разі, при виконанні арифметичних операцій, для неспівпадаючих дат значення результату замінюються на NaN (не число).

Pandas також надає можливості для маніпулювання індексами часових рядів, наприклад, для зміщення дат, вилучення певних дат або для групування даних за часовими інтервалами.

DatetimeIndex - це потужний інструмент для роботи з часовими рядами в Pandas, який дозволяє зручно та ефективно обробляти та аналізувати часові дані.

date1 = pd.date_range(‘01-01-2021’, periods = 4, freq =‘W’) print(date1)
>>>>
DatetimeIndex([‘2021-01-03’, ‘2021-01-10’, ‘2021-01-17’, ‘2021-01-24’],
dtype=‘datetime64[ns]’, freq=‘W-SUN’)

date2 = pd.date_range(‘01-01-2021’, periods = 4, freq =‘7D’) print(date2)
>>>>
DatetimeIndex([‘2021-01-01’, ‘2021-01-08’, ‘2021-01-15’, ‘2021-01-22’],
dtype=‘datetime64[ns]’, freq=‘7D’)

Тепер розглянемо особливості використання часових даних в основних об’єктах пакету pandas, а саме в об’єктах типу DataFrame.

date_rng = pd.date_range(start=‘1/1/2022’, end=‘1/08/2022’, freq=‘H’) df = pd.DataFrame(date_rng, columns=[‘Дата’])
df[‘Значення’] = np.random.randint(0,100,size=(len(date_rng))) df.head()

Перші п’ять рядочків цього об’єкту виглядають так:
Дата Значення
0 2022-01-01 00:00:00 86
1 2022-01-01 01:00:00 76
2 2022-01-01 02:00:00 80
3 2022-01-01 03:00:00 95
4 2022-01-01 04:00:00 86

Конвертація індексу DataFrame в індекс дати та часу є важливим кроком при роботі з часовими рядами в pandas. Це дозволяє легко виконувати операції, які вимагають фільтрації, групування або аналізу даних за допомогою часових міток.

df[‘datetime’] = pd.to_datetime(df[‘Дата’]) df = df.set_index(‘datetime’) df.drop([‘Дата’], axis=1, inplace=True) df.head()
>>>>

Значення
datetime
2022-01-01 00:00:00 86
2022-01-01 01:00:00 76
2022-01-01 02:00:00 80
2022-01-01 03:00:00 95
2022-01-01 04:00:00 86

Після конвертації індексу, ви більше не маєте просто числовий індекс, але часові мітки, що дозволяє вам здійснювати операції на основі часу. Наприклад, ви можете легко відфільтрувати дані, використовуючи дати. В цьому випадку,

фільтруючи дані, що відповідають лише 5 числу, ви отримаєте дані тільки за цю дату.

Нагадаємо, що зробити це можна за допомогою, наприклад, функції strptime() опис якої був наданий раніше:

timestamp_date_rng= [dt.datetime.strptime(x,'%B-%d-%Y') for x in string_date_rng_2]

де string_date_rng_2 – масив часових міток, які представлені в строковому форматі.

Коли дані надходять у строковому форматі, їх потрібно конвертувати у формат об'єкту типу часової мітки. Для цього можна використовувати функцію strptime(), яка дозволяє зчитувати строки та перетворювати їх у об'єкти дати та часу. Наприклад, якщо ваші дані представлені у строковому форматі, ви можете використати strptime() для конвертації цих даних у формат об'єкту часової мітки.

Перші декілька рядків результату показано нижче:

Значення
datetime
2022-01-05 00:00:00 27
2022-01-05 01:00:00 0
2022-01-05 02:00:00 7
2022-01-05 03:00:00 63
2022-01-05 04:00:00 18
2022-01-05 05:00:00 17
2022-01-05 06:00:00 61

Усі ці кроки дозволяють легко та ефективно маніпулювати та аналізувати часові ряди у pandas, що є важливим аспектом при роботі з даними, які містять часові залежності.

Звісно, аналогічного результату можна досягнути і з використанням безпосередньої вказівки дати, інформація про яку нас цікавить, наприклад:

df[‘2022-01-05’]

Як і в звичайному DataFrame, індекси-мітки часу можуть використовуватися для створення зрізів:

df[‘2022-01-04 02:00’:’2022-01-06 14:00’]
>>>>
2022-01-04 02:00:00 89
2022-01-04 03:00:00 88
2022-01-04 04:00:00 72
2022-01-04 05:00:00 12
2022-01-04 06:00:00 39
...
2022-01-06 10:00:00 44
2022-01-06 11:00:00 8
2022-01-06 12:00:00 42
2022-01-06 13:00:00 95

При роботі з часовими рядами у pandas можна змінювати частоту відображення даних для виконання різних операцій. Розглянемо, що відбувається при збільшенні та зменшенні частоти відображення даних.

При збільшенні частоти даних, наприклад, з квартальної на щомісячну, pandas додає нові дати до існуючого індексу DateTimeIndex, що призводить до збільшення кількості рядків. Нові рядки міститимуть пропущені значення, які можна заповнити різними методами, такими як ffill, bfill або fill_value.

При зменшенні частоти даних, кількість рядків у DataFrame зменшується. Метод resample() дозволяє змінити частоту даних часового ряду та застосувати певні операції до нової частоти. Наприклад, можна підрахувати сумарне значення за добу або середньогодинне значення. Після цього обчислення рядки можуть бути агреговані або заповнені, залежно від обраних методів.

```
# Збільшення частоти даних з квартальної на щомісячну
df_monthly = df.resample('M').asfreq()
df_monthly.fillna(method='ffill', inplace=True)
```

```
# Зменшення частоти даних з погодинної на денну та обчислення сумарного та середньогодинного значення
daily_sum = df.resample('D').sum()
daily_mean = df.resample('D').mean()
```

Таким чином, зміна частоти даних дозволяє агрегувати, або, навпаки, деталізувати дані залежно від потреб аналізу. Важливо враховувати, які методи заповнення або агрегації використовуються, щоб коректно обробляти дані при зміні їх частоти.

Створення статистики за принципом "ковзаючого вікна" дозволяє обчислити суму (або іншу статистику) значень за певну кількість попередніх періодів.

У вищезазначеному прикладі ми створили новий стовпець в DataFrame, який містить ковзаючу суму за 5 попередніх періодів:

```
df['Ковзаюче 5-денне середнє'] = df.rolling(5).sum()
```

Однак, перші чотири значення нового стовпчика не мають реальних значень, оскільки для них немає достатньо даних для обчислення ковзаючої суми. Це призводить до появи значень NaN.

Для заповнення цих відсутніх даних можна використати методи заповнення, такі як ffill або bfill. Наприклад, ми можемо створити новий стовпець і заповнити відсутні значення першим існуючим значенням в цьому стовпчику:

```
df['Ковзаюче відновлене'] = df['Ковзаюче 5-денне середнє'].fillna(method='backfill')
```

Таким чином, ми отримуємо новий стовпець, де відсутні значення заповнені першими наявними значеннями, які зустрічаються після них.

Цей метод дозволяє створити статистику за ковзаючим вікном і враховує реальні дані, які доступні на момент кожного періоду.

4 ЗАСОБИ PYTHON ДЛЯ ВИКОРИСТАННЯ В ЗАДАЧАХ ОПИСОВОЇ СТАТИСТИКИ

Описова статистика - це ключовий розділ статистики, що займається першинною обробкою емпіричних даних для їхнього аналізу та розуміння. Вона включає в себе візуалізацію та числовий опис набору даних за допомогою основних статистичних показників. Основні міри центральної тенденції, такі як середнє значення, медіана, мода, дають уявлення про "типове" значення змінної, тоді як міри мінливості, такі як стандартне відхилення, мінімальне та максимальне значення, розмах, ексцес та асиметрія, описують розподіл даних.

Розподіли змінних можуть бути дискретними, коли спостереження приймають лише цілі значення, або неперервними, коли значення є плаваючими. Наприклад, це може бути кількість пакетів, що проходять мережею за певний час (дискретний), або кількість байт інформації, що проходить мережею (неперервний).

Описова статистика є першим кроком у розумінні даних і дозволяє отримати загальну картину про їхній розподіл та характеристики перед подальшим аналізом.

4.1. Основні характеристики розподілів. Центр розподілу

4.1.1. Середнє значення вибірки

Середнє значення будь якої вибірки позначається як x і визначається за формулою:

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N}$$

де xi -елементи вибірки, N - кількість елементів в вибірці.

Середнє значення вибірки легко вирахувати без звертання до будь яких

спеціальних інструментів та додаткового імпортування бібліотек. Наприклад, одна з можливих реалізацій засобами виключно базового Python виглядає так:

```
list_x=[11,9,14,9,12,9,11,11,7,7,7]
i=0 sum=0
while i < len(list_x): sum+=list_x[i] i+=1
print(sum/i )
>>>>
```

9.727272727272727

Якщо дані представляють собою масив numpy – то все стає ще простіше:

```
arr_x=np.array([11,9,14,9,12,9,11,11,7,7,7])
mn=arr_x.sum() / len(arr_x) print(mn)
```

У програмуванні для обробки та аналізу даних існують спеціалізовані бібліотеки, що містять функції для базових операцій описової статистики. Ці функції зазвичай працюють швидше і мають більшу гнучкість у порівнянні з реалізаціями, написаними на базових інструментах. Наприклад, у пакеті numpy для обчислення середнього значення використовується метод numpy.mean():

```
import numpy as np
arr = np.arange(278, dtype=float)
mn = np.mean(arr)
print(mn)  # Виведе: 138.5
```

Проте реальні дані часто містять відсутні значення (NaN), що може призвести до проблем при обчисленні статистик. Наприклад:

```
arr[10::44] = np.nan
mn = np.mean(arr)
print(mn)  # Виведе: Nan (не число)
```

Для обрахунку статистик з масивами, що містять NaN, у numpy існують спеціальні функції, які легко розпізнати - їх назви починаються з "nan". Вони враховують відсутні значення:

```
mn = np.nanmean(arr)
print(mn)  # Виведе: 138.40959409594095
```

У numpy також є функція для обчислення зваженого середнього, що корисно для різних застосувань. Зважене середнє визначається формулою:

$$\bar{x} = \frac{\sum_{i=1}^{N} w_i x_i}{\sum_{i=1}^{N} w_i}$$

Зважене середнє арифметичне використовується тоді, коли кожному елементу вибірки приділяється різна важливість у визначенні результату. Вагові коефіцієнти надаються кожному значенню, враховуючи його вагомість у загальному обчисленні. Наприклад, у фізиці це може бути середня швидкість тіла, центр маси тіла або температура суміші. У інших областях, таких як екологія чи оцінка товарів, вагові коефіцієнти використовуються для врахування різних аспектів чи досвіду.

У numpy для розрахунку зваженого середнього використовується функція average(), де параметр weights приймає значення вагових коефіцієнтів. Ось приклад використання цієї функції та порівняння результатів з середнім арифметичним:

```
import numpy as np

x = [6.0, 1, 2.5, 6, 25.0]
w = [0.1, 0.2, 0.3, 0.25, 0.15]

mean = np.mean(x)
wmean = np.average(x, weights=w)

print('Середнє арифметичне = {}, Зважене середнє = {}'.format(mean,
wmean))
# Виведе: Середнє арифметичне = 8.1, Зважене середнє = 6.8
```

Отже, середнє арифметичне та зважене середнє можуть різнитися, оскільки зважене середнє враховує вагу кожного елементу. У вказаному прикладі зважене середнє менше, оскільки менші значення мають більшу вагу за заданими ваговими коефіцієнтами.

4.1.2. Медіана вибірки

Медіана - це значення, яке розділяє набір даних навпіл за кількістю елементів менших та більших за неї. Це означає, що половина значень у вибірці менше за медіану, а половина - більше. Її можна знайти шляхом впорядкування

даних і вибору середнього значення для непарної кількості елементів або середнього арифметичного двох серединних значень для парної кількості.

У пакеті numpy для знаходження медіани використовується функція np.median():

```
import numpy as np

np.random.seed(12345)
arr = np.random.randint(100, size=80)
mn = np.median(arr)
print(mn)  # Виведе: 42.0
```

Якщо в наборі присутня парна кількість елементів, медіана дорівнює середньому арифметичному двох серединних чисел у відранжованій послідовності.

Окрім пакету numpy, в Python існує вбудований пакет statistics, який також має методи для обчислення статистичних параметрів, включаючи медіану. Цей пакет надає додаткові можливості порівняно з numpy та scipy. Наприклад, у цьому пакеті є функції median_low() та median_high(), які знаходять медіану для парної кількості значень:

```
import statistics as st
md_l = st.median_low([1, 3, 5, 8, 11, 12])
md_h = st.median_high([1, 3, 5, 8, 11, 12])
print('md_l =', md_l)  # Виведе: md_l = 5
print('md_h =', md_h)  # Виведе: md_h = 8
```

Ці функції дають можливість отримати значення, що знаходиться нижче та вище "середньої" точки для парної кількості значень.

4.1.3. Мода вибірки

Мода представляє собою значення, яке зустрічається найчастіше у вибірці даних. У пакеті numpy немає окремого методу для обчислення моди, тому за потреби можна скористатися методом з іншого пакету, такого як scipy.stats.

Використання модулю scipy.stats дозволяє отримати значення моди та кількість його входжень в вибірку.3

```
import scipy.stats as sps

data = [1, 3, 4, 4, 7, 3, 4, 5, 8, 3, 4]
print(sps.mode(data))  # Виведе: ModeResult(mode=array([4]),
count=array([4]))
```

Також функція для обчислення моди є у пакеті statistics:

```
import statistics as st

np.random.seed(123)
arr = np.random.randint(20, size=1000)
mo = st.mode(arr)
print(mo)  # Виведе: 7
```

Мода може обчислюватися не лише для числових даних, але й для категоріальних:

```
clrs = ["red", "blue", "blue", "red", "green", "red", "red"]
print(st.mode(clrs))  # Виведе: 'red'
```

Пакет statistics має навіть дві функції для пошуку моди: одна повертає одне значення моди, а інша - всі значення, якщо вони повторюються однакову кількість разів.

```
lt = [11, 9, 14, 9, 12, 9, 11, 11, 7, 7, 7]
print(st.mode(lt))  # Виведе: 11
print(st.multimode(lt))  # Виведе: [11, 9, 7]

lt = [11, 9, 14, 9, 12, 9, 11, 11, 7, 7, 7]
print(st.mode(lt))  # Виведе: 11
print(st.multimode(lt))  # Виведе: [11, 9, 7]
```

Крім того, бібліотека pandas, яка є надбудовою над numpy, також має функції для обчислення статистичних параметрів, включаючи моду.

```
import pandas as pd

np.random.seed(123)
drr1 = np.random.randint(0, 15, 1000)
numbers = pd.Series(drr1)

print('Середнє:', numbers.mean())
print('Медіана:', numbers.median())
print('Мода:', numbers.mode()[0])
```

Пакет pandas спадкує майже всі статистичні функції з numpy та scipy, що робить його потужним інструментом для аналізу даних.

4.2. Кількісні оцінки мінливості даних

Центральних метрик недостатньо для опису даних. Практично завжди потрібні метрики оцінки варіативності даних, які кількісно визначають розкид точок даних на множині можливих значень. Найпростіший показник розкиду – розмах, тобто різниця між найбільшим та найменшим значенням даних. Його можна знайти за допомогою методу np.ptp():

```
range = np.ptp(drr1) print(‘Розмах ‘,range )
>>>> Розмах 14
```

4.2.1. Квантилі та прецентилі

На Рис. 4.1 зображено взаємозв'язок між значеннями елементів набору даних (x), квантилем (Q), функцією щільності розподілу (pdf) та кумулятивною функцією розподілу (cdf). Ці характеристики важливі для аналізу даних і можуть бути обчислені за допомогою вбудованих функцій.

Для обчислення квантилей часто використовується пакет scipy.stats:

```
np.random.seed(1935)
drr2 = np.random.normal(size=100000)
print(scs.mstats.mquantiles(drr2, prob=[0.25, 0.5, 0.75]))
```

Виведе: [-0.67655052 -0.00317365 0.67523374]

Це дозволяє отримати значення квантилей для набору даних, що допомагає зрозуміти їх розподіл.

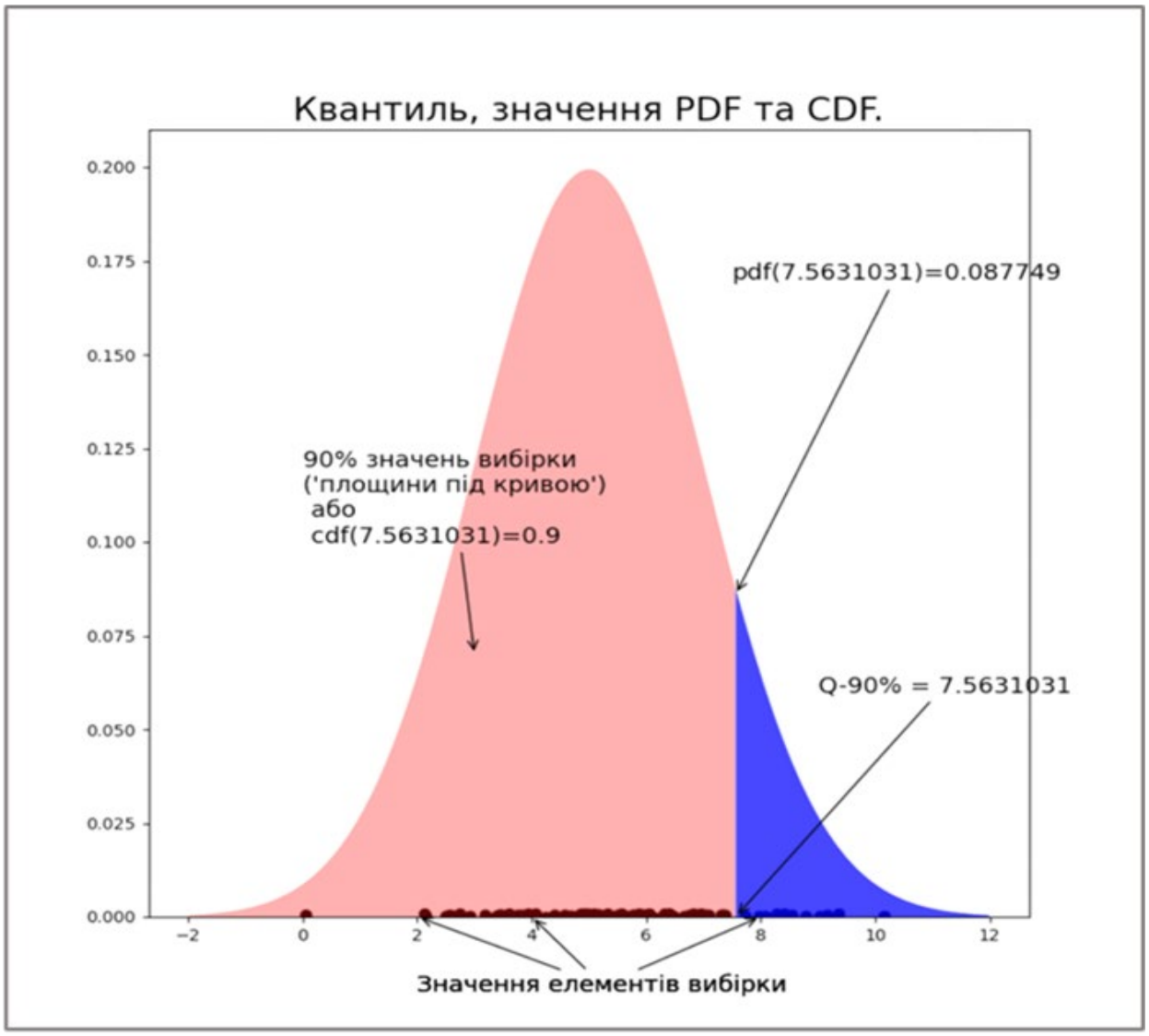

Рис. 4.1 – Основні елементи описової статистики, що використовуються для аналізу варіативногсті даних

У наведеному прикладі демонструється, що можна вирішити зворотню задачу: знайти відсоток даних у наборі, які менші за задане число, або знайти значення кумулятивної функції розподілу для заданої точки (перцентилю). Це можна зробити за допомогою функції:

print(scs.percentileofscore(drr2, 0.67523374))

Виведе: 75.0

Ця функція показує, що приблизно 75% даних у наборі менше за задане число 0.67523374.

4.2.2. Дисперсія та стандартне відхилення

Дисперсія є важливою характеристикою випадкової величини, відображаючи розкид значень навколо їх середнього. Математичне очікування визначає центр випадкової величини, а дисперсія показує, наскільки далеко точки вибірки розташовані від цього центру.

Для генеральної сукупності дисперсія обчислюється за формулою:

$$\sigma^2 = \frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N}$$

де μ - математичне очікування генеральної сукупності. Однак у практичних задачах математичне очікування зазвичай невідоме, і використовується його оцінка - середнє арифметичне.

Тому для вибіркових даних застосовується "незміщена оцінка" дисперсії:

$$s^2 = \frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N - 1}$$

де x - середнє арифметичне вибірки.

Ця оцінка зазвичай більш точна, оскільки враховується кількість ступенів свободи вибірки. Вона називається вибірковою дисперсією.

Це важливо враховувати при аналізі даних, оскільки зміщена оцінка може призвести до неточностей в оцінці розкиду значень вибірки.

5.ОХОРОНА ПРАЦІ

Небезпечні та шкідливі виробничі фактори можуть мати вплив на дослідника програмних засобів Python моделювання та системного аналізу кіберзахисту інформаційнокомунікаційних систем наступні: підвищена чи понижена температура повітря робочої зони; недостатнє освітлення робочої зони; недостатність природного освітлення; підвищений рівень шуму на робочому місці; відсутність чи нестача природнього світла; фізичні перевантаження (статичні); нервово - психічні перевантаження (перенапруга аналізаторів, емоційні навантаження).

Відповідно до визначених факторів формуємо рекомендації щодо безпечних умов праці під час дослідження програмних засобів Python моделювання та системного аналізу кіберзахисту інформаційнокомунікаційних систем.

5.1. Технічні рішення щодо безпечного виконання роботи

Однією із характерних особливостей сучасного розвитку суспільства є зростання сфер діяльності людини, в яких використовуються інформаційні технології. Широке розповсюдження отримали персональні комп’ютери. Однак їх використання загострило проблеми збереження власного та суспільного здоров'я, вимагає удосконалення існуючих та розробки нових підходів до організації робочих місць, проведення профілактичних заходів для запобігання розвитку негативних наслідків впливу ПК на здоров’я користувачів.

Відповідно до встановлених гігієнічно-санітарних вимог (ДСН 3.3.6.042-99 [11]) роботодавець зобов'язаний забезпечити в приміщеннях з ПК допустимі параметри виробничого середовища.

Робоче місце дослідника програмних засобів Python моделювання та системного аналізу кіберзахисту інформаційнокомунікаційних систем ПК повинне бути розташовані на відстані не менше 1,5 м від стіни з вікнами, від інших стін – на відстані 1м, від інших користувачів – на відстані не менше 1,5 м. Відносно вікон робоче місце доцільно розташувати таким чином, щоб природне світло падало на нього збоку, переважно зліва.

При розташуванні елементів робочого місця користувача ПК слід

враховувати:

- робочу позу користувача;
- простір для розміщення користувача;
- можливість огляду елементів робочого місця;
- розміщення документації і матеріалів, які використовуються користувачем.

Для забезпечення точного та швидкого зчитування інформації в зоні найкращого бачення площина екрана монітора повинна бути перпендикулярною нормальній лінії зору. При цьому повинна бути передбачена можливість переміщення монітора навколо вертикальної осі в межах ±30° (справа наліво) та нахилу вперед до 85° і назад до 105° з фіксацією в цьому положенні.

Клавіатура повинна бути розташована так, щоб на ній було зручно працювати двома руками. Клавіатуру слід розміщати на поверхні столу на відстані 100...300 мм від краю. Кут нахилу клавіатури до столу повинен бути в межах від 5 до 15°, зап’ястя на долонях рук повинні розташовуватись горизонтально до площини столу.

Для запобігання світлових відблисків від екрана, клавіатури в напрямку очей користувача від освітлювачів загального призначення або сонячних променів необхідно застосовувати антивідблискові сітки, спеціальні фільтри, захисні козирки, на вікнах регульовані жалюзі.

Принтер повинен бути розміщений у зручному для користувача положенні, так, що максимальна відстань від користувача до клавіш управління принтером не перевищувала довжину витягнутої руки користувача.

Конструкція робочого стола повинна забезпечувати можливість оптимального розміщення на робочій поверхні обладнання, що використовується, з врахуванням його кількості та конструктивних особливостей (розмір монітора, клавіатури, принтера, ПК та ін.) і документів, а також враховувати характер роботи, що виконується.

Згідно з вимог електробезпеки, під час проектування та експлуатації систем електропостачання для ПК необхідно дотримуватись вимог.

Лінія електромережі для живлення ПК, периферійних пристроїв ПК та устаткування для обслуговування, ремонту та налагодження ПК під час проектування виконана як окрема групова трипровідна мережа, шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів.

Категорія умов по небезпеці електротравматизму – без підвищеної небезпеки, у зв'язку з відсутністю факторів підвищеної та особливої небезпеки [12].

Для запобігання електротравмам у приміщенні здійснюються:

1) ізоляція нормально струмоведучих елементів електроустаткування відповідно з вимогами нормативів;

2) захисне заземлення із використанням природних заземлювачів;

3) систематичне проходження інструктажу з елетробезпеки.

Вимоги безпеки перед початком роботи:

1. Перевірити надійність встановлення апаратури на робочому столі.

2. Перевірити загальний стан апаратури, справність проводки електромережі, з'єднувальних шнурів, штепсельних вилок та розеток, заземлення захисного екрана.

3. Відрегулювати освітленість робочого місця.

4. Відрегулювати і зафіксувати висоту стільця (крісла), зручний для користувача нахил його спинки.

5. Увімкнути апаратуру ПК вмикачами на корпусах в такій послідовності: стабілізатор (або блок безперервного живлення), ВДТ, системний блок, принтер (якщо передбачається друкування).

6. Відрегулювати яскравість світіння екрану ВДТ, контрастність.

7. При появі несправностей комп'ютерної техніки повідомити керівника.

5.2. Технічні рішення з гігієни праці та виробничої санітарії

5.2.1 Мікроклімат

За енерговитратами робота дослідника програмних засобів Python моделювання та системного аналізу кіберзахисту інформаційнокомунікаційних систем згідно Гігієнічної класифікація праці [1] відноситься до категорії І б. Нормовані значення параметрів мікроклімату для цієї категорії наведені в табл. 5.2.1

Таблиця 5.2.1 - Нормовані параметри мікроклімату в робочій зоні

Період року	Категорія робіт	Допустимі		
		t, °C	W, %	V,м/с
Теплий	Легка І б	22-28	40-60	0,1-0,3
Холодний		20-24	75	0,2

Для забезпечення необхідних за нормативами параметрів мікроклімату здійснюються такі заходи:

1. Температура в приміщенні в холодний період року підтримується за допомогою системи центрального опалення.
2. Використовується передбачається припливна вентиляційна система.
3. Здійснюється систематичне вологе прибирання.

5.2.2. Склад повітря робочої зони

Шкідливі речовини – речовини, які при контакті з організмом людини внаслідок порушення технологічного процесу викликають професійні захворювання, виробничі травми або відхилення стану здоров’я.

Шкідливі речовини у повітря робочої зони поступають у вигляді пару, газів та пилу. Вплив на організм людини залежить від хімічного складу, розміру

(дисперсності), форми часток та їх кількості у одиниці об’єму. Найбільш небезпечний високодисперсний пил (розміром < 5 мкм), а також гострокрайовий пил. Високодисперсний пил найбільш глибоко проникає та затримується у легенях.

Забруднення повітря робочої зони регламентується граничнодопустимими концентраціями (ГДК) в мг/м3 [11]. ГДК шкідливих речовин, які знаходяться в досліджуваному приміщені, наведені в таблиці 5.2.2.

Таблиця 5.2.2 – ГДК шкідливих речовин у повітрі

Назва речовини	ГДК, мг/м^3		Клас небезпечності
	Максимально разова	Середньо добова	
Пил нетоксичний	0,5	0,15	4
Озон	0,16	0,03	1

Для забезпечення складу повітря робочої зони слід використовувати механічну вентиляцію та забезпечити регулярне прибирання.

5.2.3 Виробниче освітлення

Природне освітлення повинно бути боковим, бажано одностороннім. Для уникнення засліплюючої дії сонячних променів найкраще, коли світлові отвори (вікна) зорієнтовані на північ чи північний схід. Коефіцієнт природної освітленості (КПО) повинен бути не нижче 1,5%.

У приміщенні, де проводиться дослідження програмних засобів Python моделювання та системного аналізу кіберзахисту інформаційнокомунікаційних систем використовується штучне та природне освітлення. Норми освітленості при штучному освітленні та КПО (для III пояса світлового клімату) при природному та сумісному освітленні зазначені у таблиці 5.2.3:

Таблиця 5.2.3 - Норми освітленості в приміщенні

Характеристика зорової роботи	Найменший розмір об'єкта розрізнювання	Розряд зорової роботи	Підрозряд зорової роботи	Контраст об'єкта розрізнення з фоном	Характеристика фона	Освітленість, лк		КПО, $e_н$,%			
						Штучне освітлення		Природнє освітлення		Сумісне освітлення	
						Комбіноване	Загальне	Верхнє або верхнє	Бокове	Верхнє або верхнє	Бокове
Дуже високої точності	Від 0,15 до 0,3	II	г	великий	світлий	1000	300	7	2,5	4,2	1,5

Для забезпечення достатнього освітлення передбачені такі заходи:

1) Максимальне використання бічного природного освітлення.

2) Систематичне очищення скла від бруду – не рідше двох разів на рік.

3) Штучне освітлення в приміщенні забезпечується світильниками типу РСП08×250 (однолампові) з лампами ДРЛ-250.

5.2.4 Виробничий шум

У приміщеннях з ПК рівні звукового тиску, рівні звуку та еквівалентні рівні звуку на робочих місцях повинні відповідати вимогам ДСН 3.3.6.037-99 [10].

Рівні шуму на робочих місцях осіб, що працюють з ПК, визначені ДСанПіН 3.3.2-007-98 [9]. Допустимі значення звукового тиску під час проектування лабораторного стенду наведені в табл. 5.2.4.

Таблиця 5.2.4 - Рівень звукового тиску

Характер робіт	Допустимі рівні звукового тиску (дБ) в стандартизованих октавних смугах з середньогеометричними частотами, Гц								
	32	63	125	250	500	1000	2000	4000	8000
Постійні робочі місця в промислових приміщеннях	107	95	87	82	78	75	73	71	69

Для зниження шуму в приміщенні, необхідно:

– безпосередньо біля джерел шуму використовувати звукопоглинаючі матеріали для покриття стелі, стін;

– для боротьби з вентиляційним шумом потрібно застосовувати мало шумові вентилятори.

–

5.2.5 Виробничі випромінювання

В умовах, що розглядаються, діють електромагнітні випромінювання. Джерелами електромагнітних випромінювань являються ПК, дослідницькі установки та вимірювальні пристрої.

Рівні електромагнітного випромінювання та магнітних полів повинні відповідати вимогам. Рівні інфрачервоного випромінювання не повинні перевищувати граничних. Рівні ультрафіолетового випромінювання не повинні перевищувати допустимих [9].

Гранично допустима напруженість електростатичного поля на робочих місцях не повинна перевищувати рівнів [6].

Заходи щодо зменшення впливу на працівника електромагнітного випромінювання: оптимальна організація робочого місця, доцільне розміщення

технологічного устаткування, дотримання гігієнічно-обгрунтованих режимів праці та відпочинку, зменшення часу перебування у зоні опромінення.

5.3. Пожежна безпека

Приміщення, де здійснюється дослідження програмних засобів Python моделювання та системного аналізу кіберзахисту інформаційнокомунікаційних систем відноситься до категорії пожежної безпеки В (табл. 5.3.1).

Таблиця 5.3.1. Категорія приміщення за вибухопожежною і пожежною небезпекою

Категорія приміщення	Характеристика речовин і матеріалів, що знаходяться (обертаються) у приміщенні
В Пожежонебезпечна	Тверді горючі та важкогорючі речовини і матеріали, за умови, що приміщення, в яких вони знаходяться, не відносяться до категорій А, Б і питома пожежна навантага для твердих і рідких легкозаймистих та горючих речовин на окремих ділянках площею не менше 10 м2 кожна перевищує 180 МДж/м2.

Конструкція будівлі, де розміщено приміщення, відноситься до II-го ступеня вогнестійкості (будівля з несучою конструкцією з природних матеріалів або штучного каменю, бетону або залізобетону з застосуванням листових і плиткових негорючих матеріалів).

Згідно з правилами улаштування електроустановок, приміщення відноситься до класу П-IIа пожежної небезпеки (приміщенні, у якому знаходяться тверді горючі речовини та матеріали).

5.3.1. Технічні рішення системи запобігання пожежі

До причин, що можуть спричинити пожежу в приміщенні, відносяться:

– короткі замикання;

– перевантаження електромережі і перегріву струму несучих частин та з’єднань;

– порушення правил техніки безпеки.

При виникненні аварійних ситуацій відбувається різке виділення теплової енергії, яка може стати причиною виникнення пожежі.

Заходи щодо запобігання пожежі:

– систематична перевірка на справність струмоведучих частин обладнання;

– систематичне проведення протипожежного інструктажу;

– дотримання вимог пожежної безпеки на робочому місці.

5.3.2. Технічні рішення системи протипожежного захисту

У приміщені можливе використання системи автоматичного пожежогасіння. Також потрібно використовувати установку порошкового пожежогасіння, що використовує у якості вогнегасної речовини спеціальний порошок, який є нетоксичним. Для подачі сигналів про пожежу у приміщенні встановлено пожежну сигналізацію.

Після закінчення роботи від усіх електроприладів, а також з мереж їх живлення повинна бути відключена напруга (за винятком протипожежних та охоронних установок). Електропроводи для підключення комп'ютерів, приладів повинні прокладатися по негорючих конструктивних елементах.

Згідно норм [7] на кожні 20 м^2 площі приміщення вказаних категорії та класу пожежовибухонебезпеки та можливих класів пожеж – А, В і Е розміщується один порошковий або вуглекислотний вогнегасник з масою заряду від 3 до 5 кг. Крім того на поверсі, де знаходиться приміщення слід забезпечити наявність двох порошкових вогнегасники з масою заряду 10 кг.

Тому в приміщені буде розташовуватися два порошкових вогнегасники, які будуть розміщуватись в різних його кінцях, на висоті не більше 1,5 м від рівня підлоги до нижнього торця вогнегасника і на відстані від дверей, достатній для їх повного відчинення. Підходи до місця розташування вогнегасників мають бути завжди вільними. Для зазначення місцезнаходження вогнегасників буде встановлений вказівний знак. Знак розташовують на видних місцях на висоті 2,0 - 2,5 м від рівня підлоги.

ВИСНОВКИ

Проаналізовано сучасний стан використання часових рядів як інструменту моделювання динамічних систем. Вивчено основні методи та підходи до аналізу часових рядів, що підтвердило їхню актуальність для моніторингу шифрованого мережевого трафіку. Визначено основні завдання дослідження, необхідні для успішної обробки та аналізу даних часових рядів.

Вивчено програмні засоби Python та модулі розширення для роботи з часом та датами, включаючи модулі time, datetime, бібліотеки numpy та pandas. Описано особливості використання цих інструментів для аналізу та обробки часових рядів.

Розглянуто засоби Python для використання в задачах описової статистики, зокрема для обчислення основних характеристик розподілів, таких як середнє значення та медіана вибірки. Показано приклади обробки даних часового ряду, отриманого при моніторингу мережевого трафіку.

Проаналізовано та обґрунтовано вибір інструментів для реалізації задач обробки та аналізу даних часових рядів. Для цього обрано наступні інструменти: Python, бібліотеки numpy, pandas, та scipy. Розглянуто приклади використання цих інструментів для аналізу реальних даних.

Проведено тестування розроблених методів та інструментів для аналізу часових рядів на основі даних мережевого трафіку. Перевірено коректність роботи запропонованих алгоритмів та зроблено висновки щодо їхньої ефективності

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Atwan T. Time Series Analysis with Python Cookbook / Atwan Tarek. – Birmingham, UK : Packt Publishing, Limited, 2022. – 630 p. ISBN: 978-1-80107-554-1.

2. Auffarth B. Machine Learning for Time-Series with Python: Forecast, predict, and detect anomalies with state-of-the-art machine learning methods / Auffarth Ben. – Birmingham, UK : Packt Publishing, Limited, 2021. – 371 p. ISBN 978-1-80181-962-6.

3. Brownlee J. Introduction to Time Series Forecasting with Python: How to Prepare Data and Develop Models to Predict the Future Series / Brownlee Jason. – UK : Machine Learning Mastery, 2020. – 365 p.

4. Haslwanter T. An Introduction to Statistics with Python / Haslwanter Thomas. – Switzerland : Springer International Publishing AG, 2016. – 285 p. ISBN 978-3-319-28315.

5. Johansson R. Numerical Python: Scientific Computing and Data Science Applications with Numpy, SciPy and Matplotlib / Johansson Robert. – New York, USA : Apress Media LLC, 2018. – 700p. ISBN 978-1-4842-4245-2.

6. Lazzeri F. Machine Learning for Time Series Forecasting with Python / Lazzeri Francesca. – Indianapolis, Indiana, USA : John Wiley & Sons, Inc., Indianapolis, Indiana, 2021. – 216 p. ISBN: 978-1-119-68236-3.

7. Mather B. Time Series with Python: How to Implement Time Series Analysis and Forecasting Using Python / Mather Bob. – USA : Publisher: Bob Mather, 2020. – 222 p. ISBN: 978-0-648-78308-4.

8. Peixeiro M. Time Series Forecasting in Python / Peixeiro Marco. – Shelter Island, NY, USA : Manning Publications Co., 2022. – 458 p. ISBN: 978-1-617-29988-9.

9. Vishwas B. Hands-on Time Series Analysis with Python: From Basics to Bleeding Edge Techniques / Vishwas B., Patel A. – New York, USA : Apress Media LLC, 2020. – 420 p. ISBN: 978-1-4842-5992-4.

10.Гігієнічна класифікація праці (за показниками шкідливості і небезпеки факторів виробничого середовища від 12.08.1986 № 4137-86. - [Електронний ресурс] - Режим доступу: http://zakon4.rada.gov.ua/laws/show/v4137400-86

11.ДСТУ OHSAS 18002:2015. Системи управління гігієною та безпекою праці. Основні принципи виконання вимог OHSAS 18001:2007 (OHSAS

18002:2008, IDT). К. : ГП «УкрНИУЦ», 2016. 21 с.

12.ДСНіП3.3.6.096-2002. Державні санітарні норми і правила при роботі з джерелами електромагнітних полів. URL: http://zakon2.rada.gov.ua/laws/show/z0203-03

13.НПАОП 40.1-1.32-01 Правила будови електроустановок. Електрообла 1 днання спеціальних установок (Правила устройства электроустановок. Элект 1 рооборудование специальных установок). URL: https://zakon.rada.gov.ua/ rada/show/v0272203-01#Text

14.ДСТУ EN 62305:2012 Блискавкозахист (європейський стандарт IEC 62305:2010). URL: https://tdsb.com.ua/ru/dstu-en-62305/

15.ДСТУБ В.2.5-82:2016. Електробезпека в будівлях і спорудах. Вимоги до захисних заходів від ураження електричним струмом. К. : ДП «УкрНДНЦ», 2016. 109 с

16. Наказ Міністерства внутрішніх справ України «Про затвердження Пра 1 вил експлуатації та типових норм належності вогнегасників». URL: https://zakon.rada.gov.ua/laws/show/z0225-18#Text.

17.ДБН В.1.1-7:2016 Пожежна безпека об`єктів будівництва. Загальні ви1моги. URL: http://www.poliplast.ua/doc/dbn _v.1.1-7-2002.pdf

18.НПАОП 0.00-7.15-18 Вимоги щодо безпеки та захисту здоров'я працiв 1 ників під час роботи з екранними пристроями. URL: http://sop.zp.ua/norm_npaop _0 _ 00-7_ 15-18_01_ua.php

19.ДСН 3.3.6.037-99 Санітарні норми виробничого шуму, ультразвуку та інфразвуку. - [Електронний ресурс] - Режим доступу: http://document.ua/sanitarni-normi-virobnichogo-shumu-ultrazvuku-ta-infrazvuku-nor4878.html

20.ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих приміщень. - [Електронний ресурс] - Режим доступу: http://mozdocs.kiev.ua/view.php?id=1972

21.Правила улаштування електроустановок - [Електронний ресурс] - Режим доступу: http://www.energiy.com.ua/PUE.html

22.СН N 4557-88 Санитарные нормы ультрафиолетового излучения в производственных помещениях - [Електронний ресурс] - Режим доступу: http://www.znaytovar.ru/gost/2/455788_Sanitarnye_normy_ultraf.html

Додаток А
(обов'язковий)

ІЛЮСТРАЦІЙНА ЧАСТИНА

Програмні засоби Python моделювання та системного аналізу кіберзахисту інформаційно-комунікаційних систем

(назва бакалаврської дипломної роботи)

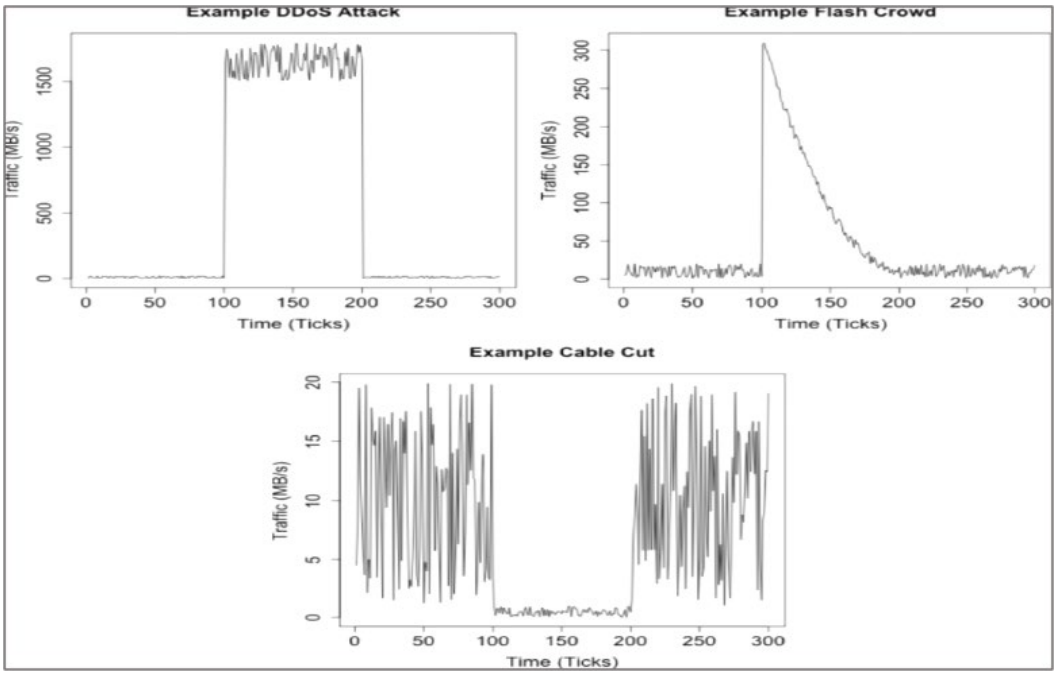

Рисунок А.1– Приклади аномалій в даних моніторингу мережевого трафіку.

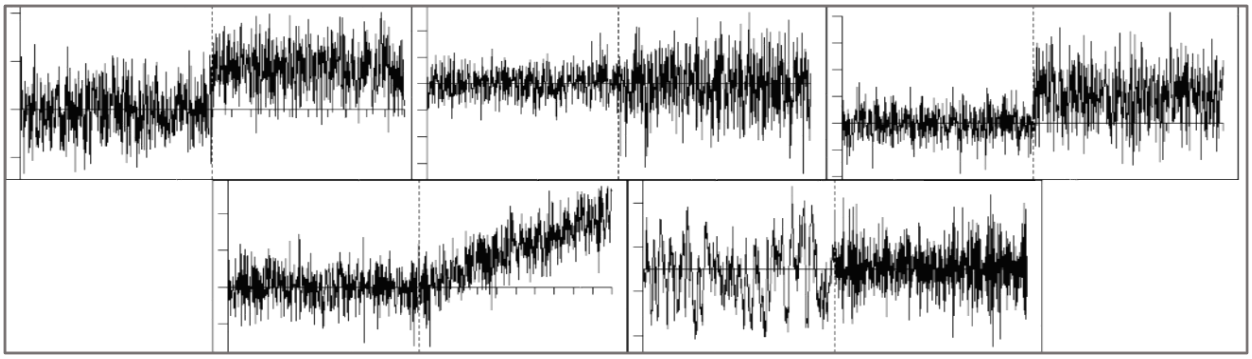

Рисунок А.2 – Деякі можливі прояви аномалій в часових рядах

Рисунок А.3 – Деякі можливі прояви аномалій в часових рядах

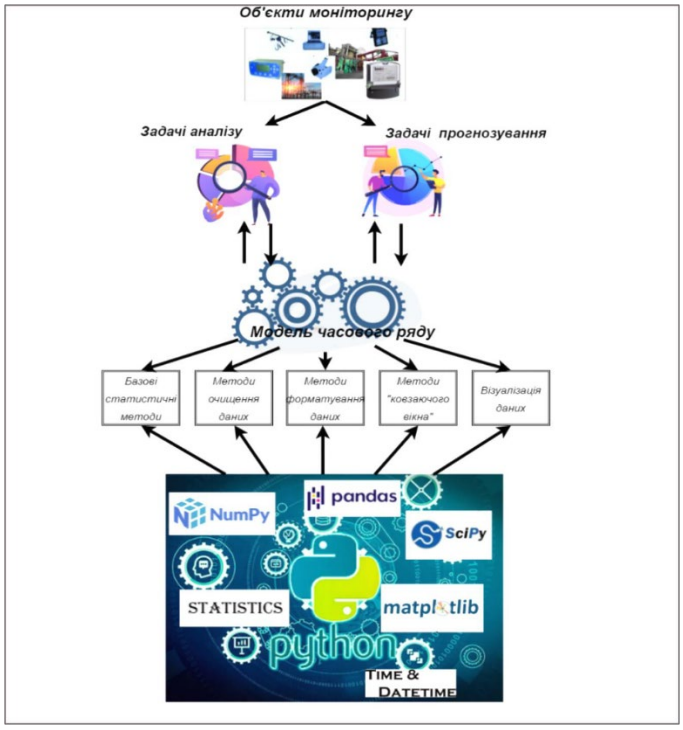

Рисунок А.4 – Узагальнюючий фреймворк часових рядів

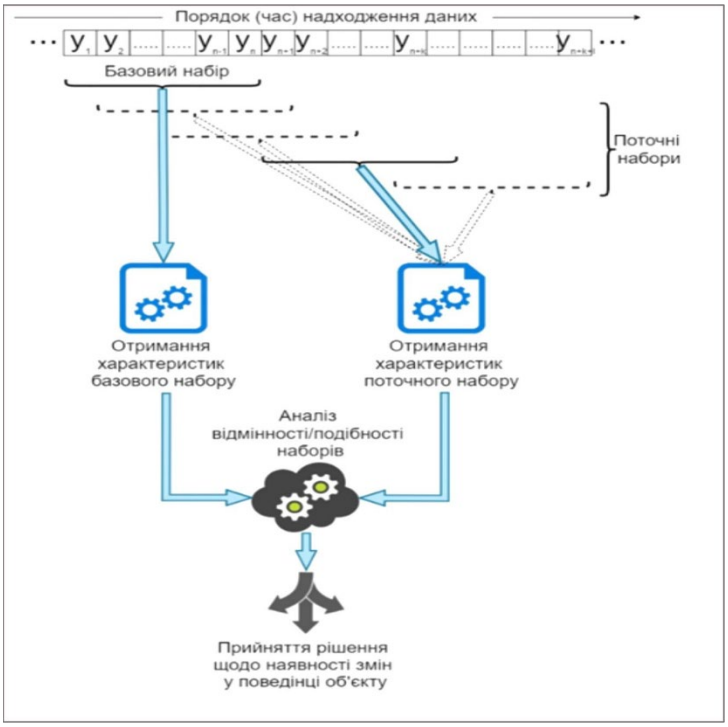

Рисунок А.5 – Алгоритм CPD автокореляції

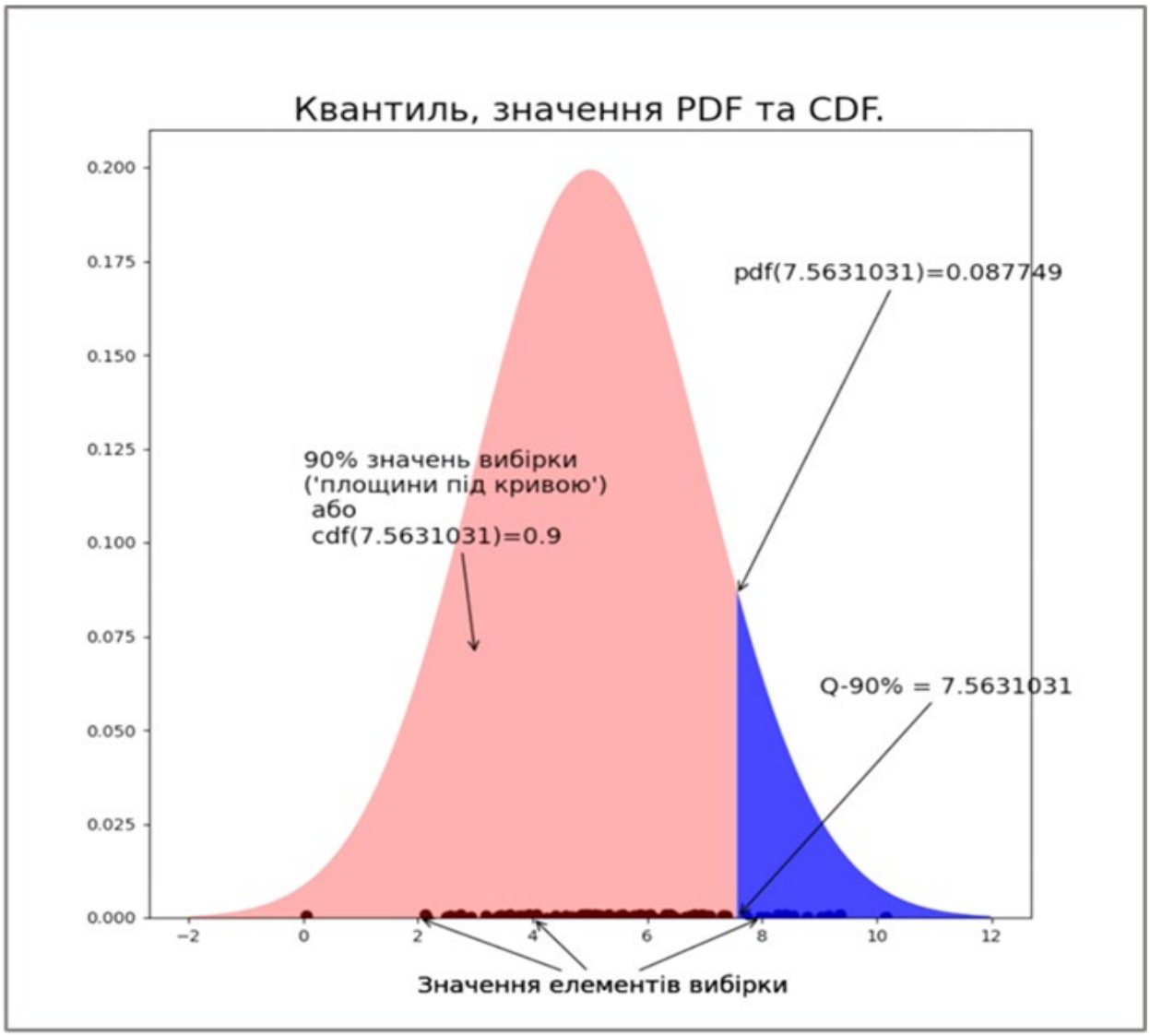

Рисунок А.6 – Основні елементи описової статистики, що використовуються для аналізу варіативногсті даних

Додаток Б

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: «Програмні засоби Python моделювання та системного аналізу кіберзахисту інформаційно-комунікаційних систем»

Тип роботи: Бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра інфокомунікаційних систем і технологій, факультет інформаційних електроннихсистем
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 71% Схожість 29%

Аналіз звіту подібності (відмітити потрібне):

,

■ 1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

☐ 2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

☐ 3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа відповідальна за перевірку _____ Васильківський М.В.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Тригубенко Я.Ю..
(підпис) (прізвище, ініціали)

Керівник роботи _____ Бортник С.Г.
(підпис) (прізвище, ініціали)