МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ОБЕРТИНСЬКИЙ ОЛЕКСАНДР ОЛЕГОВИЧ

Допускається до захисту:
в.о. завідувача кафедри
інформаційних технологій
канд. техн. наук, доцент
_____ О. В. Зелінська
«____»_____ 20__ р.

ІНФОРМАЦІЙНА СИСТЕМА АНАЛІЗУ ДАНИХ ОПЛАТИ ПРАЦІ

Спеціальність 122 Комп’ютерні науки

Кваліфікаційна (магістерська) робота

Науковий керівник:
Т. В. Січко, доцент кафедри
інформаційних технологій,
канд. техн. наук, доцент
_____

Оцінка: _____ / _____ / _____
(бали/за шкалою ЄКТС/за національною шкалою)

Голова ЕК: _____

Вінниця — 2024

АНОТАЦІЯ

Обертинський О. О. Розробка інформаційної системи аналізу даних оплати праці. Спеціальність 122 «Комп'ютерні науки». Освітня програма «Комп'ютерні технології обробки даних (Data science)». Донецький національний університет імені Василя Стуса, Вінниця, 2024.

У кваліфікаційній (магістерській) роботі було досліджено: теоретичні аспекти оплати праці, методи для обробки даних, бібліотеки мов програмування для виконання завдання. Була побудована програмна архітектура системи та описана її розробка.

Ключові слова: інформаційна системи аналізу даних оплати праці, DataSet, аномальні виміри, прогнозування, Python.

74 ст., 22 рис., 50 джерел.

Obertynskyi O.O. Development of an information system for the analysis of salary data. Specialty 122 «Computer Science», Educational program «Data science». Vasyl Stus Donetsk National University, Vinnytsia, 2024.

In the qualification (master's) thesis, the following were investigated: theoretical aspects of labor remuneration, data processing methods, language programming libraries for performing tasks. A software architectural system was built and its development described.

Key words: information systems for the analysis of wage data, DataSet, anomalous measurements, forecasting, Python.

Page 74. Fig. 22. 50 Src.

ЗМІСТ

ВСТУП

Актуальність роботи. Аналіз оплати праці є важливим інструментом для забезпечення рівності серед працівників і усунення будь-яких нерівностей. Оскільки добре розроблена система оплати праці може бути важливою частиною залучення та утримання талановитих і кваліфікованих працівників, вона також сприяє конкурентоспроможності роботодавців на ринку праці. Оптимальна оплата праці є важливим елементом стратегії мотивації працівників та забезпечення їхнього задоволення від роботи. Аналіз оплати праці дозволяє оцінити, наскільки ефективно використовуються витрати на оплату праці та чи відповідають вони ринковим стандартам. Аналіз даних також допомагає компаніям розробляти стратегії мотивації, спрямовані на підвищення продуктивності працівників. Крім того, він допомагає виявити тренди в оплаті праці, що дозволяє компаніям адаптувати свою політику до змін на ринку праці.

Мета – розробка інформаційної системи аналізу даних оплати праці для прогнозування змін у заробітній платі тестувальників.

Об’єкт дослідження - процес виявлення аномальних вимірів, фільтрації, та виявлення динаміки змін заробітної плати у тестувальників.

Предмет дослідження - методи аналізу даних, які можуть використовуватись для детекції та очищення, фільтрації даних, а також методи для прогнозування динаміки змін заробітної плати.

Завдання:

- проаналізувати способи аналізу оплати праці;
- обрати середовище створення інформаційної системи;
- створення архітектури інформаційної системи;
- обрання методів для детекцій та очищення від аномальних вимірів;
- створити прогноз щодо оплати праці.

РОЗДІЛ 1
ТЕОРЕТИЧНІ АСПЕКТИ АНАЛІЗУ ОПЛАТИ ПРАЦІ

1.1 Аналіз оплати праці

Під час проведення дослідження з теми дипломної роботи було опрацьовано програмні рішення BAS, такі як: BAS Бухгалтерія, BAS Бухгалтерія (базова), BAS Бухгалтерія КОРП, BAS Комплексне управління підприємством, BAS Малий бізнес, та програмою KBS Облік бюджетної установи. BAS та KBS програмні рішення аналізу праці для автоматизації процесів бізнесу. Працюючи з цими програмами було розглянуто як вони збирають, зберігають та обробляють дані. Програмні рішення можуть бути налаштовані для збору таких даних як: години роботи, ставки оплати утримання податків, виплати, військовій збір та інше. Ці програмні рішення можуть автоматично вираховувати суми оплати праці для різноманітних категорій робітників на основі введених даних. В нашій державі оплата праці регулюється Законом України «Про оплату праці», а також багатьма іншими законами і нормативними актами. Система оплати праці складна, часто відбуваються зміни в законодавстві, тому бухгалтерам з оплати праці доводиться постійно переглядати зміни у законодавстві. Те що в тому місяці ще було актуально в новому вже може бути застарілим. Бухгалтерам постійно необхідно відвідувати семінари та конференції для пояснень законодавства, або додатково брати платну консультацію у інших бухгалтерів. Розробники цих програм майже кожен місяць випускають оновлення, як на законодавчій основі так і на програмній. Система оплати праці - чинний на підприємстві організаційно-економічний механізм взаємозв’язку між показниками, що характеризують міру (норму) праці та міру її оплати відповідно до фактично досягнутих (щодо норм) результатів праці, тарифних умов оплати праці та погодженою між працівником і роботодавцем ціною робочої сили [1]. Типовий розрахунок заробітної плати на підприємстві включає наступні процедури:

Нарахування в розрахунковій відомості:

- Заробітна плата: згідно штатного розкладу, трудових договорів та табеля обліку робочого часу при погодинній формі оплати праці (з урахуванням державних гарантій при роботі у вихідні та святкові дні, в ненормований час, в нічний час і інших).
- Розрахунок щомісячної індексації заробітної плати.
- Розрахунок премій, надбавок, компенсацій, пільг, понаднормових, відпускних.
- Розрахунок при звільненні та скороченні.
- Облік виплат за рахунок фондів соціального страхування (лікарняні, відпустка по вагітності та пологах, путівки і т.д.).
- Розрахунок медичного страхування, страхування життя.
- Інші нарахування, передбачені законодавством України та колективним договором.
- Розрахунок винагород за договорами цивільно-правового характеру.

Утримання в розрахунковій відомості:

- За виконавчими документами (стягнення аліментів на утримання неповнолітніх дітей або непрацездатних батьків, відшкодування матеріального збитку та т.ін.).
- За заявами працівників (профспілкові внески, в недержавні пенсійні фонди та т.ін.).
- Розрахунок інших утримань, в тому числі згідно з правилами компанії - перевищення лімітів по мобільних телефонах, позики, перевищення добових і т.ін.

Розрахунок податків і зборів, що утримуються з заробітної плати, відповідно до законодавства України:

- Податку на доходи фізичних осіб у відповідності з Податковим Кодексом України (раніше - прибутковий податок з громадян).
- Військового збору.

Нарахування у бухгалтерському обліку обов'язкових резервів відповідно до ПСБО (МСФЗ):

- Резерву відпусток, а також пов'язаної суми ЄСВ.
- Резерву річного бонусу («тринадцятої» зарплати, або премії за підсумками року).

Крім розрахунку заробітної плати персоналу, підприємства повинні зробити відповідні нарахування ЄСВ (єдиного соціального внеску), виходячи з граничної суми у розрізі кожного працівника [2]. Наведемо приклад розрахунку зарплати на українському підприємстві за 2023 рік. Наприклад якщо працівнику підприємства за його роботу нараховано 150 000 гривень необхідно розрахувати ПДФО та військовий збір. Податок на прибуток або на доходи фізичних осіб - це податок, що стягується з фізичних або юридичних осіб (платників податків), який змінюється залежно від відповідного доходу або прибутку (оподатковуваного доходу). Податок на прибуток, як правило, обчислюється як добуток податкової ставки, зменшений на оподатковуваний прибуток. Ставки оподаткування можуть різнитися залежно від типу або характеристик платника податків [3]. Розрахунок заробітної плати вираховується за формулою:

Заробітна плата – ПДФО – військовий збір = Зарплата до виплати.

ПДФО складає 18%, а це означає що потрібно 150 000 грн * 18% = 27 000 грн. Військовій збір складає 1,5% і вираховується він аналогічно з ПДФО, 150 000 грн * 1,5% = 2 250 грн, отже за формулою розрахунку заробітної плати ми отримуємо 150 000(Заробітна плата) – 27 000(ПДФО) – 2 250(Військовій збір) = 120 750

Ще один додатковий функціонал цих двох програм – створення звітів, або різноманітних аналітичних документів, які за необхідністю допомагають в аналізі оплати праці. Крім звітів про витрати на оплату праці, звітів за різними періодами часу, аналітичні звіти за підрозділами можна створювати різноманітні графіки та поєднувати їх за необхідністю. Також ці програми можуть бути інтегровані в інші системи, такими як системи управління

людськими ресурсами (HRM), банківські системи які надають можливість вивантажити дані з банку, або програми та обмінятись цією інформацією.

1.2 Інформаційна система

Інформаційна система — взаємозалежна сукупність засобів, методів, які використовуються для зберігання, обробки й подання даних, відомостей з метою вирішення користувачем окреслених завдань [4]. Інформаційні системи повинні мати здатність автоматизувати процеси обробки інформації та покращувати процеси прийняття рішень.

Процеси, що відповідають за працездатність будь-якої інформаційної системи зображено на рис 1.1.

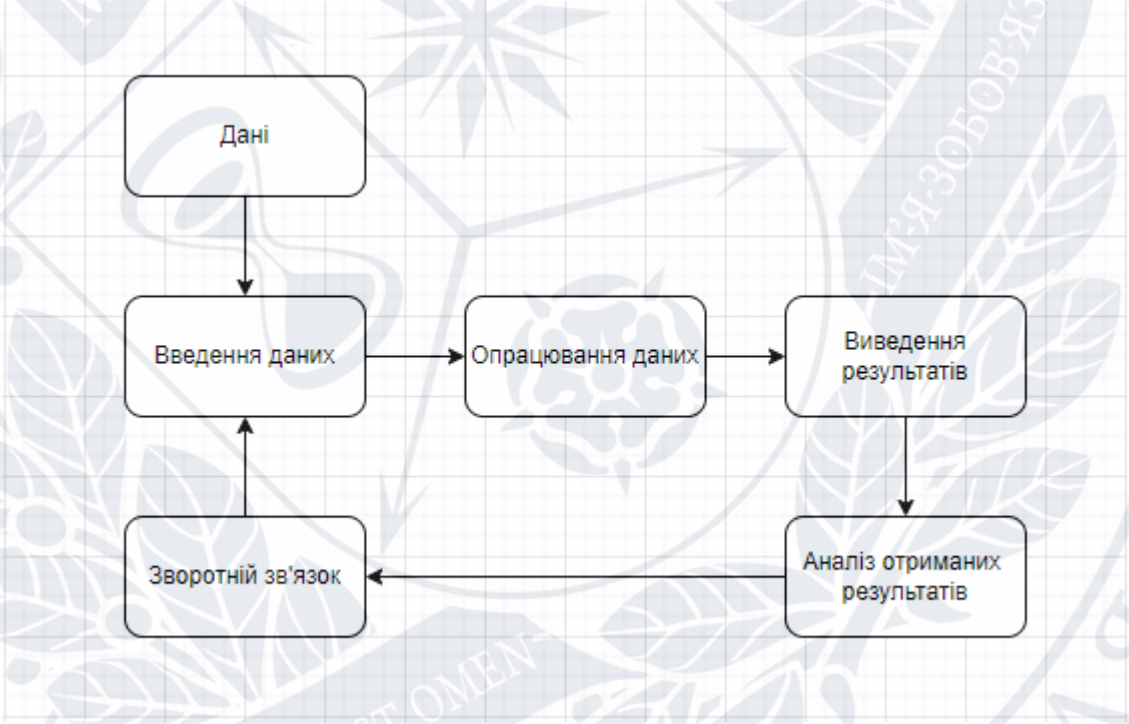

Рисунок 1.1 – Процеси інформаційної системи

Спочатку отриманні дані необхідно ввести в систему, опрацювати вхідні дані, вивести опрацьовані дані, або передати їх в іншу інформаційну систему. Аналіз отриманих результатів, опрацьовані результати якими обробляються вхідні дані.

Декомпозиція - науковий метод, що використовує структуру завдання і дозволяє замінити вирішення одного великого завдання рішенням серії менших завдань, нехай і взаємопов'язаних, але більш простих. Декомпозиція, як процес деталізації, дозволяє розглядати будь-яку досліджувану систему як складну, що складається з окремих взаємопов'язаних підсистем, які, в свою чергу, також можуть бути розділеними на частини. Як системи можуть виступати не тільки матеріальні об'єкти, а й процеси, явища і поняття [5]. Кожна з цих декомпозицій описує систему з певного боку та на різних рівнях деталізації.

Декомпозиція інформаційної системи зображена на рис 1.2.

Рисунок 1.2 - Декомпозиція інформаційної системи

Розглянемо кожну з цих підсистем більш детально. Всі елементи, пов’язані з керуванням, збором, зберіганням, обробкою та передачею інформації в системі, включаються в інформаційне забезпечення, яке є важливою частиною інформаційних систем. Його роль відповідає за надання належної, точної та своєчасної інформації, необхідної для прийняття рішень і функціонування системи. Інформаційне забезпечення дозволяє ефективно

керувати даними, захистити їх, аналізувати та використовувати їх для досягнення поставлених перед собою цілей.

Інформаційне забезпечення є важливою складовою інформаційних систем і включає наступні основні елементи:

- Збір інформації – описує процес отримання даних з різних джерел, таких як: користувачі, сенсори та бази даних. Збір інформації може включати ручне введення даних або автоматизовані процеси, які виконуються спеціалізованими системами.
- Збереження інформації – відповідає за організацію та зберігання зібраних даних для подальшого використання. Створення та управління базами даних, файловими системами, архівами та іншими засобами зберігання даних можуть бути включені до його складу.
- Обробка інформації – включає аналіз і перетворення зібраної інформації для отримання результатів. Застосування закономірностей, прогнозування, класифікації та виявлення даних тощо. Обробка може включати використання алгоритмів, моделей, методів статистики або штучного інтелекту.
- Передача інформації – вимагає передачі обробленої інформації користувачам або іншим системам. Використання доступу до інформації може включати використання мережевих технологій, протоколів зв’язку, електронної пошти, веб-сервісів або інших засобів комунікації;
- Управління інформацією – передбачає управління всіма аспектами інформаційного забезпечення. Він включає управління доступом до даних, забезпечення безпеки та захисту даних, контроль якості даних, резервне копіювання та відновлення даних і стратегічне планування розвитку інформаційних систем.
- Метадані та індексація – містить описову інформацію, яка розкриває характеристики та властивості основних даних. Це може включати назву, автора, дату створення, формат файлу, розмір, тип даних, мову,

контекст тощо. Описова інформація допомагає користувачам швидше зрозуміти, що представляють собою дані та як їх застосувати.

Метадані описують, як дані структуровані та організовані. Вони можуть показувати поля, атрибути, відношення, таблиці та інші структурні елементи даних. Це допомагає користувачам зрозуміти взаємозв'язки та організацію даних. Метадані можуть розповісти про значення та семантику даних. Вони здатні пояснити значення даних, їхнє призначення та контекст їх використання. Це допомагає користувачам розуміти та використовувати інформацію правильно. Крім того, метадані можуть містити контекстну інформацію щодо використання даних. Вони можуть містити правила доступу, права власності, ліцензії, обмеження використання, часові межі, джерела даних тощо. Це допомагає користувачам використовувати дані згідно з політикою та правилами. Індексація — це процес створення індексів або ключів для швидкого пошуку та доступу до даних у системі.

У цьому процесі використовується аналіз даних і побудова спеціальних структур даних, щоб швидко знайти відповідні дані. Індекс - це структура даних, яка відображає місце розташування запису та ключ у базі даних або іншій структурі даних. Він забезпечує ефективну організацію даних, щоб забезпечити швидкий доступ до них. Індекси можуть бути розроблені для різних полів або характеристик даних, що робить пошук і фільтрацію ефективними та швидкими. Індексація допомагає покращити продуктивність інформаційних систем шляхом забезпечення швидкого доступу до даних. Вона дозволяє виконувати операції пошуку, сортування, фільтрації та з'єднання даних у значно більш ефективний спосіб. Індексація також забезпечує підтримку індексованих структур даних, таких як бази даних, пошукові системи, веб-сторінки тощо. Існує кілька типів індексів, включаючи бінарні дерева, хеш-таблиці, сортовані послідовності та інші. У кожного типу індексу є переваги та недоліки, і рішення щодо того, який тип індексу використовувати, залежить від вимог до системи та типу даних, що індексуються.

Технічне забезпечення інформаційних систем включає мережеві технології, апаратне та програмне обладнання та інші технічні ресурси, необхідні для ефективного функціонування системи. Це дуже важливо, оскільки продуктивність, надійність і здатність інформаційної системи виконувати свої функції значною мірою залежать від якості технічного забезпечення, яке вона використовує.

- Апаратне забезпечення: Це фізичні частини системи, які включають комп'ютери, сервери, зберігання даних, пристрої мережі та периферійні пристрої. Якість апаратного забезпечення впливає на масштабованість, надійність, швидкодію обробки даних та інші характеристики системи.
- Мережеві технології: Це компоненти, які забезпечують зв'язок і обмін даними між різними частинами системи. Це можуть бути протоколи, мережеві кабелі, комутатори та маршрутизатори. Швидкість передачі даних, стабільність з'єднання та безпеку мережі залежать від якості мережевих технологій.
- Інші технічні ресурси: Це додаткові компоненти, такі як резервне живлення, системи охолодження, фізична безпека приміщень та заходи з резервного копіювання даних. Ці ресурси забезпечують безперебійну роботу системи та захищають дані від втрати.

Програмне забезпечення інформаційних систем складається з набору програм і процедур, які забезпечують роботу та функціонування системи. Включає різні програми, операційні системи, системи управління базами даних та інше програмне забезпечення. Програмне забезпечення є важливою частиною інформаційних систем і відповідає за керування та координацію різних частин системи. Основні завдання, пов'язані з керуванням ресурсами комп'ютера, включають планування задач, введення-виведення та управління пам'яттю. Системи управління базами даних забезпечують зберігання, організацію та доступ до великих обсягів даних. Програмне забезпечення виконує певні функції в інформаційних системах. Вони можуть бути розроблені для автоматизації бізнес-процесів, обробки даних, аналізу

інформації, спілкування тощо. Різні програми можуть бути розроблені відповідно до потреб користувачів і потреб системи. Програмне забезпечення є важливим компонентом, який забезпечує функціональність, продуктивність і надійність інформаційної системи. Завдяки стабільності, безпеці, ефективності та здатності задовольняти потреби користувачів можна визначити якість програмного забезпечення. Для успішної роботи інформаційної системи необхідно правильно вибрати та налагодити програмне забезпечення. Математичне забезпечення в інформаційних системах охоплює використання математичних методів, алгоритмів та моделей для вирішення завдань обробки даних і розв'язання різних викликів. Це забезпечення відіграє важливу роль у виконанні завдань, таких як аналіз даних, оптимізація процесів, математичне моделювання, прогнозування та інші області.

Математичні методи і алгоритми використовуються для обробки даних і знаходження корисної інформації з великих обсягів даних. Вони допомагають виявляти закономірності, залежності та тренди в даних, що дозволяє зробити аналітичні висновки та прийняти обґрунтовані рішення. Математичні моделі дозволяють створювати абстрактні моделі реальних ситуацій або процесів. Вони дозволяють використовувати математичні рівні та відношення для опису системи, що дозволяє проводити аналіз, прогнозування та оптимізацію. Розв’язання оптимізаційних задач, у яких потрібно знайти оптимальні значення для певних параметрів або функцій, також вимагає використання математичного забезпечення. Пошук мінімуму або максимуму функції, враховуючи обмеження та умови, може бути частиною цього процесу. Оскільки вони пропонують аналітичні та прогностичні можливості та допомагають у вирішенні складних завдань, ці математичні забезпечення є життєво важливими компонентами інформаційних систем. Інформаційні системи можна зробити більш ефективними та потужними за допомогою використання математичних методів, алгоритмів і моделей.

Організаційне забезпечення інформаційних систем включає структуру, процеси та стратегії управління, спрямовані на оптимальне використання ресурсів системи та її ефективну роботу. Цей компонент забезпечення є важливим для координації технічного, програмного та інформаційного обладнання, а також для забезпечення того, щоб інформаційна система відповідала стратегічним цілям та потребам організації. Організаційна структура визначає, як розподіляються обов'язки, ролі та функції в інформаційній системі. Це може включати призначення команд та визначення ланцюга командування, а також встановлення процедур та політик для управління системою. Процеси управління включають методи та процедури, які використовуються для контролю, планування, впровадження та моніторингу інформаційної системи. Це може охоплювати процеси збору та аналізу вимог користувачів, розробки та впровадження програмного забезпечення, забезпечення безпеки та здійснення резервного копіювання даних. Стратегії управління визначають загальні цілі, плани та напрямки розвитку інформаційної системи відповідно до потреб організації. Це може включати розробку довгострокових планів, прийняття рішень щодо розподілу ресурсів та інвестицій в інформаційну систему, а також визначення політик безпеки та конфіденційності даних. Організаційне забезпечення допомагає досягти стратегічних цілей організації, сприяє ефективному функціонуванню інформаційної системи та забезпечує взаємодію між різними її компонентами. Це включає правильне вирішення завдань управління, організацію комунікаційного ланцюга та визначення ролей і відповідальності в інформаційній системі.

Правове забезпечення інформаційних систем включає визначення та дотримання правил, норм і законів, що регулюють використання та обробку інформації. Ця частина є важливою для забезпечення конфіденційності, цілісності та доступності даних, а також для врахування прав і обов'язків учасників інформаційного процесу. Правове забезпечення включає виконання законів, таких як авторські права, кібербезпека та захист персональних даних,

серед інших. Воно вимагає, щоб організації та користувачі інформаційних систем дотримувалися правил щодо збору, зберігання, передачі та використання інформації. Правове забезпечення також включає в себе встановлення політик та процедур, спрямованих на забезпечення відповідності законодавчих вимог. Це може включати в себе впровадження правил конфіденційності даних, систем контролю доступу, методів ідентифікації та аутентифікації, а також використання шифрування та інших технологій для захисту даних. Додатково, правове забезпечення включає укладання угод, умови використання та ліцензування програмного забезпечення, а також захист прав інтелектуальної власності. Забезпечення дотримання законодавства, захисту прав і інтересів учасників інформаційних процесів і збереження довіри до інформаційних систем є основними цілями правового забезпечення інформаційних систем. Врахування правових аспектів допомагає забезпечити етичне та законне використання інформації та зменшити ризики пов'язані з порушенням законодавства.

В інформаційних системах лінгвістичне забезпечення включає елементи, пов’язані з мовою та комунікацією, і відіграє важливу роль у забезпеченні ефективної взаємодії між користувачами та системою, а також у забезпеченні того, щоб усі лінгвістичні елементи були зрозумілими та точними. Розробка та використання мовних моделей, алгоритмів та інструментів для обробки мови є частиною лінгвістичного забезпечення, яке дозволяє системам розпізнавати, розуміти та створювати людську мову. Синтаксичний і семантичний аналіз, машинний переклад, розпізнавання мовлення, автоматична генерація тексту та інші лінгвістичні функції можуть бути частиною цього. Крім того, лінгвістичне забезпечення включає створення та підтримку мовних ресурсів, таких як тезауруси, граматичні правила, словники та інші лінгвістичні бази даних. Цей набір ресурсів забезпечує точність і якість лінгвістичних операцій, а також допомагає системі розуміти та обробляти мову. Культурні та соціальні аспекти мови, такі як мовні варіанти, жаргони, діалекти та інші лінгвістичні відмінності, є частиною лінгвістичного забезпечення. Це допомагає

забезпечити адаптацію системи до різних мовних і культурних контекстів, що полегшує взаємодію користувачів з системою. Загалом, використання лінгвістики в інформаційних системах допомагає покращити комунікацію, зробити повідомлення більш точними та зрозумілими, зробити взаємодію більш зручною та допомогти людям краще зрозуміти один одного з різних культур.

1.3 Постановка задачі

Інформаційна система аналізу даних оплати праці є комплексною системою, яка дозволяє збирати, обробляти і аналізувати дані про оплату праці в Україні за спеціальністю QA (Тестувальник). Інформаційна система аналізу даних оплати праці має включати наступні етапи:

1. Збір даних:

Система повинна мати можливість збирати дані про оплату праці для спеціальності QA. Ці дані можуть бути зібрані з різних джерел, таких як: файли з розширенням .xls або .csv, бази даних, API тощо.

2. Обробка даних:

Після збору даних система повинна проводити їх обробку. Це включає перевірку та валідацію даних, наприклад, перевірку на правильний формат чисел та відсутність помилок. Система повинна відкидати недійсні або неправильні дані та забезпечувати правильну структуру даних для подальшого аналізу.

3. Розрахунок статистичних характеристик:

Система має забезпечувати можливість розрахунку різних статистичних характеристик для оплати праці QA спеціалістів. Наприклад, система може розраховувати математичне сподівання (середнє значення), дисперсію, стандартне квадратичне відхилення та інші показники, що допомагають зрозуміти розподіл та варіацію оплати праці.

4. Методи детекції та очищення від аномальних вимірів, фільтрація, згладжування та прогнозування:

Система повинна містити методи для виявлення аномальних значень в даних оплати праці та їх відкидання або обробки. Наприклад, можуть бути застосовані методи виявлення викидів або інші статистичні методи для виявлення аномалій. Також система може включати функціонал для фільтрації та згладжування даних, а також для прогнозування майбутніх значень оплати праці на основі наявних даних.

5. Візуалізація результатів:

Результати аналізу оплати праці QA спеціалістів мають бути візуалізовані у вигляді графіків, діаграм або інших графічних зображень. Це допоможе краще розуміти дані та виявляти залежності та тенденції. Система повинна мати можливість побудови різноманітних графічних представлень даних, які дозволять користувачам швидко сприймати інформацію та знаходити візуальні закономірності.

1.4 Огляд аналогів

Розрахунок і використання ефективних інструментів, програм або методик є ключовими елементами успішної реалізації проектів у сучасному світі. У різних галузях, починаючи від науки та технологій, закінчуючи бізнесом і управлінням, існує широкий спектр аналогових рішень, які сприяють досягненню поставлених цілей. Розрахунок і використання ефективних інструментів також сприяють ефективному управлінню часом і оптимізації ресурсів. Збільшення продуктивності, вдосконалення процесів і зниження ймовірності помилок можна досягти за допомогою сучасних програмних методів і рішень. Зокрема, у сфері науки та технологій використання сучасних інструментів дозволяє швидше проводити дослідження, аналізувати великі обсяги даних і розробляти новаторські рішення. Компанії можуть отримати конкурентоспроможність на ринку, управління ризиками та покращення маркетингових стратегій за допомогою ефективних програм і методів.

Одними з популярних і надійних джерел інформації щодо оплати праці в Україні є сайти працевлаштування, такі як work.ua та rabota.ua. Ці сайти не лише надають можливість пошуку вакансій, але й містять значну кількість об'єктивних даних про ринкову вартість різних спеціальностей та рівень оплати праці в різних галузях. Вбудовані в ці сайти системи аналізу даних оплати праці дозволяють користувачам переглядати середній рівень заробітної плати рис 1.3 та рис 1.4 для певних посад або спеціальностей.

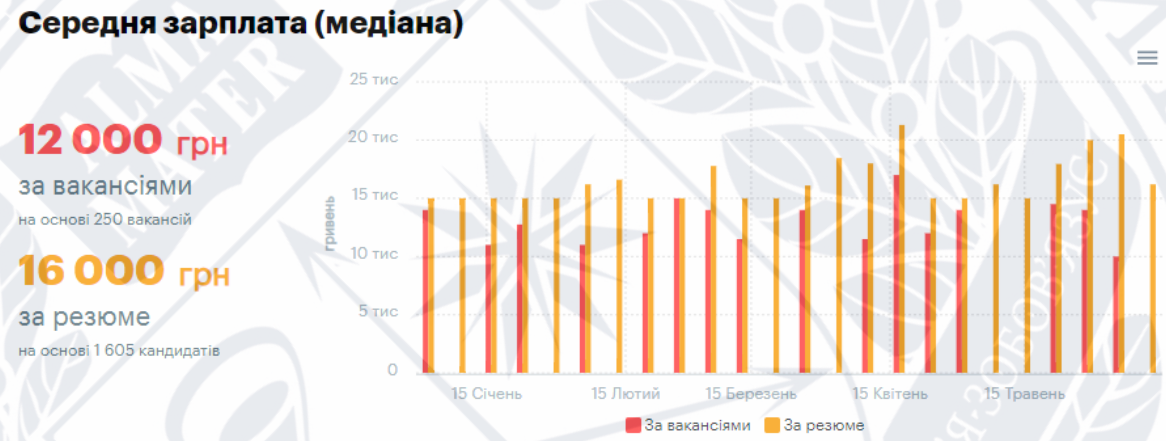

Рисунок 1.3 - Середня зарплата rabota.ua

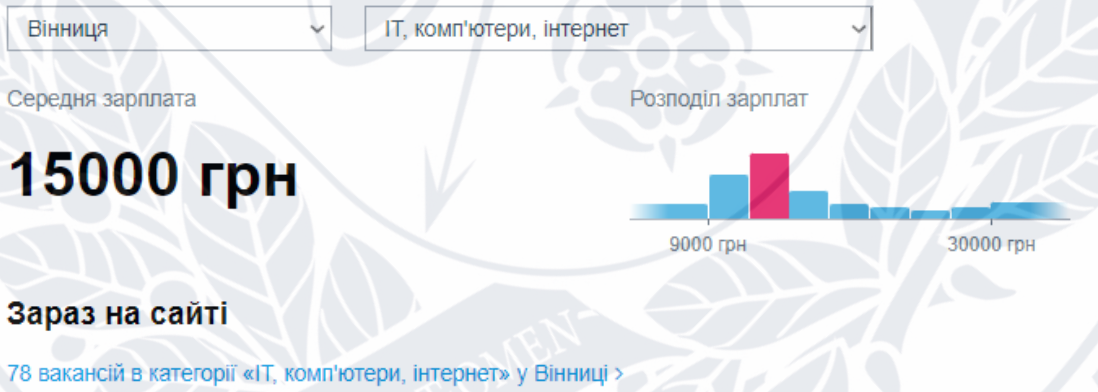

Рисунок 1.4- Середня зарплата work.ua

Головною відмінністю цих двох графіків це те, що на сайті rabota.ua можна побачити більш детальний графік, а саме: зарплату за перший та другий квартал, останні три місяці та цілий рік. Також сайт пропонує обрати за який рік користувач хоче отримати інформацію. Дана інформація заснована на об'єктивних даних, отриманих з опитувань працівників, статистичних звітів та даних про заробітну плату відповідних фахівців у відповідних галузях. Ці дані

дозволяють отримати уявлення про середній рівень оплати праці у різних сферах і порівняти його зі своїм фінансовим становищем. Будучи потужними інструментами, системи аналізу даних оплати праці на сайтах work.ua та rabota.ua дозволяють фахівцям оцінити свою фінансову ситуацію, порівняти її з іншими спеціалістами у тій же галузі та зробити обґрунтовані висновки щодо своєї кар'єри. Вони сприяють збору об'єктивних даних, уникненню спекуляцій та неправдивих тверджень, а також допомагають виявляти можливі нерівності або дискримінацію в оплаті праці.

Основний недолік цих сайтів це те, що користувач може не отримати інформацію по своїй, або цікавій йому спеціальності, наприклад QA - інженер. Якщо обрати сайт rabota.ua та виставити критерій QA – інженер, сайт не надасть необхідну інформацію рис 1.5, а сайт work.ua надасть інформацію лише по всій Україні, Києві, або дистанційно рис 1.6.

Рисунок 1.5 - Середня зарплата QA rabota.ua

Рисунок 1.6- Середня зарплата QA work.ua

Інші недоліки цих сайтів:

- Обмежений доступ до даних - інформація про оплату праці сайти отримують від виставлених вакансій, а велика частина вакансій не вказує заробітну плату.
- Сайти не враховують додаткові фактори, які впливають на заробітну працю – системи аналізу дані оплати праці, враховують лише вказану заробітну плату, а деякі вакансії вказують лише початкову зарплату та прописують за надбавку в залежності від професійних навичок робітника.
- Відсутність актуальності даних – інформація про оплату може мати свою актуальність лише деякий проміжок часу, наприклад роботодавець виставляє вакансію за деяку суму та зникає на пів року, а за ці пів року оплата могла змінитись.

Ще два українські сайти, але уже спеціалізовані для пошуку роботи, або аналізу оплати праці в сфері IT це: djinni.co та jobs.dou.ua. Розробники цих сайтів зробили ставку не лише на великому функціоналі для пошуку робити, а й на аналізі даних оплати праці. Українець, який перебуває не на території України, або прагне знайти роботу за кордоном може в критерії пошуку обрати майже любу країну світу. Спеціалізації тут також набагато детальніше

розкриті. Аналіз оплати праці на сайтах djinni.co та jobs.dou.ua побудований не лише за мінімальною, середньою та максимальною оплатою праці так як це зробили розробники на сайтах work.ua та robota.ua, що дає можливість зрозуміти, що функціонал був не такою і важливою частиною при розробці цих сайтів. Графік зарплати тестувальників зображені на рис 1.7 та 1.8.

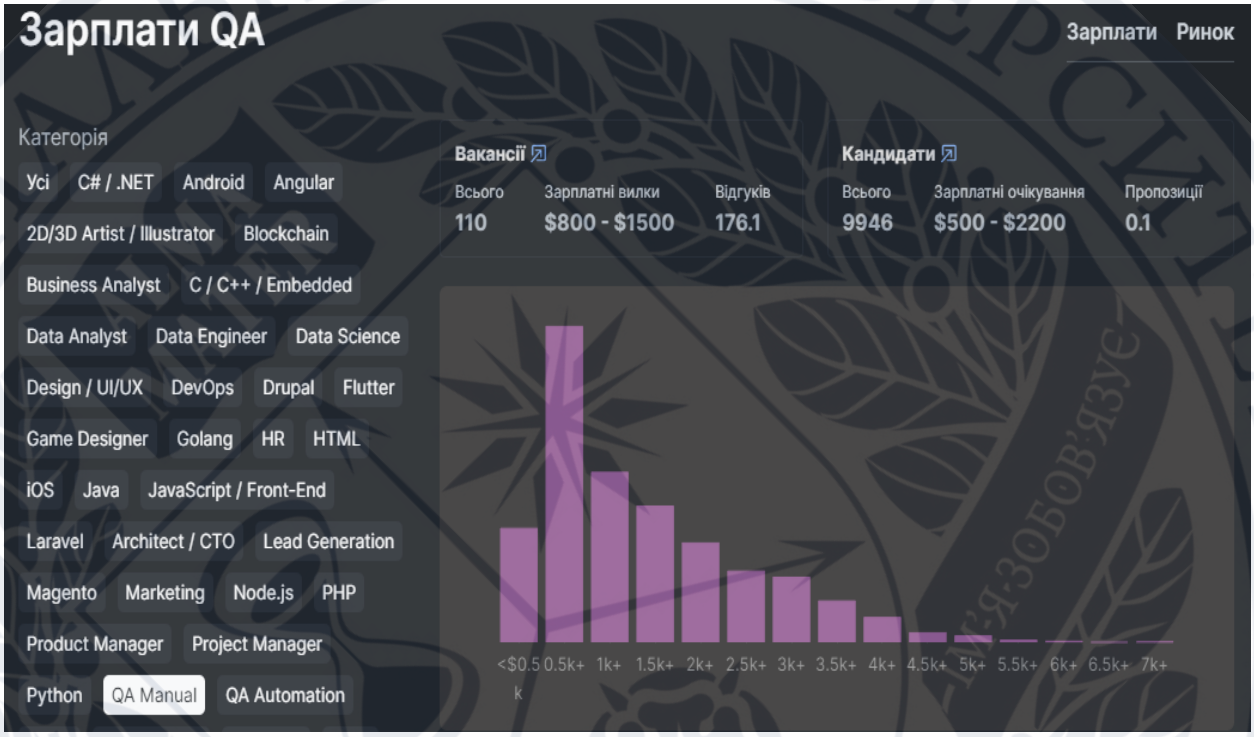

Рисунок 1.7 - Графік зарплати тестувальників dijinni

Так, окрім критеріїв «категорія» та «місто», що є спільна для цих сайтів на сайті dijinni добавленно ще критерій з:

1. Досвід роботи
    1.1. Будь який;
    1.2. Менше 1 року;
    1.3. 1-2 роки;
    1.4. 2-3 роки;
    1.5. 3-5 роки.
2. Рівнем знання англійської мови
    2.1. No English;
    2.2. Beginner/Elementary;
    2.3. Pre-Intermediate;

2.4. Intermediate;

2.5. Upper-Intermediate;

2.6. Advanced/Fluent.

А на сайті dou

1. Домен
   1.1. AR/VR;
   1.2.Travel;
   1.3. ERP;
   1.4. Internet of Things;
   1.5. Public services / Government;
   1.6. Real estate;
   1.7. Blockchain;
   1.8. Та інші.
2. Тип компанії
   2.1. Аутсорсингова;
   2.2. Аутстафінгова;
   2.3. Продуктова;
   2.4. Стартап.

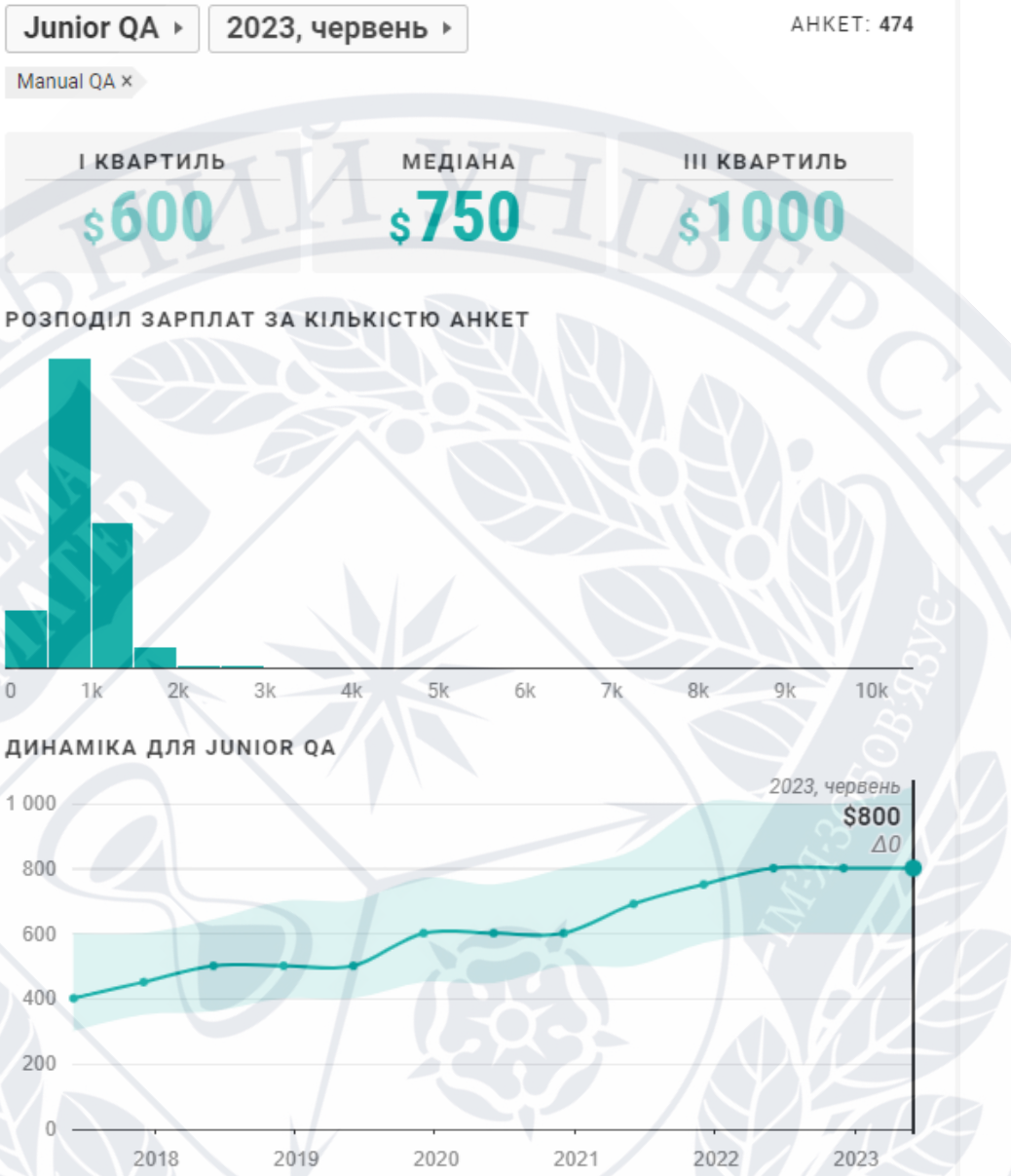

Рисунок 1.8 - Графік зарплати тестувальників dou

Отже, основною відмінністю між цими чотирма сайтами є те, що останні два сайти спіалізовані під IT, тому і вакансії тут набагато серйозніші та з більшими запитами для майбутніх працівників. Більшість іноземних компаній та українських флагманів у сфері IT викладають свої вакансії саме на цих сайтах, тому для сфер IT вакансій тут значно більше та різноманітніші. А це значить, що статистика на цих сайтах буде показувати різну інформацію, тому

що в них різна база даних і якщо на одному сайті якась компанія виклала своє резюме – це не означатиме що вона викладе резюме на інших трьох сайтах.

І останній аналог, але уже програмна платформа – ADP Workforce Now. ADP Workforce Now На GoSM це додаток для роботодавців, щоб переглядати, керувати та управляти інформацією про заробітну плату та час своєї компанії в будь-який час, з будь-якого місця. Ця програма революціонує шлях доступу роботодавця до своєї інформації про заробітну плату та час роботи через простий у використанні інструмент, доступний на мобільних пристроях, планшеті та настільних комп'ютерах. ADP Workforce Now надає широкий спектр функцій та інструментів, спрямованих на оптимізацію і автоматизацію процесів управління персоналом, включаючи наступні аспекти:

- Управління персоналом - дозволяє управляти інформацією про співробітників, включаючи досьє співробітника, документи, час роботи, відпустки та відсутності на роботі.
- Заробітна плата та оплата праці –дозволяє автоматизувати процеси розрахунку заробітної плати, включаючи обробку годин праці, податкові утримання, виплати та звітність.
- Управління перевагами - допомагає керувати програмами, такими як: медичне страхування, пенсійні та інші соціальні пакети.
- Аналітика та звітність - ADP Workforce Now надає розширені засоби аналізу даних та створення звітів щодо різних аспектів управління персоналом та оплати праці.
- Взаємодія зі співробітниками – програма надає можливість співробітникам самостійно управляти своїми персональними даними, включаючи доступ до графіків роботи, заявок на відпустку та інших важливих інформаційних ресурсів.

Переваги:

1. Інтегрована система в якій запропоновано багато видів функцій та інструментів, які об’єднані в одній платформі;

2. Автоматизація процесів – завдяки цій програмі можна зберегти багато часу, який пов’язаний з управлінням персоналу та оплати їхньої праці. Зменшує багато машинальних помилок, які може допустити користувач;
3. Аналітика та звітність.

Недоліки:

1. Вартість – для деяких компаній вартість цієї програми може бути завеликою, а також ціна змінюється від обсягу функціоналу та потреб, які необхідні компанії;
2. Складність впровадження – в залежності від функціоналу системи час на впровадження може бути дуже довгим та ресурсозатратним. Програма вимагає налагодження інтеграції з існуючими системами. Також програма є доволі важкою для навчання персоналу;
3. Залежність від постачальника, а саме від ADP.

Важливо враховувати, що переваги та недоліки ADP Workforce Now можуть бути індивідуальними для кожної компанії, залежно від її потреб, розміру та бізнес-моделі.

Висновки до розділу

В даному розділі був проведений аналіз оплати праці, наведені формули розрахунку заробітної плати.

Розглянуто поняття інформаційної системи, складові та необхідне технічне забезпечення для роботи інформаційнійної системи.

Прописана постановка задача та проведений огляд аналогів.

РОЗДІЛ 2
ОГЛЯД МОВ ПРОГРАМУВАНЯ ТА ЇХ БІБЛІОТЕК

2.1 Python

Python був створений в 1991 році Гвідо ван Россумом і з тих пір став однією з найпопулярніших мов програмування. Він має простий і зрозумілий синтаксис, що робить його відмінним вибором для початківців у програмуванні. Python підтримує багато парадигм програмування, включаючи процедурне програмування, об'єктно-орієнтоване програмування та функціональне програмування [6].

Щоб розпочати програмування на Python, спочатку потрібно встановити Python на свою систему. Можна завантажити офіційну версію Python з веб-сайту python.org і встановити її на свій комп'ютер, дотримуючись інструкцій для операційної системи. Після встановлення Python можна запускати програми Python через командний рядок або інтегроване середовище розробки (IDE), таке як PyCharm, Visual Studio Code, IDLE або Jupyter Notebook. Ці середовища надають зручні інструменти для редагування, виконання та налагодження коду Python. Також було створено для початківців інтегроване середовище розробки (IDE) – Thonny рис 2.1. Thonny - це інтегроване середовище розробки для Python, призначене для початківців. Воно підтримує різні способи покрокового виконання коду, покрокове обчислення виразів, детальну візуалізацію стеку викликів і режим пояснень концепцій посилань та купа. Воно надає простий та зрозумілий інтерфейс, що допомагає новачкам швидко освоїти основи програмування [7].

Основні особливості Thonny включають:

- Спрощений інтерфейс: Thonny має мінімалістичний та зрозумілий інтерфейс, який полегшує розуміння та використання для новачків.
- Відладка: Thonny надає можливість крок за кроком відлагоджувати програми, допомагаючи виявляти та виправляти помилки.

- Автодоповнення: IDE надає автоматичне доповнення коду, що полегшує написання програм та допомагає уникнути помилок.
- Перевірка синтаксису: Thonny відображає помилки синтаксису під час написання коду, що дозволяє виявляти та виправляти їх на ранніх етапах.
- Інтегрована довідка: IDE містить вбудовану довідку, яка надає інформацію про різні функції та методи мови Python.
- Підтримка мікроконтролерів: Thonny має можливість взаємодії з мікроконтролерами, такими як Arduino, що дозволяє програмувати їх безпосередньо з IDE.

Thonny є відкритим програмним забезпеченням та підтримується активною спільнотою розробників. Його можна встановити на різні операційні системи, такі як Windows, macOS та Linux. Thonny є чудовим вибором для початківців, які хочуть швидко вивчити основи програмування на Python.

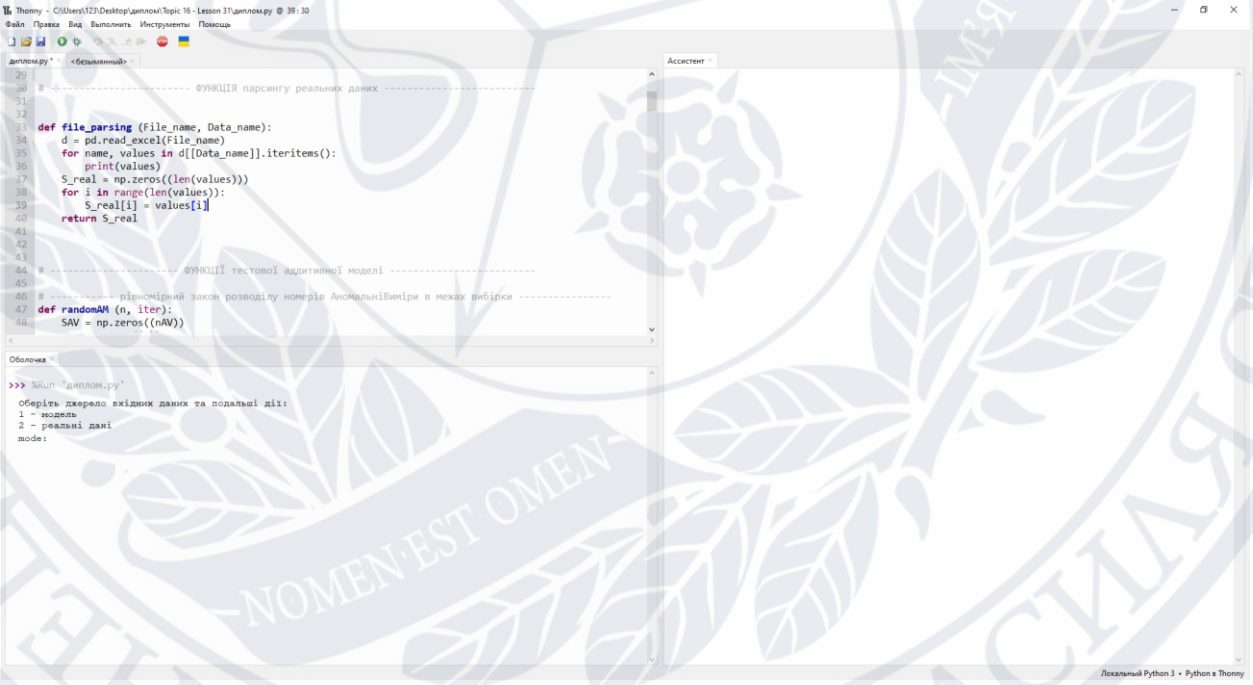

Рисунок 2.1 – «Thonny»

На даний момент існує дві активно підтримувані версії Python: Python 2 і Python 3. Рекомендується використовувати Python 3, оскільки Python 2 перестав отримувати оновлення підтримки з 2020 року.

Python є однією з найбільш популярних мов програмування, і вона має свої переваги та недоліки.

Переваги Python:

1. Простота вивчення і використання: Python має простий і зрозумілий синтаксис, що дозволяє швидко розуміти і писати код;
2. Багата бібліотека та велика спільнота: Python має велику кількість бібліотек для різних задач, таких як наукові обчислення, машинне навчання, обробка даних, візуалізація тощо. Крім того, є велика спільнота розробників, яка активно підтримує та розвиває мову;
3. Крос-платформенність: Python працює на багатьох платформах, включаючи Windows, macOS, Linux та інші;
4. Інтерактивність: Python має інтерактивну оболонку, що дозволяє виконувати код крок за кроком, щоб перевіряти результати;
5. Відкритий код: Python має відкритий вихідний код, що дозволяє розробникам вносити власні внески в розвиток мови.

Недоліки Python:

1. Швидкість виконання: Python не є найшвидшою мовою програмування і може бути повільним для великих обчислювальних задач;
2. Пам'ять: Python може займати більше пам'яті, ніж інші мови програмування, такі як C++ або Java;
3. Гнучкість: Python може бути менш гнучким порівняно з іншими мовами програмування, такими як C++, які надають більше контролю над системою та ресурсами;
4. GIL: Python має Global Interpreter Lock (GIL), що може обмежувати багатопоточність та призводити до проблем зі збоєм.

2.2 Бібліотека NumPy

Numpy — розширення мови Python, що додає підтримку великих багатовимірних масивів і матриць, разом з великою бібліотекою високорівневих математичних функцій для операцій з цими масивами. NumPy

є основою для багатьох інших бібліотек та пакетів, що використовуються у наукових обчисленнях та аналізі даних, таких як SciPy, pandas, Matplotlib і багато інших.

Основна перевага використання NumPy полягає в його здатності до швидкого та ефективного обчислення над масивами даних. Він надає оптимізовані операції з векторами та матрицями, що значно прискорює виконання обчислень. Крім того, NumPy має вбудовані функції для виконання різних математичних операцій, таких як обчислення середнього значення, медіани, варіації, сортування та багато інших.

Основні особливості NumPy включають:

- Масиви: NumPy надає об'єкти масивів, що дозволяють зручно представляти та маніпулювати числовими даними.
- Багатовимірні масиви: Масиви NumPy можуть мати багато вимірів, що дозволяє представляти та операціювати багатовимірні дані.
- Універсальні функції: NumPy надає широкий спектр універсальних функцій, що дозволяють виконувати операції над масивами елемент за елементом.
- Індексація та зрізи: Масиви NumPy підтримують потужну індексацію та зрізи, що дозволяють вибирати та маніпулювати конкретними елементами або підмасивами.
- Векторизація: Завдяки оптимізованим операціям над масивами, NumPy сприяє векторизації коду, що дозволяє виконувати обчислення на великій кількості даних ефективно та швидко.

Установка та імпорт NumPy включає наступні кроки:

1. Установка NumPy:
   1.1. Відкрийте командний рядок (термінал) на вашому комп'ютері;
   1.2. Виконайте наступну команду для установки NumPy за допомогою pip (пакетний менеджер Python) pip install numpy.

Для успішної установки NumPy може знадобитися підтримка інтернет-підключення.

1. Імпорт NumPy у програму Python:
    1.1. Відкрийте вашу програму Python або інтерактивну оболонку Python;
    1.2. Додайте наступний рядок коду у верхній частині вашої програми import numpy as np.

Цей рядок коду імпортує бібліотеку NumPy та дозволяє використовувати її функції та методи у вашій програмі. Ключове слово np є популярним псевдонімом, який використовується для звернення до NumPy. Також можна використати команду «from numpy import *», але такий спосіб не є рекомендований через ряд помилок та проблем під час на писання коду.

Масиви є основними структурами даних у бібліотеці NumPy. Вони дозволяють зберігати та операціювати над числовими даними, такими як числа або значення. Основні аспекти, які варто розглянути:

1. Створення масивів:
    1.1. Ручне створення масивів з використанням np.array();
    1.2. Використання вбудованих функцій для створення масивів, таких як np.ones(), np.arange(), np.linspace() та інші;
    1.3. Задання розмірності та типу даних масиву.
2. Робота з масивами:
    2.1. Отримання та зміна значень елементів масиву;
    2.2. Отримання розміру масиву за допомогою shape;
    2.3. Отримання кількості вимірів масиву за допомогою ndim.
3. Операції над масивами:
    3.1. Математичні операції, такі як додавання, віднімання, множення, ділення та піднесення до степеня;
    3.2. Операції порівняння та логічні операції між масивами;
    3.3. Універсальні функції NumPy для операцій над масивами, такі як np.sum(), np.mean(), np.max(), np.min() та інші.
4. Індексація та зрізи:
    4.1. Вибір окремих елементів масиву за допомогою індексів;
    4.2. Використання зрізів (slicing) для отримання підмасивів;

4.3. Використання умовної індексації для отримання підмасиву, що задовольняє певні умови.

5. Робота з багатовимірними масивами:
   5.1. Робота з масивами з більш ніж двома вимірами;
   5.2. Індексування та зрізи багатовимірних масивів.
6. Функції для обробки масивів:
   6.1. Функції для сортування, фільтрації та перетворення масивів;
   6.2. Матричні операції, такі як транспонування, обертання та обчислення детермінанту.

Крім базового варіанта (багатомірні масиви в базовому варіанті) NumPy включає набір пакетів для вирішення спеціалізованих завдань, наприклад:

1. Математичні функції:

np.add(): Додавання елементів масивів.

np.subtract(): Віднімання елементів масивів.

np.multiply(): Множення елементів масивів.

np.divide(): Ділення елементів масивів.

np.power(): Піднесення елементів масиву до заданого степеня.

np.sqrt(): Обчислення квадратного кореня кожного елементу масиву.

2. Математичні функції статистики:

np.mean(): Обчислення середнього значення елементів масиву.

np.median(): Обчислення медіани елементів масиву.

np.var(): Обчислення дисперсії елементів масиву.

np.std(): Обчислення стандартного відхилення елементів масиву.

3. Функції для роботи зі формою масиву та самим масимов:

np.reshape(): Зміна форми масиву.

np.flatten(): Перетворення масиву в одновимірний масив.

np.zeros(): Створює масив заповнений нулями;

np.median(): Обчислення медіани масиву чисел;

np.var(): Обчислює дисперсію масиву чисел.

2.3 Бібліотека math

Бібліотека math є вбудованою бібліотекою в Python і надає функції та константи для виконання математичних операцій. Для її використання необхідно написати лише import math, але для більш зручнішого використання краще прописувати import math as mt. Таким способом щоб використовувати функції з бібліотеки math необхідно прописувати лише mt. Ця бібліотека містить широкий спектр математичних функцій, які можуть бути корисними для різних обчислювальних задач. Ось декілька основних функцій та констант, які включені в бібліотеку math:

1. Математичні константи:

math.pi: Ця константа представляє значення числа π (пі).

math.e: Ця константа представляє значення числа e, основи натурального логарифма.

2. Математичні функції:

math.sqrt(x): Обчислює квадратний корінь числа x.

math.exp(x): Обчислює експоненту (e^x) числа x.

math.log(x, base): Обчислює логарифм числа x за підставою base. Якщо base не вказано, використовується натуральний логарифм (з підставою e).

math.log10(x): Обчислює десятковий логарифм числа x.

math.sin(x), math.cos(x), math.tan(x): Обчислюють синус, косинус та тангенс кута x (в радіанах).

math.degrees(x): Перетворює кут x з радіанів в градуси.

math.radians(x): Перетворює кут x з градусів в радіани.

2.4 Бібліотека matplotlib

Matplotlib — бібліотека на мові програмування Python для візуалізації даних двовимірною 2D графікою (3D графіка також підтримується). Приклад зображення графіка зображено на рис 2.2. Основною частиною Matplotlib є модуль pyplot, який надає інтерфейс для створення графіків та налаштування їх вигляду. Для того щоб вставити дану бібліотеку небхідно:

1. Відкрити командний рядок або термінал у системі;
2. Введіть наступну команду для встановлення matplotlib за допомогою пакетного менеджера pip: pip install matplotlib

Зачекати доки процес встановлення не буде завершений. Після цього matplotlib буде успішно встановлено.

Основні можливості бібліотеки Matplotlib включають:

1. Побудова графіків:

Лінійні графіки - plot();

Точкові графіки - scatter();

Стовпчасті - діаграми bar(), barh();

Гістограми - hist();

Кругові діаграми - pie();

Ящики з вусами - boxplot();

Теплові карти - imshow();

та багато інших типів графіків.

2. Налаштування вигляду графіків:

Додавання заголовка, підписів осей, легенди;

Зміна кольорів, стилів ліній, заповнення областей;

Налаштування меж значень на осі;

Додавання сітки, маркерів, тексту;

Зміна шрифтів, розмірів, орієнтації графіків.

3. Робота зі зображеннями:

Завантаження та відображення зображень - imread(), imshow();

Масштабування, обрізання, обробка зображень;

Додавання колірної шкали, підписів.

4. Збереження графіків:

Збереження графіків у різних форматах PNG, PDF, SVG, і т.д.

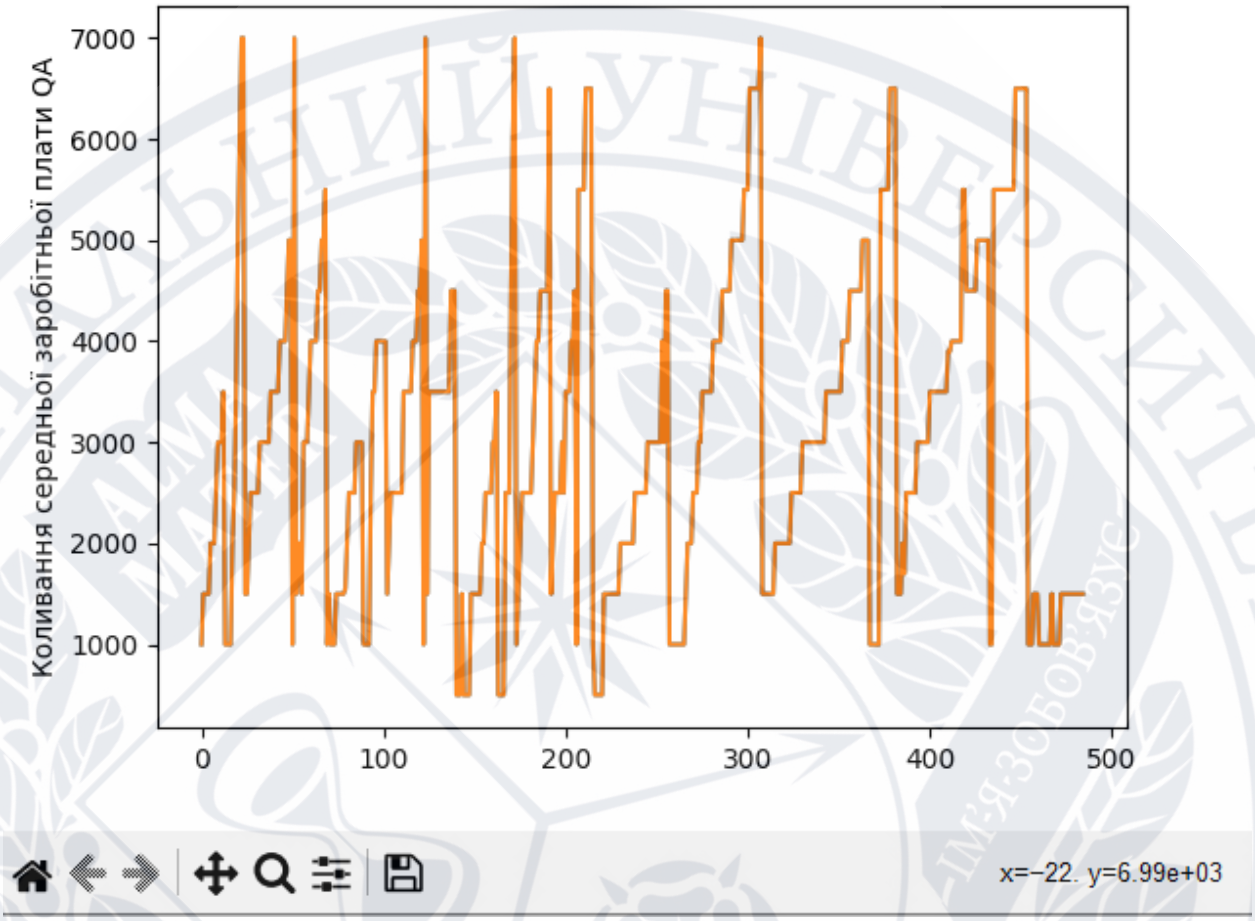

Рисунок 2.2 – приклад зображення графіка

Matplotlib також працює добре з іншими бібліотеками для наукових обчислень, такими як NumPy та Pandas, і надає зручні інструменти для візуалізації даних з цих бібліотек.

2.5 Бібліотека Pandas

Pandas - програмна бібліотека, написана для мови програмування Python для маніпулювання даними та їхнього аналізу. Вона, зокрема, пропонує структури даних та операції для маніпулювання чисельними таблицями та часовими рядами. Основні особливості Pandas:

- Структури даних: Pandas пропонує дві основні структури даних - Series та DataFrame. Ці структури даних дозволяють легко організовувати та маніпулювати даними.

- Завантаження та збереження даних: Pandas надає зручні інструменти для імпорту даних з різних джерел, таких як CSV-файли, Excel-файли, бази даних, а також для збереження даних у різних форматах.
- Обробка та маніпуляція даними: Pandas дозволяє виконувати різноманітні операції з даними, включаючи сортування, фільтрацію, вибірку, об'єднання, з'єднання та злиття даних. Вона також надає можливість застосування функцій та операцій до даних, що дозволяє зручно виконувати розрахунки та обчислення.
- Візуалізація даних: Pandas має інтегровану підтримку візуалізації даних, що дозволяє легко створювати графіки, діаграми та інші візуальні елементи для аналізу та представлення даних.
- Робота з пропущеними даними: Pandas має вбудовану підтримку для роботи з пропущеними даними, що дозволяє ефективно виявляти, обробляти та заповнювати пропущені значення у наборах даних.

Pandas є незамінним інструментом для аналізу та обробки даних у Python, і вона широко використовується у галузі науки про дані, фінансів, соціальних науках, машинному навчанні та багатьох інших сферах. Її простий синтаксис та потужні можливості роблять її відмінним вибором для роботи з даними у Python.

Щоб встановити бібліотеку Pandas та імпортувати її у свій проект, необхідно виконати наступні кроки:

Відкрити командний рядок або термінал у системі та ввести наступну команду для встановлення Pandas за допомогою пакетного менеджера pip: pip install pandas. Зачекати доки процес встановлення не буде завершений. Після цього Pandas буде успішно встановлено на вашу систему. Для імпорт Pandas у проект необхідно у коді Python, імпортувати її на початку вашого скрипта. Це можна зробити за допомогою наступного рядка коду - import pandas as pd. У цьому прикладі імпортується Pandas та використовуємо псевдонім pd. Це стандартна практика, що дозволяє коротше звертатися до функцій та класів Pandas під час написання коду. Після імпортування Pandas надається

можливіть використовувати усі функції та класи для обробки та аналізу даних у вашому проекті.

У бібліотеці Pandas доступні дві основні структури даних: Series та DataFrame. Розглянемо кожну з цих структур детальніше:

Series: Series є одновимірною структурою даних, що містить послідовність значень. Вона схожа на масив або список у Python, але має додаткову можливість індексації, що дозволяє легко маркувати та доступатися до елементів у серії.

Основні особливості Series:

- Кожен елемент у серії може бути будь-якого типу даних, такого як числа, рядки, булеві значення тощо.
- Серія має індекс, що ідентифікує кожен елемент у серії. За замовчуванням, індексування відбувається автоматично, але можна задати власний індекс.
- Серія може бути створена зі списку, масиву NumPy або іншої серії за допомогою конструктора pd.Series().
- DataFrame: DataFrame є двовимірною таблицею даних, що складається з рядків та стовпців. Вона є основною структурою даних у Pandas і надає потужний інструмент для роботи з табличними даними.
- Основні особливості DataFrame:
- Кожен стовпець у DataFrame представляє собою окремий об'єкт Series.
- DataFrame має два індекси: індекс рядків та індекс стовпців. Індекс рядків ідентифікує кожен рядок у таблиці, а індекс стовпців - кожен стовпець.
- DataFrame може бути створений зі списку, словника, масиву NumPy або інших DataFrame за допомогою конструктора pd.DataFrame().

Структури даних Series та DataFrame дозволяють легко та ефективно працювати зі структурованими даними, виконувати операції фільтрації,

сортування, об'єднання, агрегування та багато інших операцій для аналізу та обробки даних.

2.6 Бібліотека SciPy

SciPy (Scientific Python) - це бібліотека для наукових обчислень у Python. Вона містить багато модулів для різноманітних задач наукових обчислень, таких як оптимізація, інтерполяція, обробка сигналів, обробка зображень та чисельна лінійна алгебра. Основна мета бібліотеки SciPy полягає в тому, щоб забезпечити користувачам Python інструменти для проведення наукових обчислень з високою якістю та ефективністю. SciPy базується на NumPy та використовує багато функцій NumPy для роботи з масивами.

Основні можливості SciPy:

Оптимізація: включає методи мінімізації, знайдення коренів рівнянь та розв'язування диференціальних рівнянь;

Інтерполяція: дозволяє проводити інтерполяцію даних, знаходження сплайнів та інших типів апроксимації;

Обробка сигналів: включає фільтрацію сигналів, статистичний аналіз та обробку звуку;

Обробка зображень: дозволяє виконувати операції обробки зображень, такі як розмиття, збільшення та зменшення розміру, розпізнавання об'єктів та інші;

Чисельна лінійна алгебра: включає методи розв'язування систем лінійних рівнянь, обернення матриць та знаходження власних значень.

2.7 Мова R

R — мова програмування і програмне середовище для статистичних обчислень, аналізу та зображення даних в графічному вигляді. Розробка R відбувалась під істотним впливом двох наявних мов програмування: мови програмування S з семантикою, успадкованою від Scheme. R названа за першою літерою імен її засновників Роса Іхаки (Ross Ihaka) та Роберта

Джентлмена, (Robert Gentleman) працівників Оклендського Університету в Новій Зеландії. Незважаючи на деякі принципові відмінності, більшість програм, написаних мовою програмування S запускаються в середовищі R. R використовує текстовий інтерфейс, однак існують різні графічні інтерфейси. R має значні можливості для здійснення статистичних аналізів, включаючи лінійну і нелінійну регресію, класичні статистичні тести, аналіз часових рядів (серій), кластерний аналіз і багато іншого. Основні особливості мови R:

1. R має вбудовані функції та пакети для широкого спектру статистичних методів, включаючи дескриптивну статистику, регресійний аналіз, аналіз варіант, кластерний аналіз, аналіз часових рядів та багато іншого;
2. R має потужні засоби для створення високоякісних графіків та візуалізації даних. Він підтримує різноманітні типи графіків, включаючи гістограми, діаграми розсіювання, лінійні графіки, кругові діаграми та теплові карти;
3. R має велику кількість пакетів, які розширюють його функціональність. Ці пакети дозволяють виконувати різноманітні завдання, від машинного навчання до біоінформатики. Користувачі можуть створювати власні пакети та додатки для розвитку нових функцій;
4. R надає інтерактивне середовище для розробки та виконання коду. Користувачі можуть взаємодіяти з мовою R через консоль або інтерактивні ноутбуки, такі як RStudio, що дозволяють виконувати код по частинах та спостерігати результати безпосередньо;
5. R є кросплатформенним середовищем, що означає, що він може працювати на різних операційних системах, включаючи Windows, macOS та Linux;
6. R є вільним програмним забезпеченням з відкритим вихідним кодом. Це означає, що користувачі можуть переглядати та модифікувати вихідний код мови R, додатків та пакетів.

Інтерфейс мови R зображено на рис. 2.3

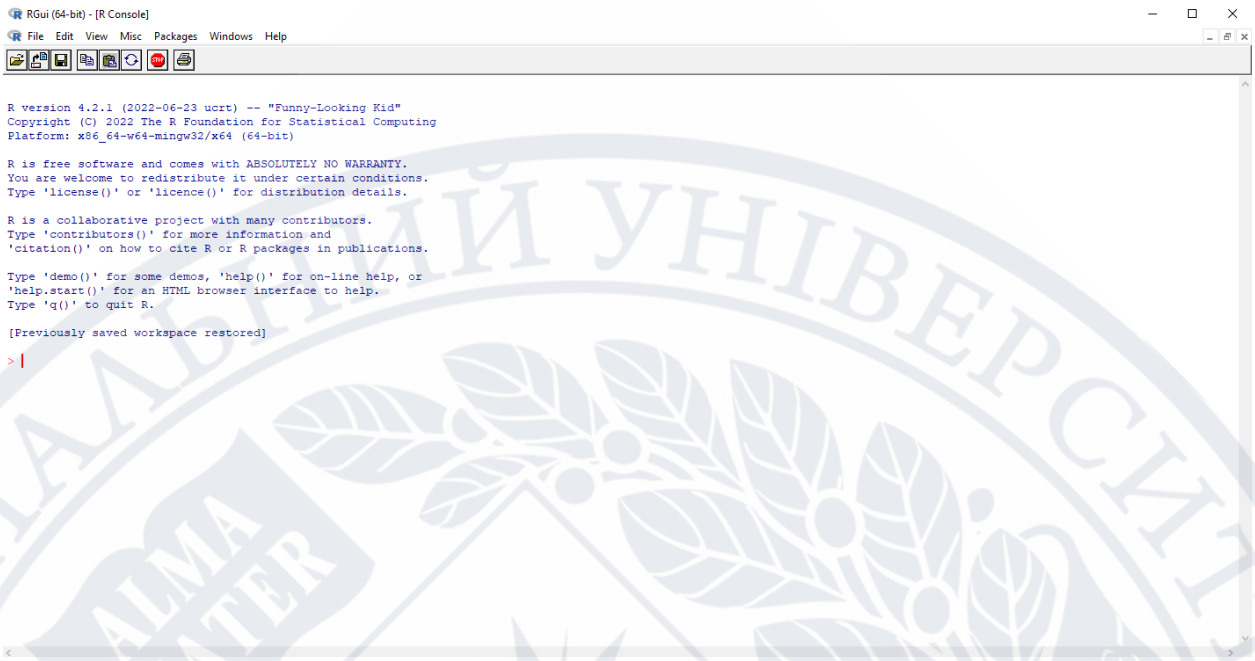

Рисунок 2.3 - Інтерфейс мови R

Мова R є однією з найпопулярніших мов програмування серед статистиків та аналітиків даних завдяки своїй потужності, гнучкості та розширюваній функціональності. Використання R дозволяє робити складні обчислення, аналізувати та візуалізувати дані, будувати статистичні моделі та виконувати багато інших завдань, пов'язаних з обробкою даних. Окрім того, R має активну спільноту користувачів та розробників, що сприяє швидкому вирішенню проблем та обміну знаннями. У мові R також існує велика кількість документації, пакетів та розширень, що робить її потужним інструментом для роботи з даними. Варто зазначити, що R використовується в різних сферах, включаючи академічні дослідження, фінанси, біоінформатику, соціальні науки, медицину та багато інших. Його гнучкість та масштабованість дозволяють використовувати його для різноманітних завдань з обробки та аналізу даних.

Переваги мови R:

- R має велику кількість пакетів та бібліотек, що надають розширені можливості для аналізу даних, статистики та машинного навчання. Це дозволяє користувачам виконувати різноманітні завдання без необхідності писати код з нуля.

- R має потужні засоби для візуалізації даних. Завдяки пакетам, таким як ggplot2, користувачі можуть створювати високоякісні графіки та графічні представлення даних для виведення важливої інформації з наборів даних.
- R є популярним серед статистиків та дослідників, оскільки має вбудовані функції для виконання широкого спектру статистичних аналізів. Він підтримує регресійний аналіз, аналіз варіант, кластерний аналіз, аналіз часових рядів та багато інших статистичних методів.
- R надає можливість взаємодії з кодом у режимі реального часу через консоль або інтерактивні ноутбуки, такі як RStudio. Це дозволяє користувачам експериментувати з даними, виконувати швидкий аналіз та отримувати миттєвий зворотний зв'язок.
- R є вільним програмним забезпеченням з відкритим вихідним кодом. Це означає, що ви можете переглядати та змінювати вихідний код, що дає вам більшу гнучкість та контроль над мовою.

Недоліки мови R:

- Порівняно з деякими іншими мовами програмування, такими як Python або C++, R може бути повільним. Це особливо помітно при обробці великих обсягів даних або при виконанні складних обчислень. Однак, існують способи оптимізувати швидкодію R, наприклад, використовуючи векторизацію та компіляцію коду.
- Вивчення мови R може зайняти більше часу та зусиль, особливо для початківців. Синтаксис R може бути складним для розуміння, і вимагає певної кількості практики, щоб стати ефективним користувачем.
- Мова R не надає таку саму міру підтримки для розробки великомасштабних проектів, як деякі інші мови програмування. Вона більш придатна для аналізу даних та прототипування, ніж для розробки великих програмних систем.

- Хоча R має деякі пакети для розробки веб-додатків (наприклад, Shiny), він не так широко використовується для веб-розробки, як, наприклад, Python або JavaScript.
- Хоча R має деякі пакети для машинного навчання, такі як caret або randomForest, він не має такого широкого спектру бібліотек та інструментів для машинного навчання, як Python. Тому, якщо основна мета - машинне навчання, можливо, варто розглянути інші мови програмування.

До аналогових бібліотек які були використані при роботою з Python для мови R можуть бути:

1. Бібліотека NumPy - Rcpp та RcppArmadillo. Rcpp - це пакет, який дозволяє використовувати C++ код в R, що забезпечує швидкодію та ефективність. RcppArmadillo - це пакет, який надає широкий спектр функцій для чисельних обчислень, включаючи роботу з масивами та матрицями;
2. Бібліотека math - базова R має вбудовані математичні функції, які надають подібний функціонал до бібліотеки math в Python. Наприклад, функції sin(), cos(), exp(), log() тощо;
3. Бібліотека matplotlib - в R є кілька пакетів для візуалізації даних, таких як ggplot2, lattice та базова R graphics. ggplot2 є одним з найпопулярніших пакетів для створення високоякісних графіків, який надає декларативний підхід до побудови графіків;
4. Бібліотека Pandas - в R найбільш близьким еквівалентом Pandas є пакет data.table. data.table надає швидкі та ефективні інструменти для маніпулювання та аналізу даних у форматі таблиць;
5. Бібліотека SciPy - в R існує декілька пакетів, які надають функціонал, схожий на SciPy. Наприклад, пакети stats та MASS містять багато статистичних методів та алгоритмів, а пакет optim дозволяє розв'язувати оптимізаційні задачі.

Обираючи між Python і R для аналізу даних, по наведених перевагах та недоліків було прийняте рішення писати код на мові Python тому що ця мова

є загальнопризначеною мовою програмування, це означає, що можна використовувати її не тільки для аналізу даних, але й для різних інших завдань програмування. Python підтримує широкий спектр бібліотек, модулів та фреймворків для аналізу даних, машинного навчання, веб-розробки та багатьох інших областей. Має одну з найбільших та найактивніших спільнот серед мов програмування. Завдяки цьому можна дуже легко і швидко знайти велику кількість ресурсів, документації, пакетів та підтримки від спільноти. Python має потужну екосистему для аналізу даних, включаючи популярні бібліотеки, такі як NumPy, Pandas, Matplotlib і SciPy, а також інструменти для машинного навчання, такі як TensorFlow і Scikit-learn. Синтаксис Python є простим та лаконічним, що робить його легким для вивчення та розуміння, особливо для новачків у програмуванні. Python прагне до створення зрозумілого та читабельного коду, що сприяє співпраці та розумінню коду командами. Python має підтримку для швидкого виконання завдяки використанню бібліотек, таких як NumPy та Pandas, які оптимізовані для ефективної обробки числових даних та масивів.

Висновки до розділу

В даному розділі був проведений огляд мови Python та R, історія створення, основні особливості, переваги та недоліки цих програм. Наведена інформація про використані бібліотеки. Та розписано чому саме було обрано мову програмування Python, а не R.

РОЗДІЛ 3

РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ АНАЛІЗУ ДАНИХ ОПЛАТИ ПРАЦІ

3.1 Архітектура програмного забезпечення

Як було сказано у розділі 1.2, одним з модулів інформаційної системи є модуль програмного забезпечення. Архітектура програмного забезпечення - це розроблена структура додатка, яка включає визначення взаємодії компонентів інтерфейсу з внутрішніми процесами програми [8]. Архітектура створеного програмного забезпечення зображена на рис 3.1.

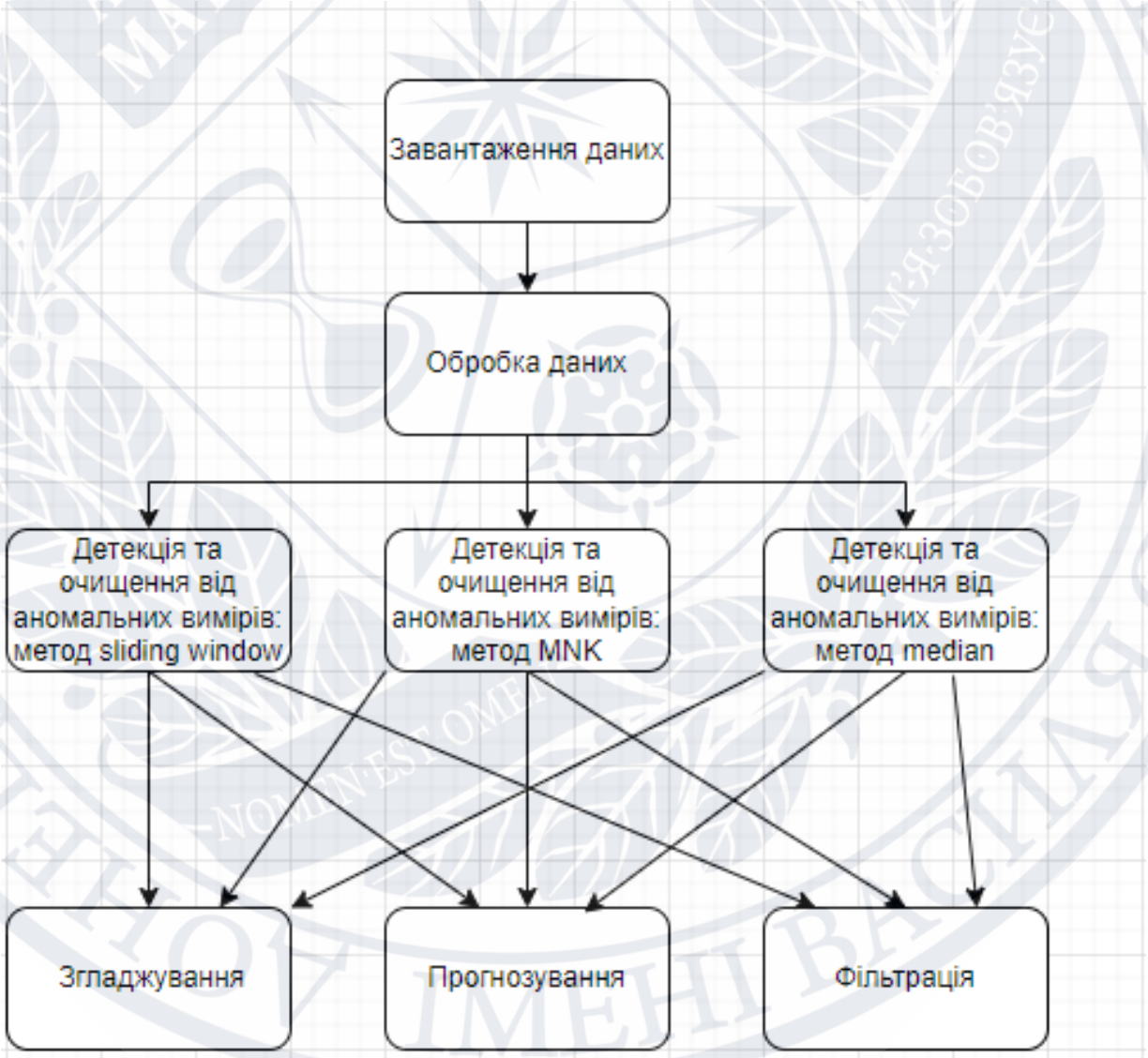

Рисунок 3.1 - Архітектура програмного забезпечення

Перш ніж користувач розпочне взаємодію з програмою, необхідно завантажити дані з DataSet та впровадити їх у програму. На етапі обробки даних, програма виконує перевірку формату даних та отримує лише числові значення. Після цього етапу виникає необхідність обрати метод для виявлення та очищення даних від аномальних вимірів. Наступним кроком стає вибір користувачем подальших дій: проведення згладжування даних, здійснення прогнозування або використання фільтрації.

Детекція та очищення від аномальних вимірів, часто відомих як аномалії або викиди, є процесом виявлення та видалення нетипових, незвичайних або аномальних даних у наборі. Цей процес важливий у багатьох областях, таких як обробка сигналів, аналіз даних, машинне навчання, фінанси, медицина та інші. Детекція аномалій – це етап, на якому визначаються або виявляються аномалії в даних. Він може включати визначення точкових аномалій (окремих вимірювань, що суттєво відрізняються від інших) або аномалій у групах даних. Методи детекції аномалій можуть включати:

- Статистичні методи: застосування статистичних підходів, таких як Z-оцінка, для визначення відхилень від типового значення.
- Машинне навчання: Використання алгоритмів класифікації або кластеризації для визначення, які екземпляри вважаються аномаліями.
- Ідентифікація зміни підпису: Порівняння зразка даних з попереднім станом або нормою для виявлення змін.

Очищення від аномалій:

Це етап, на якому виявлені аномалії піддаються обробці, яка може включати видалення аномальних даних, заміну їхніми середніми значеннями або іншими методами корекції. Методи очищення від аномалій можуть включати:

- Вилучення аномальних значень - просте видалення аномальних вимірювань з набору даних.
- Заміна значень - заміна аномальних значень на середні або медіанні значення для зменшення впливу аномалій.

- Використання алгоритмів інтерполяції - використання алгоритмів для визначення недостаючих або аномальних значень на основі інших даних.

Існують багато методів для детекції та очищення даних від аномалій. Кожен з цих методів призначений для якоїсь конкретної цілі тому не існує якогось універсального методу для обробки даних. Три метода для детекції та очищення даних, які пропонує система:

1. Детекція та очищення від аномальних вимірів методом median;
2. Детекція та очищення від аномальних вимірів методом найменших квадратів;
3. Детекція та очищення від аномальних вимірів методом sliding window.

Метод медіани може використовуватися для очищення від аномалій у даних. Він полягає у заміні аномальних значень на медіану набору даних. Медіана є статистичною мірою центральної тенденції, яка представляє середнє значення в середині впорядкованого набору даних. Вона менш чутлива до викидів або аномалій, ніж середнє значення, що робить її корисною для очищення даних від аномалій. Процедура очищення даних методом медіани виглядає наступним чином.

1. Виявлення аномальних значень - визначивши медіану потрібно виявити аномальні значення, які відрізняються від медіани на задану порогову величину. Порогова величина може бути встановлена на основі досвіду або доменних знань. Зазвичай використовується стандартне відхилення, де значення, що відрізняються від медіани більше ніж на кілька стандартних відхилень, вважаються аномальними;
2. Заміна аномальних значень на медіану - після виявлення аномалій, їх можна замінити на медіану. Це означає, що аномальні значення будуть замінені на значення медіани набору даних. Цей підхід дозволяє замінити аномальні значення більш типовими показниками.

Метод найменших квадратів (МНК) не є безпосередньою технікою для детекції та очищення аномалій, але його можна використовувати для цих цілей

в певних випадках. МНК може бути використаний для детекції та очищення аномалій таким способом:

1. Побудова лінійної моделі - МНК використовується для побудови лінійної моделі, що найкращим чином апроксимує залежність між залежною (вимірювана) змінною і незалежними (пояснюючими) змінними. Це може бути корисно для опису типової поведінки даних;
2. Виявлення аномалій залишків - побудувавши модель за допомогою МНК, можна оцінити залишки (різницю між спостереженими значеннями та прогнозованими значеннями моделі). За допомогою статистичних методів, таких як розподіл залишків чи порогова величина, можна виявити аномальні значення залишків, що вказують на нетипову поведінку даних;
3. Очищення аномалій - за допомогою виявлення аномалій залишків МНК, можна видалити або виправити аномальні виміри. Це можна зробити, замінивши аномальні значення на прогнозовані значення моделі або видаливши їх з набору даних.

Метод "sliding window" (кочуючого вікна) ідея полягає в тому, щоб рухати вікно фіксованого розміру по послідовності даних і аналізувати кожен підвіконний набір даних з метою виявлення аномалій.

Основні кроки методу "sliding window" для детекції та очищення аномалій:

1. Визначення розміру вікна - спочатку потрібно визначити розмір кочуючого вікна, який відповідає кількості даних, яку ви хочете аналізувати одночасно. Цей розмір може бути встановлений на основі знань про домен, характеристики даних або експериментальним шляхом;
2. Обчислення метрики - в кожному вікні визначається метрика, яка використовується для оцінки аномальності даних. Це може бути, наприклад, середнє значення, стандартне відхилення, медіана або будь-яка інша метрика, яка відображає типову поведінку даних в цьому вікні;
3. Виявлення аномалій - за допомогою визначеної метрики аналізується кожне вікно, і виміри, які відрізняються від типового поведінки, вважаються аномаліями. Порогове значення для визначення аномалій може

бути встановлено експериментальним шляхом або на основі статистичних розрахунків;

4. Очищення аномалій - після виявлення аномалій їх можна очистити, замінивши або видаливши їх. Залежно від конкретного випадку, аномальні значення можуть бути замінені на значення з попереднього або наступного вікна, на середнє значення вікна або на інше значення, яке відповідає типовій поведінці даних.

Фільтр Калмана, відомий також як лінійно-квадратичне оцінювання - це алгоритм, що використовує послідовності вимірювань протягом часу, які містять шум (випадкові відхилення) та інші неточності, й видає оцінки невідомих змінних, що є потенційно точнішими за базовані на самих лише вимірюваннях. Формальніше, фільтр Калмана працює рекурсивно на потоках зашумлених вхідних даних, і видає статистично оптимальну оцінку базового стану системи. Фільтр названо на честь Рудольфа Калмана, одного з головних розробників його теорії [9]. Фільтр Калмана (альфа-бета фільтр) є модифікацією класичного фільтра Калмана і використовується для оцінки стану системи на основі вимірів з датчиків. Цей фільтр особливо корисний, коли система має нелінійну динаміку. Основна ідея альфа-бета фільтра полягає у розрахунку оцінки стану системи, використовуючи два коефіцієнти: альфа і бета. Альфа відповідає за оновлення оцінки стану на основі нових вимірів, а бета відповідає за ковзне середнє попередньої оцінки стану. Основні кроки альфа-бета фільтра:

1. Ініціалізація - фільтр починається з початкової оцінки стану системи та початкової коваріаційної матриці. Початкова оцінка стану може бути отримана з наявних даних або задана на підставі досвіду. Початкова коваріаційна матриця відображає невизначеність початкової оцінки стану;
2. Оновлення на основі вимірів - фільтр отримує нові виміри з датчиків і оновлює оцінку стану. Оновлення включає розрахунок двох компонентів: альфа-компоненту і бета-компоненту;

2.1. Альфа-компонента - альфа-компонента використовується для оновлення оцінки стану на основі нових вимірів. Вона враховує різницю між вимірами та прогнозованими значеннями, а також коваріаційну матрицю шуму виміру;
2.2. Бета-компонента - бета-компонента використовується для оновлення коваріаційної матриці оцінки стану. Вона враховує попередню коваріаційну матрицю, матрицю шуму процесу та матрицю шуму виміру;
3. Повторення кроку 2 для кожного нового виміру - фільтр постійно оновлює оцінку стану та коваріаційну матрицю на основі нових вимірів.

Це основний процес альфа-бета фільтра. Важливо зазначити, що ефективність фільтра залежить від правильного налаштування коефіцієнтів альфа і бета. Вони визначають, як сильно нові виміри та попередня оцінка стану впливають на оновлення. Альфа-бета фільтр є більш простим в реалізації порівняно з класичним фільтром Калмана, оскільки не вимагає прогнозування. Однак, він може бути менш точним у випадках, коли система має складну нелінійну динаміку або коли шум виміру та шум процесу є великими.

Метод найменших квадратів прогнозування та згладжування. Метод найменших квадратів (МНК) широко використовується у статистиці і економетриці для побудови регресійних моделей прогнозування. Основна ідея МНК полягає у знаходженні лінійної або нелинійної функції, яка найкраще відповідає спостереженим даним, шляхом мінімізації суми квадратів відхилень між спостереженими значеннями і прогнозованими значеннями моделі. Конкретні кроки методу найменших квадратів для прогнозування:

1. Вибір моделі - спочатку потрібно вибрати математичну модель, яка найкраще відповідає характеру даних і типу залежності між змінними. Це може бути лінійна модель, поліноміальна модель, експоненціальна модель або інша;
2. Побудова функції прогнозування - наступним кроком є побудова функції прогнозування, яка описує залежність між незалежними і залежною

змінною. Наприклад, у випадку лінійної моделі, функція прогнозування може мати вигляд y = mx + b, де y - залежна змінна, x - незалежна змінна, m - нахил (коэфіцієнт), b - зсув (перехоплення);

3. Оцінка параметрів моделі - за допомогою МНК розраховуються оптимальні значення параметрів моделі, які мінімізують суму квадратів відхилень між спостереженими значеннями і прогнозованими значеннями моделі. Це може виконуватись за допомогою аналітичних методів або чисельних алгоритмів оптимізації;
4. Прогнозування - після оцінки параметрів моделі можна використовувати цю модель для прогнозування значень залежної змінної для нових незалежних значень. Прогнози можуть бути отримані шляхом обчислення значень функції прогнозування для нових значень незалежних змінних.

Метод найменших квадратів (МНК) згладжування даних означає видалення шуму або випадкових коливань з даних, щоб виділити загальну тенденцію або показати більш плавну залежність між змінними. Один з підходів до згладжування даних за допомогою МНК полягає в побудові регресійної моделі, де залежна змінна прогнозується на основі незалежних змінних. Проте, у випадку згладжування ці моделі зазвичай мають просту структуру і низький ступінь складності, щоб уникнути перенавчання. Основні кроки методу найменших квадратів для згладжування даних:

1. Вибір моделі - спочатку потрібно вибрати просту математичну модель, яка найкраще відповідає загальній тенденції даних. Наприклад, це може бути лінійна модель, поліноміальна модель невисокого ступеня або експоненціальна модель;
2. Оцінка параметрів моделі - за допомогою МНК розраховуються оптимальні значення параметрів моделі, які найкраще відповідають даним. Це може виконуватись за допомогою аналітичних методів або чисельних алгоритмів оптимізації;
3. Прогнозування - після оцінки параметрів моделі можна використовувати цю модель для згладжування даних. Замість прогнозування нових значень

залежної змінної, ви можете використовувати модель для прогнозування значень наявних точок даних. Отримані прогнозовані значення стануть згладженими значеннями даних. Прогнозовані значення можуть бути використані для аналізу тенденцій, виявлення піків або впадин, або для отримання загальної плавної кривої, яка відображає основну залежність між змінними.

3.2 Розробка програмного забезпечення

Для виконання кваліфікаційної (магістерської) роботи було вирішено знайти, проаналізувати та скачати актуальний DataSet заробітної плати тестувальників за 2022 рік в Україні. Проаналізувавши декілька загально відкритих наборів даних, знайшовши актуальніший та доповнивши його деякою додатковою інформацією і видозмінивши його для зручнішої праці з ним в програмі, було отримано такий вигляд рис. 3.2.

1	місто	досвід	зарплата_сер	зарплата_мін	зарплата_макс
2	Вінниця	1-3 роки	1000	500	1000
3	Вінниця	1-3 роки	1500	1000	1500
4	Вінниця	1-3 роки	1500	1000	1500
5	Вінниця	1-3 роки	1500	1000	1500
6	Вінниця	1-3 роки	1500	1000	1500
7	Вінниця	1-3 роки	2000	1500	2000
8	Вінниця	1-3 роки	2000	1500	2000
9	Вінниця	1-3 роки	2000	1500	2000
10	Вінниця	1-3 роки	2500	2000	2500
11	Вінниця	1-3 роки	3000	2500	3000
12	Вінниця	1-3 роки	3000	2500	3000
13	Вінниця	1-3 роки	3000	2500	3000
14	Вінниця	1-3 роки	3500	3000	3500
15	Вінниця	1-3 роки	1000	500	1000
16	Вінниця	1-3 роки	1000	500	1000
17	Вінниця	1-3 роки	1000	500	1000
18	Вінниця	1-3 роки	1000	500	1000
19	Вінниця	10+ років	2000	1500	2000
20	Вінниця	10+ років	2500	2000	2500
21	Вінниця	10+ років	3500	3000	3500
22	Вінниця	10+ років	5000	4500	5000
23	Вінниця	10+ років	6000	5500	6000
24	Вінниця	10+ років	7000	6500	7000
25	Вінниця	10+ років	7000	6500	7000
26	Вінниця	4-6 років	1500	1000	1500
27	Вінниця	4-6 років	1500	1000	1500
28	Вінниця	4-6 років	2000	1500	2000
29	Вінниця	4-6 років	2500	2000	2500
30	Вінниця	4-6 років	2500	2000	2500
31	Вінниця	4-6 років	2500	2000	2500
32	Вінниця	4-6 років	2500	2000	2500
33	Вінниця	4-6 років	2500	2000	2500
34	Вінниця	4-6 років	3000	2500	3000

Рисунок 3.2 – отриманий DataSet

Подальша робота полягала у виборі необхідної мови програмування для аналізу даних, статистики та розробки інформаційної системи. Для поставленої задачі одними з найпопулярнішими мовами програмування є R та Python, але з урахуванням усіх переваг та недоліків було обрано мову програмування Python. Для отримання інформації з DataSet була використана функція парсинга. Парсинг (аналіз) - це процес розбору структурованої або напівструктурованої інформації з одного формату в інший або з одного джерела до іншого.

Парсинг може бути використаний для різних завдань, таких як:

- Отримання даних зі структурованих форматів, таких як XML (eXtensible Markup Language) або JSON (JavaScript Object Notation).

- Обробка HTML-сторінок для отримання певної інформації з веб-сайтів (наприклад, витягнення заголовків новин або цін товарів).
- Розбір лог-файлів для аналізу подій або помилок.
- Аналіз та обробка структурованих даних, таких як CSV (Comma-Separated Values) або TSV (Tab-Separated Values).

Функція парсингу:

```
def file_parsing (File_name, Data_name):
    read = pd.read (File_name)
    for name, values in read[[Data_name]].iteritems():
        print(values)
    F_p = np.zeros((len(values)))
    for i in range(len(values)):
        F_p[i] = values[i]
    return F_p
```

Функція file_parsing призначена для обробки файлів та витягування певного стовпця даних, який вказується програмістом.

Аргументи функції:

- File_name: рядок, який представляє ім'я файлу, з якого потрібно прочитати дані.
- Data_name: рядок, який представляє ім'я стовпця даних, які потрібно витягнути з файлу.

Функція використовує бібліотеки pandas та numpy для роботи з даними. Спочатку вона зчитує вміст файлу за допомогою pd.read (File_name). Далі, за допомогою циклу for, проходиться по стовпцю даних Data_name за допомогою iteritems(), щоб отримати доступ до значень цього стовпця. Кожне значення виводиться на друк за допомогою print(values).

Після чого створюється масив F_p з нульовими значеннями розміром, що дорівнює кількості елементів у values. За допомогою циклу for проходиться по значеннях values і кожне значення присвоюється відповідному індексу в масиві F_p за допомогою F_p[i] = values[i].

Наостанок, функція повертає масив F_p, який містить дані файли стовпця зарплата середня. Отриману інформацію можна побачити на рис 3.3

```
0       1000
1       1500
2       1500
3       1500
4       1500
       ...
480     1500
481     1500
482     1500
483     1500
484     1500
Name: зарплата_сер, Length: 485, dtype: int64
```

Рисунок 3.3 – Отримана інформація з DataSet

Наступний хід роботи полягав в перетворені отриманої інформації в графічний вигляд для більш зручного відображення інформації. Для цього була побудована функція Plot_Vector.

```
def Plot_ Vector (Vector1, Vector2, Text):
plt.clf()
plt.plot(Vector2)
plt.plot(Vector1)
plt.ylabel(Text)
plt.show()
return
```

Функція Plot_Vector призначена для побудови графіку двох векторів Vector1 і Vector2 залежно від їх значень та надання підпису графіку.

Аргументи функції:

Vector1: вектор даних, який буде відображений на графіку.

Vector2: вектор даних, який також буде відображений на графіку.

Text: рядок, який представляє підпис графіку.

У функції спочатку викликається plt.clf(), щоб очистити поточний графік перед побудовою нового.

Потім за допомогою plt.plot(Vector2) та plt.plot(Vector1) будуються дві лінії графіку для векторів Vector2 та Vector1 відповідно.

plt.ylabel(Text) встановлює підпис осі Y графіку, який вказується в аргументі Text.

За допомогою plt.show() графік відображається на екрані рис 3.4.

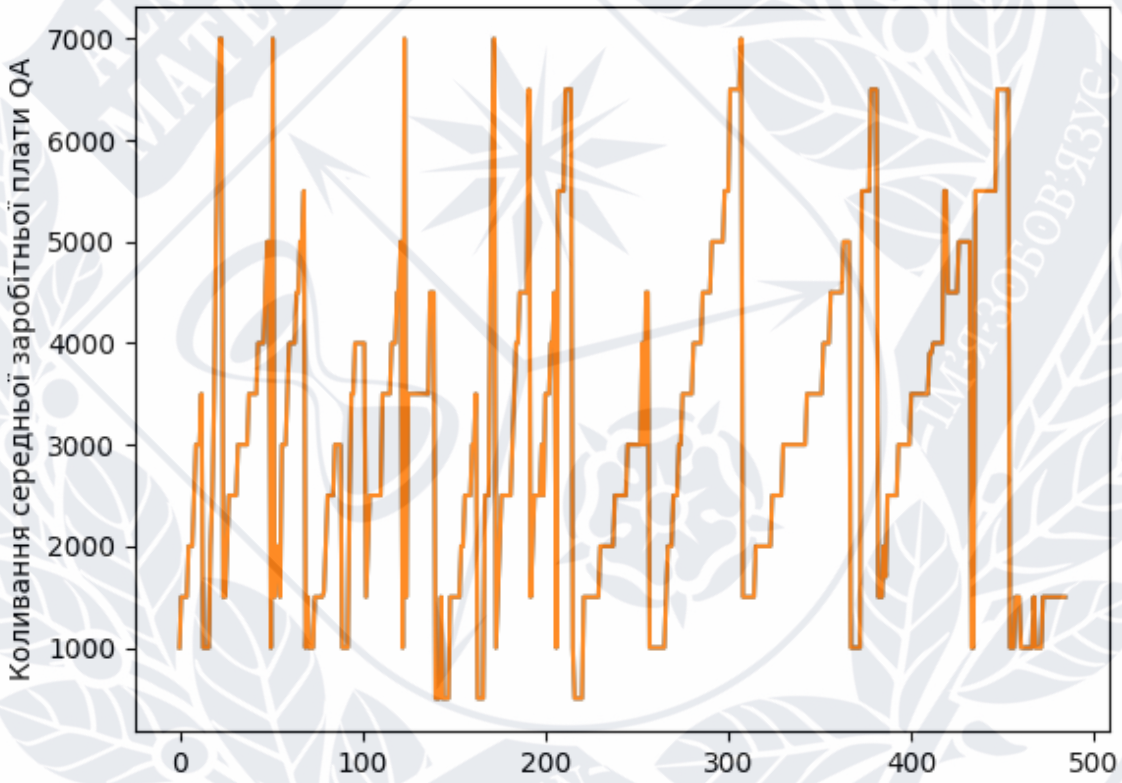

Рисунок 3.4 – Відображений графік

Наступним кроком став розрахунок статистичних характеристик. Програмою було прораховано:

1. Математичне сподівання випадкової величини – числова характеристика випадкової величини, що відповідає інтуїтивному поняттю її середнього значення [15];

2. Дисперсія випадкової величини – Диспе́рсія (англ. variance) — це міра розсіяння значень випадкової величини відносно середнього значення розподілу. Більші значення дисперсії свідчать про більші відхилення значень випадкової величини від центру розподілу [16];
3. Стандартне квадратичне відхилення випадкової величини - Станда́ртне відхи́лення (англ. standard deviation) або середнє квадратичне відхилення — у теорії ймовірностей і статистиці один із найпоширеніших показників розсіювання (розкиду) значень випадкової величини відносно її математичного сподівання, тобто центру розподілу. Має ту ж розмірність, що і випадкова величина. В літературі для позначення стандартного відхилення використовується літера грецької абетки сигма σ [17].

```
def Stat_char (mas, Text):
  D1 = MNK_Stat_char (mas)
   iter = len(D1)
   mas0 = np.zeros((iter ))
   for i in range(iter):
      mas0[i] = mas[i] – D1[i, 0]
   median = np.median(mas0)
   var = np.var(mas0)
   sqrt = mt.sqrt(dS)
   print('------------', Text ,'-------------')
   print('математичне сподівання Випадкової Величини=', median)
   print('дисперсія Випадкової Величини =', var)
   print(' Стандартне Квадратичне Відхилення Випадкової Величини=',
sqrt)
   print('--------------------------------------------------')
   return
```

Функція Stat_char призначена для обчислення статистичних характеристик вибірки з урахуванням тренду.

Аргументи функції:

- SL: масив, який представляє вибірку.
- Text: рядок, який містить текстову назву для виведення результатів.

У функції спочатку викликається функція MNK_Stat_char (SL), яка обчислює характеристики з урахуванням тренду для вибірки mas. Результат зберігається у змінній D1. Далі створюється масив mas0 з нульовими значеннями розміром, що дорівнює iter (довжині D1). За допомогою циклу for проходиться по значеннях D1 і обчислюється різниця між mas[i] та D1[i, 0]. Отримані значення зберігаються у масиві mas0.

Потім обчислюються статистичні характеристики вибірки mas0:

- median: медіана (np.median) вибірки mas0.
- var: дисперсія (np.var) вибірки mas0.
- sqrt: стандартне квадратичне відхилення (mt.sqrt) вибірки mas0.

Результати виводяться на екран за допомогою функції print, включаючи назву (Text) та обчислені значення статистичних характеристик.

```
def MNK_Stat_char (S0):
   iter = len(S0)
   Yin = np.zeros((iter, 1))
   mas = np.ones((iter, 3))
# формування структури вхідних матриць МетодНайменшихКвадратів
   for i in range(iter):
# формування матриці
 Yin[i, 0] = float(S0[i])
      F[i, 1] = float(i)
      F[i, 2] = float(i * i)
   FT=F.T
   FFT = FT.dot(F)
   FFTI=np.linalg.inv(FFT)
   FFTIFT=FFTI.dot(FT)
   C=FFTIFT.dot(Yin)
   D1=F.dot(C)
```

return D1

Функція MNK_Stat_char виконує метод найменших квадратів для обчислення статистичних характеристик на основі вхідних даних S0.

Аргументи функції:

- S0: вектор вхідних даних.

У функції створюються різні матриці, такі як Yin, F, FT, FFT, FFTI, FFTIFT, C та D1, для розрахунків методу найменших квадратів. У циклі for створюються матриці Yin та F з використанням вхідних даних S0. Значеннями матриці F заповнюються індекси та їх квадрати. Далі, виконується транспонування матриці F за допомогою FT = F.T. Потім обчислюється матриця FFT як результат множення транспонованої матриці FT на матрицю F. Наступною кроком є обчислення оберненої матриці FFTI від FFT за допомогою np.linalg.inv(FFT). Потім обчислюється матриця FFTIFT як результат множення FFTI на транспоновану матрицю FT. За допомогою матриці FFTIFT та вхідних даних Yin, обчислюється вектор C. Обчислюється вихідний вектор D1 шляхом множення матриці F на вектор C. Результат D1 повертається з функції. На рис 3.5 зображені результати розрахунків.

```
---------- Коливання середньої заробітньої плати QA ----------
математичне сподівання Випадкової Величини= -171.74018587992305
дисперсія Випадкової Величини = 2555988.9475701544
 Стандартне КвадратичнеВідхилення Випадкової Величини= 1598.7460547473306
--------------------------------------------------
```

Рисунок 3.5 – Результат розрахунків

Наступний крок очищення графіка від аномальних вимірів методом найменших квадратів.

Функція для виявлення аномалій методом найменших квадратів.

```
def Sliding_Window_AV_Detect_MNK (S0, Q, n_Wind):
    iter = len(S0)
  j_Wind=mt.ceil(iter-n_Wind)+1
  S0_Wind=np.zeros((n_Wind))
  Speed_standart = MNK_AV_Detect(SV_AV)
  Yout_S0 = MNK(SV_AV)
```

```
    for j in range(j_Wind):
      for i in range(n_Wind):
        l=(j+i)
        S0_Wind[i] = S0[l]
        dS = np.var(S0_Wind)
      scvS = mt.sqrt(dS)
        Speed_standart_1 = abs(Speed_standart * mt.sqrt(iter))
      Speed_1 = abs(Q * Speed_standart * mt.sqrt(n_Wind) * scvS)
      if Speed_1  > Speed_standart_1:
        # детектор виявлення АномальніВиміри
        S0[l]=Yout_S0[l,0]
    return S0
```

Оголошується функція Sliding_Window_AV_Detect_MNK, яка приймає три аргументи: S0, Q і n_Wind. Ця функція призначена для виявлення аномалій в часовому ряді S0 за допомогою методу найменших квадратів (МНК) та ковзного вікна. Обчислюється довжина часового ряду S0 і зберігається в змінній iter. Це важливо, оскільки нам потрібно буде перебирати дані в часовому ряді. Обчислюємо кількість ковзних вікон j_Wind. Це робиться за допомогою функції mt.ceil, яка округлює значення до найближчого цілого числа. Кількість ковзних вікон залежить від довжини S0 та ширини ковзного вікна n_Wind. Створюється масив S0_Wind розміром n_Wind. Цей масив буде використовуватися для зберігання вимірів, які потрапляють в поточне ковзне вікно. Обчислюється значення Speed_standart, яке, отримується з іншої функції MNK_AV_Detect, використовуючи дані SV_AV. Це значення служить в якості "стандарту" для подальшого порівняння. Також обчислюється значення Yout_S0 за допомогою іншої функції MNK. Починається цикл, який проходить через ковзні вікна в часовому ряді. Цей цикл виконується для кожного ковзного вікна. Усередині кожного ковзного вікна дані зберігаються у масив S0_Wind. Обчислюються статистичні характеристики цього ковзного вікна, такі як дисперсія dS та середньоквадратичне відхилення scvS. Проводиться

порівняння швидкостей Speed_1 та Speed_standart з використанням деяких обчислень. Якщо Speed_1 більше за Speed_standart, то це вважається аномалією, і дані в часовому ряді S0 замінюються на відповідні значення з Yout_S0. На виході функція повертає оновлений часовий ряд S0. В результаті чого ми отримуємо графік рис 3.6.

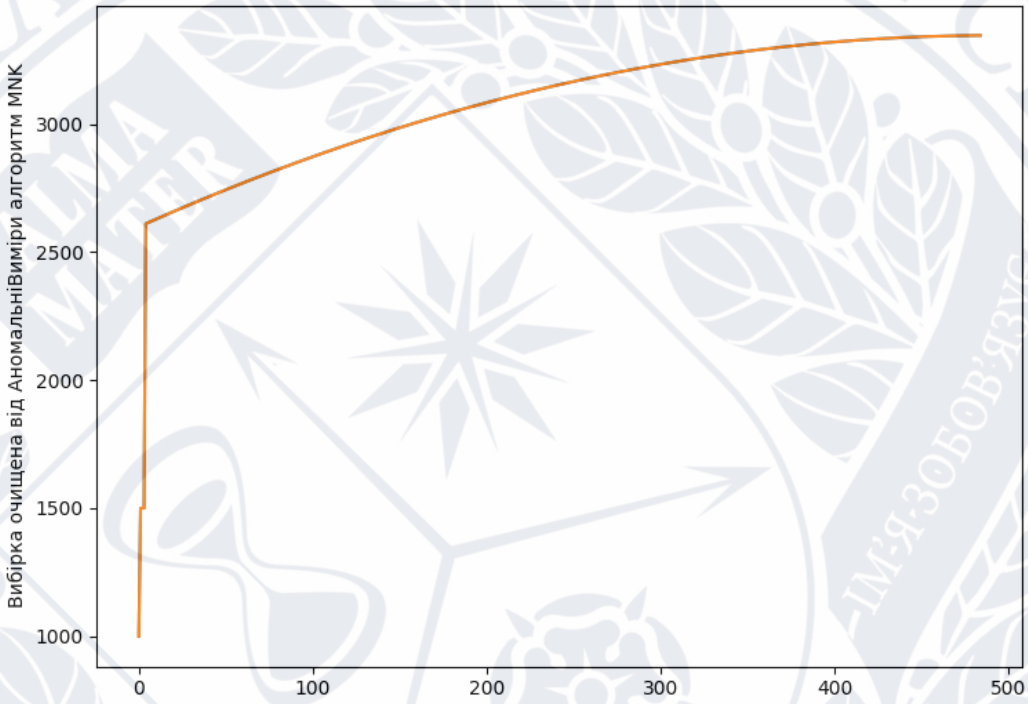

Рисунок 3.6 – Результат методу найменших квадратів

Ще один метод очищення від аномальних вимірів це "ковзного вікна" (Sliding Window).

Функція виявлення аномальних вимірів методом sliding window.

```
def Sliding_Window_AV_Detect_sliding_wind (S0, n_Wind):

  iter = len(S0)
  j_Wind=mt.ceil(iter-n_Wind)+1
  S0_Wind=np.zeros((n_Wind))
  Midi = np.zeros(( iter))
    for j in range(j_Wind):
     for i in range(n_Wind):
```

```
            l=(j+i)
            S0_Wind[i] = S0[l]
        # - Стат хар ковзного вікна --
        Midi[l] = np.median(S0_Wind)
    S0_Midi = np.zeros((iter))
    for j in range(iter):
        S0_Midi[j] = Midi[j]
    for j in range(n_Wind):
        S0_Midi[j] = S0[j]
    return S0_Midi
```

Функція отримує два аргументи: S0, який є вхідним масивом даних, і n_Wind, який вказує ширину ковзного вікна для обробки. Визначається довжина вхідного масиву S0 і зберігається в змінній iter. Обчислюється кількість ковзних вікон j_Wind, яка залежить від довжини S0 і параметра n_Wind. Ця кількість вікон визначається так, щоб вікна накривали усі можливі позиції в масиві S0. Створюється масив S0_Wind, який використовуватиметься для зберігання поточного вікна даних. Також створюється масив Midi, який буде використовуватися для зберігання медіан кожного ковзного вікна. Проводимо обхід ковзних вікон: для кожного вікна внутрішній цикл обчислює медіану значень вікна S0_Wind і зберігає її в відповідному елементі масиву Midi. Ця медіана є результатом обробки кожного вікна.

Створюється ще один масив під назвою S0_Midi, який буде використовуватися для зберігання обробленого результату. Вхідний масив S0 копіюється в S0_Midi, за винятком перших n_Wind елементів, які замінюються результатами медіанного фільтру для відповідних ковзних вікон. Після завершення обробки всіх ковзних вікон, функція повертає масив S0_Midi, який містить оброблені дані з використанням методу "ковзного вікна" з медіанним фільтром. Результат роботи зображено на рис 3.7

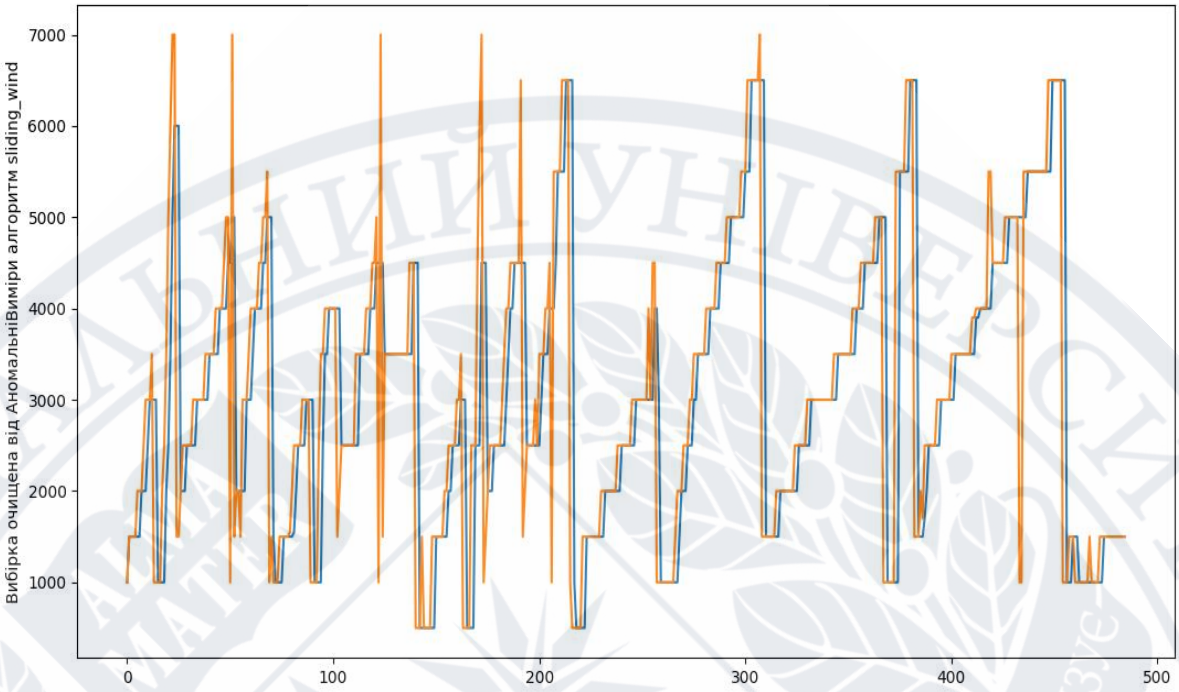

Рисунок 3.7 – Результат методу sliding window

Завдяки цьому методу дані було очищено від аномальних вимірів, та усі подальші дії були виконані з очищеними даними методом sliding window.

Фільтр Калмана (альфа-бетта фільтр).

```
def ABF (S0):
  iter = len(S0)
  Yin = np.zeros((iter, 1))
  YoutAB = np.zeros((iter, 1))
  T0=1
  for i in range(iter):
    Yin[i, 0] = float(S0[i])
  # ---------- початкові дані для запуску фільтра
  Yspeed_retro=(Yin[1, 0]-Yin[0, 0])/T0
  Yextra=Yin[0, 0]+Yspeed_retro
  alfa=2*(2*1-1)/(1*(1+1))
  beta=(6/1)*(1+1)
  YoutAB[0, 0]=Yin[0, 0]+alfa*(Yin[0, 0])
  # ---------- рекурентний прохід по вимірам
```

```
    for i in range(1, iter):
        YoutAB[i,0]=Yextra+alfa*(Yin[i, 0]- Yextra)
        Yspeed=Yspeed_retro+(beta/T0)*(Yin[i, 0]- Yextra)
        Yspeed_retro = Yspeed
        Yextra = YoutAB[i,0] + Yspeed_retro
        alfa = (2 * (2 * i - 1)) / (i * (i + 1))
        beta = 6 /(i* (i + 1))
    return YoutAB
```

Основні кроки в коді:

1. Ініціалізація змінних:

iter: Кількість вимірювань або точок у вхідному сигналі S0.

Yin: Вхідний сигнал (вимірювання) в форматі колонки.

YoutAB: Вихідний сигнал після фільтрації.

2. Ініціалізація параметрів фільтра:

T0: Початковий час між вимірюваннями.

Yspeed_retro: Початкова швидкість (похідна від сигналу) на основі перших двох точок.

Yextra: Додатковий параметр, який використовується для обчислення прогнозованого значення на кожному кроці.

3. Початкове значення фільтра:

Обчислюється початкове значення фільтрованого сигналу YoutAB[0, 0] з використанням альфа-бета параметрів.

4. Рекурентний прохід по вимірам:

За допомогою циклу для кожної точки вимірювання обчислюється нове значення YoutAB[i, 0].

Обчислюється швидкість Yspeed та використовується для оновлення Yextra.

Альфа і бета перераховуються на кожному кроці відповідно до формул в коді результат альфа бета метода зображено на рис 3.8.

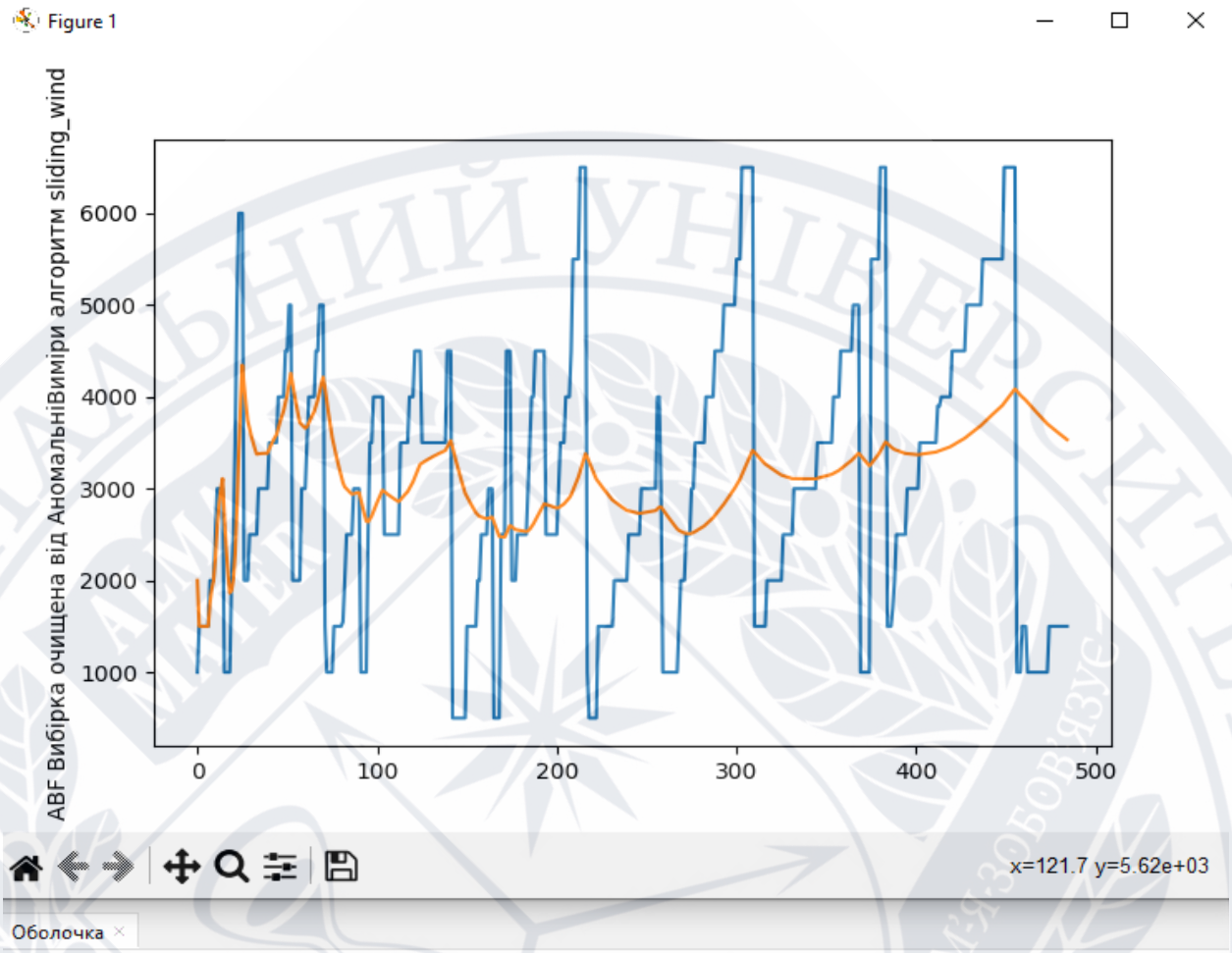

Рисунок 3.8 – Результат методу альфа бета фільтра

Даний метод зберіг основні тренди та забрав шуми, що дозволило отримати більш чіткі та стабільні данні.

Метод найменших квадратів згладжування

```
def MNK_ Smoothing (S0):
  iter = len(S0)
  Yin = np.zeros((iter, 1))
  F = np.ones((iter, 3))
   # формування структури вхідних матриць МетодНайменшихКвадратів
  for i in range(iter):
     # формування матриці
     Yin[i, 0] = float(S0[i])
```

```
        F[i, 1] = float(i)
        F[i, 2] = float(i * i)
    FT=F.T
    FFT = FT.dot(F)
    FFTI=np.linalg.inv(FFT)
    FFTIFT=FFTI.dot(FT)
    C=FFTIFT.dot(Yin)
    Yout=F.dot(C)
    print('Регресійна модель:')
    print('y(t) = ', C[0,0], ' + ', C[1,0], ' * t', ' + ', C[2,0], ' * t^2')
    return Yout
```

Функція MNK_Smoothing(S0) приймає вхідний масив S0, який містить початкові дані, що потребують згладжування.

1. Ініціалізація змінних:

iter - кількість елементів у вхідному масиві S0.

Yin - масив нулів розміром (iter, 1), який буде містити вхідні дані для МНК.

F - матриця розміром (iter, 3), де кожен рядок містить початкові значення 1 для першого стовпця, i для другого стовпця (де i - індекс рядка) та i^2 для третього стовпця.

2. Формування вхідних матриць МНК:

У циклі for проходимо крізь кожен елемент вхідного масиву S0.

Значення кожного елемента S0[i] приводимо до типу float і записуємо в Yin[i, 0].

Заповнюємо стовпці матриці F значеннями i та i^2.

3. Обчислення коефіцієнтів регресійної моделі:

Транспонуємо матрицю F і отримуємо матрицю FT.

Обчислюємо добуток FT і F і отримуємо матрицю FFT.

Обчислюємо обернену матрицю FFT за допомогою np.linalg.inv().

Обчислюємо добуток FFTI і FT і отримуємо матрицю FFTIFT.

Обчислюємо добуток FFTIFT і Yin і отримуємо матрицю C, яка містить коефіцієнти регресійної моделі.

4. Згладжування даних:

Обчислюємо добуток матриці F і C і отримуємо матрицю Yout, яка містить згладжені значення даних.

5. Виведення регресійної моделі:

Виводимо регресійну модель у вигляді y(t) = a + b * t + c * t^2, де a, b, c - коефіцієнти регресії.

6. Повернення згладжених значень даних:

Повертаємо матрицю Yout, яка містить згладжені значення даних. Результат роботи зображено на рис 3.9.

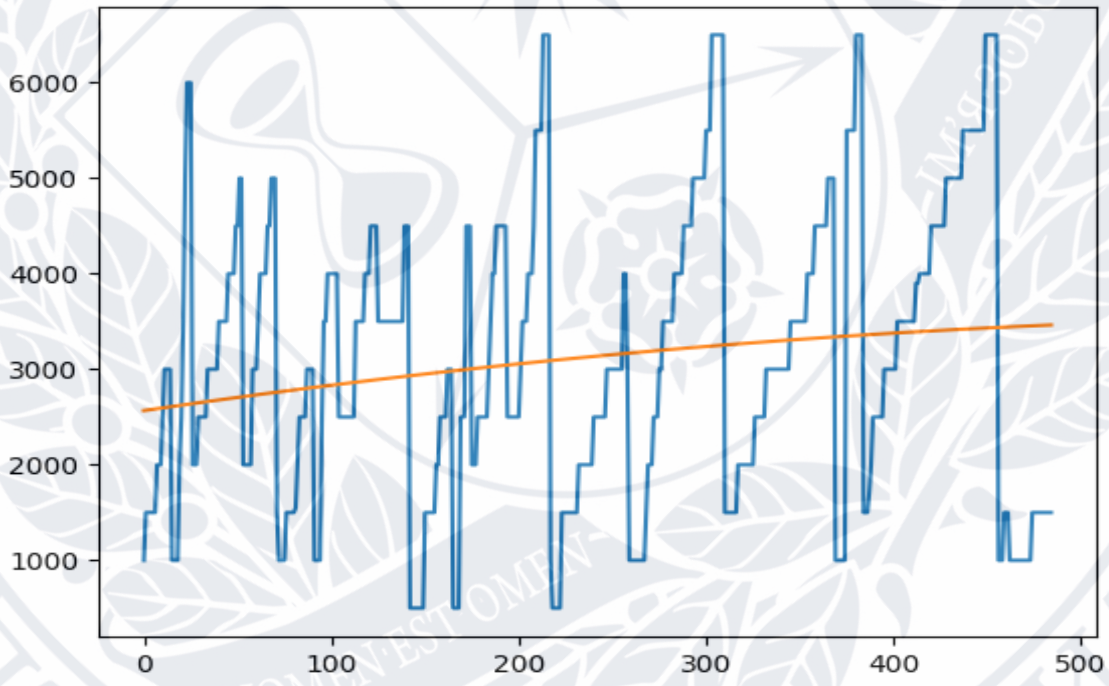

Рисунок 3.9 – Результат МНК згладжування

За даним результатом можна побачити зменшення випадкових, або систематичних коливань, була побудована більш гладка крива, яка полегшує аналіз та робить дані менш схильними до впливу аномалій.

В кінці розробки був реалізований метод найменших квадратів (МНК) для побудови поліноміальної регресійної моделі та проведення прогнозування значень на основі цієї моделі.

```
def MNK_Extrapol (S0, koef):
    iter = len(S0)
    Yout_Extrapol = np.zeros((iter+koef, 1))
    Yin = np.zeros((iter, 1))
    F = np.ones((iter, 3))
    for i in range(iter):
        Yin[i, 0] = float(S0[i])
        F[i, 1] = float(i)
        F[i, 2] = float(i * i)
    FT=F.T
    FFT = FT.dot(F)
    FFTI=np.linalg.inv(FFT)
    FFTIFT=FFTI.dot(FT)
    C=FFTIFT.dot(Yin)
    print('Регресійна модель:')
    print('y(t) = ', C[0, 0], ' + ', C[1, 0], ' * t', ' + ', C[2, 0], ' * t^2')
    for i in range(iter+koef):
# проліноміальна крива МетодНайменшихКвадратів - прогнозування
 Yout_Extrapol[i, 0] = C[0, 0]+C[1, 0]*i+(C[2, 0]*i*i)
    return Yout_Extrapol
```

Першим кроком визначається довжина вхідного масиву S0, який містить дані, які потрібно обробити. Наступний крок полягає в створені декількох масивів, один із них це пустий масив Yout_Extrapol, в якому будуть зберігатися результати екстраполяції. Розмірність масиву визначається як iter + koef, де koef - це кількість додаткових точок, які потрібно екстраполювати. Наступний створений пустий масив це Yin, який буде використовуватися для зберігання вхідних даних. Цей масив заповнюватиметься значеннями із

вхідного масиву S0. Створюється масив F, який представляє матрицю вхідних факторів для метода найменших квадратів. В даному випадку ця матриця має розмірність (iter, 3) і містить першим стовпцем одиниці (для коефіцієнта зсуву в регресійному рівнянні), другим стовпцем - значення i, і третім стовпцем - значення i^2, де i - індекс елемента вхідного масиву S0. Формується транспонована матриця FT і обчислюється FFT - результат множення транспонованої матриці FT на саму себе F. Ця операція необхідна для подальших обчислень МНК. Обчислюємо обернену матриця FFTI для матриці FFT з використанням функції np.linalg.inv з бібліотеки NumPy. Обернена матриця необхідна для обчислення коефіцієнтів регресії. Розраховуємо добуток оберненої матриці FFTI на транспоновану матрицю FT, що необхідно для обчислення коефіцієнтів регресії. Обчислюються коефіцієнти регресії, які представляють параметри поліноміальної моделі. Коефіцієнти зберігаються у матриці C. Виводяться на екран параметри регресійної моделі, що вказують на рівняння регресії у вигляді y(t) = C[0, 0] + C[1, 0] * t + C[2, 0] * t^2. Це рівняння представляє поліном другого ступеня.

Проводиться екстраполяція (прогнозування) значень для iter + koef точок на основі розрахованих коефіцієнтів. Результат роботи зображено на рис 3.10

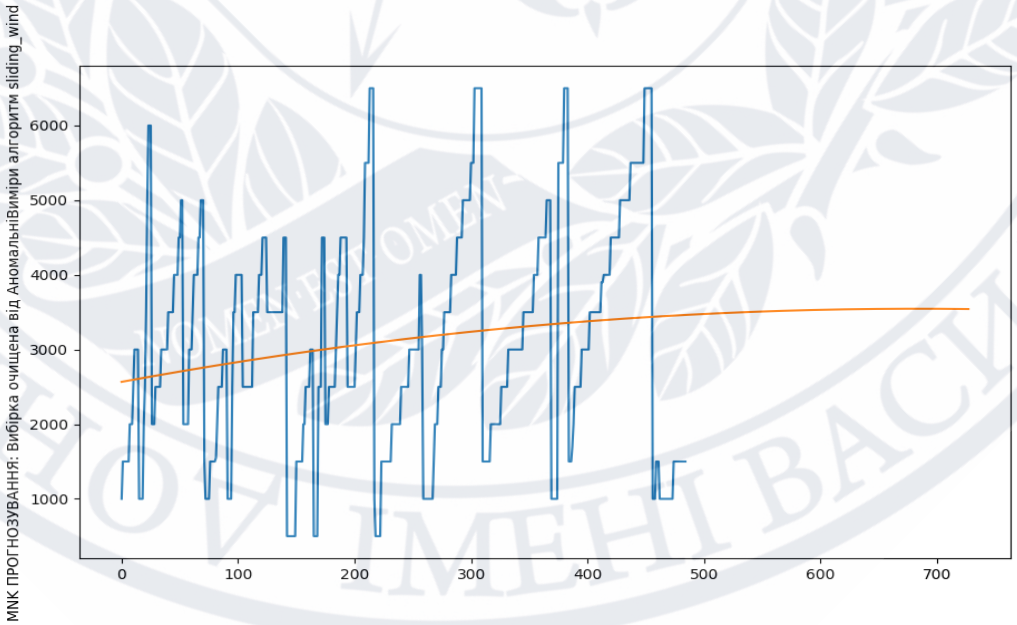

Рисунок 3.10 – результат МНК прогнозування зростання середньої заробітної плати

Застосовуючи метод найменших квадратів було зроблено прогноз на три роки, на отриманому графіку зображено зростаючу криву що свідчить про динаміку зростання середньої заробітної плати тестувальників.

Висновки до розділу

У розділі продемонстровано створену архітектуру програмного забезпечення та розписаний кожен крок роботи з даною програмою. Було наведено графічну візуалізацію даних із DataSet. Продемонстровано очищення DataSet від аномальних вимірів. За методом альфа бета фільтру було збережено основні тренди та забрані шуми, що дозволило отримати більш чіткі та стабільні данні.

Реалізовано метод згладжування, що зменшило випадкові, або систематичні коливання, візуалізовано більш гладка криву, яка полегшує аналіз та робить дані менш схильними до впливу аномалій.

Створення прогнозування на три роки, візуалізація зростаючої кривої, що свідчить про тенденцію зростання заробітної плати тестувальників в Україні.

ВИСНОВКИ

В роботі було створено інформаційну систему, яка аналізує DataSet з інформацією про заробітну плату тестувальників за 2022 рік. Вона була реалізована на мові програмуванні Python та були використані бібліотеки NumPy, math, matplotlib, pandas, scriPy.

Для обробки даних були використані такі методи:

1. Методи найменших квадратів;
2. Метод медіан;
3. Метод sliding window.

Ці методи знаходять та очищують аномальні виміри. Наступні кроки роботи уже використовували дані очищені методом sliding window для отримання більш точніших висновків. Завдяки альфа бета фільтрації були збережені основні тренди та очищено від шуму, що зменшило коливання на графіку та візуально спростило початковий графік. За методом найменших квадратів згладжування були зменшені коливання. Отримано криву для полегшення аналізу даних та зменшено вплив аномальних вимірів.

Останній крок завдання полягав у створенні прогнозування на три роки. На базі аналізу DataSet заробітної плати тестувальників за 2022 рік було побудовано прогнозування, яке свідчить про збільшення заробітної плати тестувальників.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Заробітна плата URL: https://www.kadrovik.ua/content/zarobitna-plata-organizaciyniy-mehanizm-formuvannya-struktura-harakteristika-ta-poryadok (дата звернення 15.06.2023)
2. Нарахування заробітної плати в Україні URL: https://www.accounting-ukraine.kiev.ua/poslugi/oblik_narahuvannya_zarobitnoyi_plati_zarplati.htm (дата звернення 15.06.2023)
3. Податок на доходи URL: https://tax.gov.ua/nk/rozdil-iv--podatok-na-dohodi-fizichnih-o/ (дата звернення 15.06.2023)
4. Болдирева, Л.І., Бурлакова, І.Д., Костромін, О.П. Інформаційні технології та системи. Київ, 2007 С. 14
5. Декомпозиція URL: https://foxminded.ua/dekompozytsiia-v-prohramuvanni/ (дата звернення 15.06.2023)
6. Мова Python URL: https://freehost.com.ua/ukr/faq/wiki/chto-takoe-jazik-programmirovanija-python/ (дата звернення 17.06.2023)
7. Thonny URL: https://thonny.org (дата звернення 17.06.2023)
8. Роберт Мартин. Чиста архітектура. Мистецтво розробки програмного забезпечення США, С 130-134
9. Фільтр Калмана URL: https://ukrayinska.libretexts.org/Інженерна/Машинобудування/Вступ_до_автономних_роботів_(Correll)/09%3A_Локалізація/9.04%3A_Фільтр_Калмана (дата звернення 20.06.2023)
10. Енциклопедія сучасної України URL: _ https://esu.com.ua/article-67449 (дата звернення 20.06.2023)
11. Дисперсія випадкової величини URL: http://pms.ptngu.com/page%27s/l11.html (дата звернення 20.08.2023)
12. Організаційне забезпечення URL: https://studfile.net/preview/5064248/page:11/ (дата звернення 20.06.2023)
13. Буйницька О. П.. Інформаційні технології та технічні засоби навчання. Київ, 2007 С. 100-200

14. Мистецтво статистики. Як знаходити відповіді у даних URL: https://flibusta.su/book/9639-iskusstvo-statistiki-kak-nahodit-otvetyi-v-dannyih/read/ (дата звернення 20.06.2023)
15. Грицунов О. В.. Інформаційні системи та технології. Харків, ХНАМГ 2010 С. 123-189
16. Peter Bruce, Andrew Bruce. Практична статистика для спеціалістів Data Science. Boston 2018. С. 150-200
17. Алекс Дж. Готтман, Джордан Голд Мейє. Розберися в Data Science. Як освоїти науку про дані та навчитися думати як експерт 2021. С.50-100
18. Класифікація інформаційних технологій URL: https://studfile.net/preview/5064248/page:11/ (дата звернення 20.06.2023)
19. К. Еременко. Робота з даними у будь-якій сфері. Альпина Диджитал, 2018 С. 3-26
20. Інформаційне забезпечення URL: https://library.if.ua/book/80/5658.html (дата звернення 20.06.2023)
21. Основи статистики та аналізу даних URL: https://socialdata.org.ua/manual/manual4/#види-шкал-та-змінних (дата звернення 20.06.2023)
22. Технічне забезпечення URL: https://buklib.net/books/22551/ (дата звернення 20.06.2023)
23. Правове забезпечення URL: https://studfile.net/preview/5064248/page:11/ (дата звернення 20.06.2023)
24. Математичне забезпечення URL: https://pidru4niki.com/1830100247741/informatika/matematichne_zabezpechennya (дата звернення 20.06.2023)
25. Програмне забезпечення URL: https://buklib.net/books/22551/ (дата звернення 20.06.2023)
26. Класифікація інформаційних систем URL: https://studfile.net/preview/5064248/page:12/ (дата звернення 20.06.2023)

27. Лінгвістичне програмне забезпечення URL: http://ni.biz.ua/2/2_6/2_6127_lingvisticheskoe-obespechenie.html (дата звернення 20.06.2023)
28. Стандартні методи кластеризації даних URL: http://csc.knu.ua/media/study/asp/mod_probl_inf_tech_sys_analysis_ivohin/lecture/lec2.pdf (дата звернення 20.06.2023)
29. Метод найменших квадратів URL: http://physics.zfftt.kpi.ua/repository/coursefilearea/file.php/1/Labs/МЕТОД_НАЙМЕНШИХ_КВАДРАТІВ.pdf (дата звернення 20.06.2023)
30. Розберися в Data Science. Як освоїти науку про дані та навчитися думати як експерт URL: https://flibusta.su/book/150150-razberis-v-data-science-kak-osvoit-nauku-o-dannyih-i-nauchitsya-dumat-/ (дата звернення 25.06.2023)
31. Stochastic Gradient Descent URL: https://scikit-learn.org/stable/modules/sgd.html#stochastic-gradient-descent (дата звернення 25.06.2023)
32. Кравець І. Big Data Analytics: Від теорії до практики. Таллін 2018. С. 140-200
33. Мельник С. Python для аналізу даних та машинного навчання. Дніпро, 2019 С. 100-150
34. Григорчук О. Статистика для бізнесу: Застосування у прийнятті рішень. Одеса, 2021 С. 100-180
35. Сидоренко В. Машинне навчання та аналіз даних з використанням R. Харків. 2016 С. 234-250
36. Марія Іваненко М. Data Science: Від теорії до практики. Кишинів, 2017 С. 3-20
37. Петров О. Введення в аналіз даних за допомогою Python. Львів, 2019 С. 170-200
38. Іванова А. Основи статистики: Від теорії до практики. Київ, 2018 С. 130-170

39. Соколова Н. Ефективний аналіз даних з використанням Pandas та NumPy. Львів, 2022 С. 210-250
40. Ковальчук Д. Методи машинного навчання для аналізу часових рядів. Харків, 2019 С. 190-230
41. Шевченко Н. Ефективний аналіз даних: Від ідеї до висновку. Київ, 2022 С. 160-230
42. Веса МакКінні Python для аналізу даних. США, 2012 С. 132-169
43. Фернандес Л. Практична наука про дані з використанням Python. Барселона, 2018. С 20-132
44. МакКінні В. Python для аналізу даних. Нью-Йорк, 2012. С 34-122
45. Вандер Плас Дж. "Python для аналізу даних: обробка, візуалізація і знаходження висновків". Сан-Франциско, 2016. С. 122-162
46. Хейкенс Дж. "Статистика: зрозуміти світ даних". Лондон, 2019. С. 44-92
47. Бекер Дж., Босвелл А., Снайдер Дж. Практична наука про дані: Вивчайте відкриті дані за допомогою Python, R та SQL. Сан-Франциско, 2019. С. 64-100
48. Аббас І. Основи аналізу даних: З використанням Python. Лондон, 2017. С. 100-124
49. Шет Ш., Дарвін К., Вільямс Т. Введення в машинне навчання з використанням Python: Підручник для інженерів. Нью-Йорк, 2017. С. 43-121
50. Дюплессі Д. Повний посібник з інтелектуального аналізу даних з використанням Python. Лондон, 2017. С. 58-143

Додаток 2 до наказу
від «31» березня 2023 року
№119/05

ДЕКЛАРАЦІЯ

про дотримання академічної доброчесності

Я, _____

_____

Повністю вказується ПІБ та статус (посада для працівників, освітня (освітньо-наукова) програма – для здобувачів вищої освіти)

що нижче підписалась/підписався, розуміючи та підтримуючи загальновизнані засади справедливості, доброчесності та законності,

ЗОБОВ'ЯЗУЮСЬ:

дотримуватися принципів та правил академічної доброчесності, що визначені законодавством України, локальними нормативними актами Донецького національного університету імені Василя Стуса, положеннями, правилами, умовами, визначеними іншими суб'єктами, та не допускати їх порушення.

ПІДТВЕРДЖУЮ:

що мені відомі положення статті 42 Закону України «Про освіту»;

що у даній роботі не представляла/представляв чиїсь роботи повністю або частково як свої власні. Там, де я скористалася/скористався працею інших, я зробила/зробив відповідні посилання на джерела інформації;

що дана робота не передавалась іншим особам і подається вперше, не порушує авторських та суміжних прав закріплених статтями 21-25 Закону України «Про авторське право та суміжні права», а дані та інформація не отримувались в недозволений спосіб.

УСВІДОМЛЮЮ:

що ця робота може бути перевірена університетом на плагіат або інші порушення академічної доброчесності, в тому числі з використанням спеціалізованих сервісів;

що у разі порушення академічної доброчесності, до мене можуть бути застосовані процедури, передбачені законодавством України та Кодексом академічної доброчесності та корпоративної етики Донецького національного університету імені Василя Стуса, іншими локальними нормативними актами університету, та я можу бути притягнута/притягнутий до академічної відповідальності.

_____ _____

(дата) (підпис)