Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем та програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавра
(назва освітнього ступеня)

на тему: «Розробка алгоритму обробки даних захворюваності на COVID-19 з візуалізацією даних у Python»

Виконав(ла): студент(ка) IV курсу, групи СП-41
спеціальності 121
«Інженерія програмного забезпечення»
(шифр і назва спеціальності)

_____ Ковцун І. А.
(підпис) (прізвище та ініціали)

Керівник _____ _____
(підпис) (прізвище та ініціали)

Нормоконтроль _____ _____
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ _____
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль 2023

РЕФЕРАТ

Кваліфікаційна робота студента- бакалавра Ковцуна І. А., Тернопільський національний технічний університет імені Івана Пулюя, кафедра програмної інженерії, спеціальність 121 «Інженерія програмного забезпечення». ТНТУ, 2023. Сторінок: ___, таблиць: ___, рисунків: ___, додатки: ___.

Тема роботи: Розробка алгоритму обробки даних захворюваності на COVID-19 з візуалізацією даних у Python.

Завданням даної розробки є проектування та розробка алгоритму обробки даних медичного спрямування, а саме обробка статистичних накопичених людством за період пандемії даних по захворюваності на COVID-19 з можливістю візуалізувати програмним способом. Під розробкою системи розуміється програма, яка буде отримувати дані від сервісів або зовнішніх служб (або ж від інших систем), обробляти їх та надавати інформацію у відповідь.

В кваліфікаційній роботі висвітлено постановку завдання щодо виконання проєкту, аналіз предметної області та сучасних напрацювань науки про дані.

В другому розділі описано етапи проектування та конструювання програмної системи з описом технологій та підходів до розробки.

В Третьому розділі описано етапи реалізації та кодингу програми, а також елементи тестування та впровадження системи.

В спеціальному розділі враховано питання вимог охорони праці з ціллю збереження здоров'я при роботі за комп’ютерною технікою та пожежна безпека при улаштуванні електроустановок.

За результатами виконання роботи отримано програмну систему для відкритої та прозорої співпраці медичних та наукових закладів із існуючими потужними базами знань та даних про захворювання COVID-9, а також розроблено базову модель нейронної мережі для перевірки роботи системи.

Ключові слова: ОХОРОНА ЗДОРОВ’Я, МАШИННЕ НАВЧАННЯ, ОБРОБКА ДАНИХ, ЗБІР МЕДИЧНИХ ДАНИХ, PYTHON, COVID-19.

ANNOTATION

Certification work of the student I. A. Kovtsun Ternopil Ivan Puluj National Technical University, Department of Software Engineering, specialty 121 "Software engineering". TNTU, 2023. Pages: ___, tables: ___, figures: ___, annex: ___.

Work topic: Development of an algorithm for processing data on the incidence of COVID-19 with data visualization in Python.

The task of this development is the design and development of a medical data processing algorithm, namely the processing of statistical data on the incidence of COVID-19 accumulated by humanity during the pandemic period with the possibility of software visualization. System development means a program that will receive data from services or external services (or from other systems), process them and provide information in response.

In the qualification paper, the statement of the task regarding the implementation of the project, the analysis of the subject area and modern developments in data science are highlighted.

The second chapter describes the stages of designing and constructing a software system with a description of technologies and development approaches.

The third section describes the stages of implementation and coding of the program, as well as elements of testing and implementation of the system.

In a special section, the issue of occupational health and safety requirements for the purpose of health preservation when working with computer equipment and fire safety when installing electrical installations is taken into account.

Based on the results of the work, a software system was obtained for open and transparent cooperation of medical and scientific institutions with the existing powerful knowledge and data bases about the COVID-9 disease, as well as a basic neural network model was developed to test the system.

Keywords: HEALTH CARE, MACHINE LEARNING, DATA PROCESSING, MEDICAL DATA COLLECTION, PYTHON, COVID-19.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

COVID-19 – інфекційна хвороба , спричинена вірусом важкого гострого респіраторного синдрому коронавірус 2, швидко поширилася по всьому світу, що призвело до пандемії.

Візуалілазція даних – один з напрямів науки про дані, вміння представити, візуалізувати результати свого аналізу так, щоб це найкраще сприймалось.

Мова Python – це мова програмування високого рівня загального призначення, підтримує кілька парадигм програмування , включаючи структуроване (зокрема процедурне), об’єктно-орієнтоване та функціональне програмування.

NumPy – це бібліотека для мови програмування Python , яка додає підтримку для великі багатовимірні масиви та матриці разом із великою колекцією математичних функцій високого рівня для роботи з цими масивами.

Machine Learning – це сфера, присвячена розумінню та розробці методів, які дозволяють машинам «навчатися», які використовують дані для покращення продуктивності комп’ютера.

UML (Unified Modeling Language) – уніфікована мова моделювання яка використовується при розробці ПЗ.

Software architecture – архітектура програмного продукту. Це структура програми або обчислювальної системи, яка включає програмні компоненти, а також відносини між ними.

Проектування ПЗ – це процес визначення архітектури, набору компонентів, їх інтерфейсів, інших характеристик системи і кінцевого складу програмного продукту.

Вимоги – це умови, можливості, які повинні задовольнятися системою, компонентом системи, або якими система/компонент системи повинна володіти для забезпечення умов контракту, стандартів, специфікацій чи інших регулюючих документів.

ЗМІСТ

ВСТУП

Предметна область даної роботи знаходиться на межах медицини та машинного навчання. Використання машинного навчання у цілях автоматизації роботи медичного персоналу є доволі поширеним та ефективним методом впровадження інформаційних технологій в охороні здоров'я.

Завданням даної розробки є проектування та розробка алгоритму обробки даних медичного спрямування, а саме обробка статистичних накопичених людством за період пандемії даних по захворюваності на COVID-19 з можливістю візуалізувати програмним способом. Слід розробити систему, яка буде отримувати дані від сервісів або зовнішніх служб (або ж від інших систем), обробляти їх та надавати інформацію у відповідь. Визначним фактором при розробці архітектури системи в даному випадку є потік даних. Оскільки система є орієнтованою на дані, аналіз слід починати з огляду їх джерел і видів.

Мета роботи: проаналізувати вимоги, можливості та використання системи аналізу глобальних даних про захворювання COVID-19 для спостереження за тенденціями та викликами. За допомогою методів розробки ПЗ спроектувати простий та ефективний спосіб обробки та візуалізації даних, який допоможе слідкувати та змінами в світових показниках захворювання.

Візуалізація даних – це велика сфера. Існує багато інших бібліотек Python для візуалізації даних, як-от ggplot, plotly та seaborn. Ці бібліотеки надають розширені можливості візуалізації даних, які можуть задовольнити ваші ділові та наукові вимоги візуалізації.

Розробка програмної системи поділяється на декілька частин, а саме: розробка алгоритму логіки обробки даних, розробка графічного інтерфейсу користувача з візуалізацією даних, розробка логіки програми.

Представлена нижче кваліфікаційна робота описує процес та основні етапи створення програмного забезпечення підтримки наукових досліджень по обробці даних захворювання COVID-19 з можливістю візуалізації та елементами машинного навчання.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області та використовуваних методів

З самої появи алгоритмів штучного інтелекту, машинне навчання застосовувалось у аналізі медичних наборів даних. Сьогодні машинне навчання є незамінним інструментом аналізу даних, що стало можливим через впровадження технологій зберігання цифрових даних. Особливо позитивно це впливає на медичні установи, що накопичують чималі обсяги даних, які можна застосувати для навчання. Штучний інтелект в цьому випадку дозволяє вирішувати специфічні діагностичні проблеми.

На момент написання роботи світ зіткнувся з безпрецедентним стихійним лихом, пандемією (інфекційною хворобою, що поширюється на континентах), спричиненою важким гострим респіраторним синдромом, коронавірусом (SARS-CoV-2), і викликаною ним хворобою, відомою як COVID 19. Цей вірус походить з того самого сімейства вірусів (коронавірусів), які викликають нелетальні захворювання, як-от звичайна застуда, і більш смертоносні захворювання, як-от SARS і MERS. Ці віруси викликають інфекції дихальних шляхів у ссавців і людей і можуть бути летальними, якщо їх не лікувати вчасно.

Інформатизація та автоматизація будь-яких процесів є невід’ємною частиною розвитку людства на даному етапі розвитку. Все більше галузей застосовують інформаційні технології для покращення ефективності їх роботи, зменшення виробничих витрат, підвищення кваліфікації та полегшення роботи персоналу.

Застосування мови програмування Python та її дистрибутивних розширень у різних предметних областях інформаційних технологій і зокрема інженерії програмного забезпечення триває вже понад 15 років. Зокрема ця мова програмування застосовується у різних сферах технічної діяльності людини, для різних завдань, таких як автоматизація, графіка, Інтернет речей та наука особливо у науці про дані. Засоби мови Python є дуже зручним і вільно поширюваним інструментом для створення наукових візуалізацій бізнесу, заснованих на аналізі

та візуалізації масивів даних різної природи. Для створення візуалізацій за допомогою Python потрібно менше рядків коду, оскільки Python може отримувати дані з різних джерел. Поєднання цієї функції з різними бібліотеками візуалізації робить Python ідеальним інструментом для різних типів вимог до візуалізації, разом із цим в загальному випадку включаючи налаштування та різні режими, а також безліч бібліотек візуалізації. Задачі з аналізу даних та їх подальшої обробки та візуалізації часто виникають у міждисциплінарних дослідженнях, таких як для прикладу медична інформатика. Робота у міждисциплінарних предметних областях спричиняє вимогу вивчити більше бібліотек для візуалізації даних у Python, оскільки більшість з них підтримує наукову екосистему Python і бібліотеку NumPy, що є основами при роботі з даними. Крім того, для візуалізації отриманих результатів із обробки дуже важливим моментом є застосування методів бібліотеки matplotlib.pyplot.

В даній роботі розроблено методологію візуалізації актуальних даних медичної статистики, що пов'язані з пандемією COVID-19, що тривала до останнього часу як в Україні, так і у світі загалом.

Для реалізації візуалізації даних за допомогою Python застосовано засоби середовища Jupyther в якому здійснено розробку моделювання відповідного програмного забезпечення та його безпосередню розробку.

Важливими вимогами предметної області є безпека персональних даних, зокрема у зв’язку із статтею 286 Цивільного Кодексу України, згідно з якою фізичні особи мають право на таємницю про їх стан здоров’я, факт звернення до медичної установи та відомості, одержані від них при медичному обстеженні.

Згідно із специфікою даних які планується збирати, а також способом їх подальшого використання, набори даних повинні бути ідентифіковувані, а система повинна надавати можливість за даними ідентифікаторами отримувати вихідний набір даних. Завдяки цьому попередньо згенеровані набори даних можна поширювати, а також робити на них посилання.

Програмна система повинна обробляти медичні дані, що надходитимуть від персоналу різних медичних установ. Споживачами даних є дослідницькі організації, тож слід забезпечити відповідне форматування даних при їх видачі.

Для процесу діагностики планується використовувати різні методи машинного навчання, проте для початку слід зосередитись на одному з них, забезпечивши методи розширення для імплементації решти алгоритмів в майбутньому.

Для досягнення цілей магістерської роботи, було прийнято рішення розробити мікросервісну архітектуру з підтримкою масштабованості та розроділеного розгортання. Таким чином ланки програмної системи можна модифікувати та заміняти на рівні мікросервісів

1.2 Наукова екосистема Python та NumPy

Як відомо, найпростішою методологією того, як створювати прості візуалізації за допомогою Python 3 є бібліотека візуалізації даних leather. Проте зрозуміло, що за допомогою бібліотеки візуалізації даних про leather можна підготувати лише примітивні візуалізації. Для більш складних та опрацьованих візуалізацій нам необхідно використовувати бібліотеки з розширеними можливостями обробки даних та візуалізації. Для таких речей безпосередньо використовується наукова екосистема Python та її компоненти. Зробимо короткий огляд бібліотеки NumPy з демонстраціями коду, який використовується для візуалізації масивів даних. Для цього ми розглянемо такі основні аспекти:

- Наукова система Python;
- NumPy та ndarrays;
- Властивості Ndarray;
- Константи NumPy.

Розглянемо метод дослідження багатьох компонентів наукової екосистеми Python у зв’язку один за одним. У дипломній роботі будуть використовуватися різні бібліотеки, які є частиною наукової екосистеми Python.

Аналіз структури використовуваних методів, стануть основою розробки системи, поданої у дипломній роботі. Тут ми приділятимемо основну увагу практичній імплементації розглядуваних методів та їхній структурний зміст.

Наукова екосистема Python (SciPy) — це набір бібліотек Python для математики, науки та інженерії. SciPy має такі основні компоненти:

1. Мови програмування Python: ми інсталяція окремих модулів та бібліотек орієнтовані під застосування синтаксису мови Python 3.

2. NumPy: це чисельна бібліотека Python, основний пакет для чисельних обчислень на Python. Він визначає N-вимірний тип даних, який можна використовувати для числових обчислень на багатовимірних даних.

3. Бібліотека SciPy: включає багато процедур для математичних і наукових обчислень, які можна використовувати для наукових додатків.

4. Matplotlib: це бібліотека на основі MATLAB для візуалізації даних у Python 3.

Є кілька додаткових важливих бібліотек-додатків цієї екосистеми.

1. Pandas: це визначає набір засобів аналізу даних Python. Він надає універсальні структури даних, такі як ряди та кадри даних.

2. SymPy: бібліотека символічних обчислень у Python. Він використовується в основному для символічної математики та алгебри.

3. Scikit-image: у цій бібліотеці містяться процедури обробки зображень.

4. Scikit-learn: у цій бібліотеці містяться підпрограми для машинного навчання.

Інтерактивні середовища, які зазвичай використовуються з SciPy, це IPython або Jupyter Notebook. В подальшому ми будемо Jupyter Notebook застосовувати в процесі розробки системи, в подальшому розглядаючи особливості бібліотеки NumPy.

1.3Джерела даних для виконання обчислень статистичної вибірки

Поверхневе дослідження існуючих систем та аналогів серед відкритих медичних даних та ресурсів не виявило повних даних що походять з українських інституцій. Це свідчить про явну необхідність впровадження інструменту збору та

поширення таких даних, що позитивно вплине на розвиток інформаційних технологій в області медицини.

Серед доступних сервісів джерел даних, найбільш близьким до ідеї розроблюваної програмної системи є UCI Machine Learning Repository.

Даний репозиторій містить набори даних, в тому числі й медичних, які готові до використання у моделях машинного навчання.

Даний репозиторій є колекцією баз даних, доменів і генераторів даних, що використовується спільнотою машинного навчання для емпіричного аналізу різних алгоритмів машинного навчання. Архів було створено як FTP сервіс у 1987 році Девідом Аха та випускниками Каліфорнійського Університету Ірвін. З цього часу репозиторій використовується студентами, викладачами та дослідниками з усього світу як головне джерело даних для машинного навчання. Архів має досить серйозний вплив, адже ог обуло офіційно процитовано більше 1000 разів, що робить його одним з поміж сотні набільш цитованих інтернет публікацій у області комп’ютерних наук [1].

Також якісні дані для машинного навчання можна отримати із урядового порталу Австралії [2], що надає зручний доступ до даних, що можуть використовуватись у дослідженнях. Даний портал дозволяє шукати, отримувати доступ та повторно використовувати публічні набори даних, джерелом яких є урядові організації. Основною ціллю порталу є заохочення публічного доступу та повторного використання даних. Він був створений у відповідності до декларації Відкритого Уряду. Разом із публічно відкритими даними, портал також містить дані від різних інституцій, доступ до яких можна придбати.

Ще одним сервісом, що походить із Австралії, є ANDS [3] – Australian National Data Service. Даний сервіс повстав із співпраці Університету Монаш, Австралійського Національного Університету та Науково-Промислової Дослідницької Організації (CSRIO). Даний сервіс має на меті зробити дослідницькі набори даних більш цінними для дослідників та дослідницьких організацій. Він дозволяє знаходити, отримувати доступ та використовувати дані інших австралійських дослідницьких організацій та урядових агенцій.

Дослідження існуючих рішень показує зацікавленість деяких інституцій державного рівня у заохоченні дослідників, зокрема у області інформаційних технологій та медицини. Суттєвим фактором у дослідженнях такого роду є саме доступність даних, від якості яких може залежати успішність досліджень в цілому.

Джерела даних для даних про COVID-19. Багато організацій відстежують випадки COVID-19 у всьому світі та періодично оновлюють дані на своїх веб-сайтах і веб-сервісах. Найвідомішими є Університет Джонса Гопкінса (https://coronavirus.jhu.edu/map.html) і World-O-Meter (https://www.worldometers.info/coronavirus/).

Це дуже надійні джерела даних щодо COVID-19, і вони дуже часто оновлюють свою статистику (принаймні один раз на 24 години), щоб системи, що перебувають на наступному рівні, отримували найновіші дані.

Отримати ці дані можливо за допомогою спеціальних бібліотек у Python. Одну з таких бібліотек можна знайти за адресою https://ahmednafies.github.io/covid/. Він може отримувати дані як з Університету Джона Гопкінса, так і з World-O-Meter.

2. ПРОЄКТУВАННЯ ТА МОДЕЛЮВАННЯ ПЗ ОБРОБКИ ДАНИХ З ВИКОРИСТАННЯМ PYTHON

2.1 Вибір методології та підходу до розробки проєкту

Аби досягнути успіху та результатів розробки програмного продукту слід якісно організувати сам процес виконання проєкту. Вірно підібраний підхід до процесу виконання проєкту та якісний життєвий цикл розробки програмного продукту дозволяє розробнику або команді ефективно працювати та виконувати поставлені замовником або лідом функції.

Застарілий «водоспадний» метод розробки програмного забезпечення залишає бажати кращого. Процес «планувати, проектувати, будувати, тестувати, поставляти» добре працює для створення автомобілів або будівель, але не так добре для створення систем програмного забезпечення. У бізнес-середовищі, де апаратне забезпечення, попит і конкуренція є змінними, що швидко змінюються, надається перевага гнучкості.

Гнучка методологія – Agile доповнена дванадцятьма принципами гнучкого програмного забезпечення , ця філософія стала універсальним і ефективним новим способом управління проектами. Визначальним чинником гнучких методів є роботодтаний продукт на ранніх термінах виконанняпродукт. Завдяки безпосередньому спілкуванню виконавців із замовниками, гнучкі методи зменшують обсяг письмової документації в порівнянні з іншими методами.

Agile швидко став галузевим стандартом управління проектами. За оцінками, близько 95 відсотків організацій у тій чи іншій формі прийняли Agile . У той же час залишилося багато роботи, щоб практика зріла.

Переваги Agile безпосередньо пов’язані з його швидшим, легшим і активнішим мисленням. Цей процес забезпечує те, що клієнт хоче, коли він цього хоче. Набагато менше витрачається часу на розробку в неправильному напрямку, і вся система швидше реагує на зміни. Щоб отримати повний список переваг, перегляньте цю публікацію . Швидший життєвий цикл розробки програмного забезпечення означає менше часу між оплатою та отриманням.З Agile клієнти не

чекають місяцями чи роками, щоб отримати саме те, чого вони не хотіли. Натомість вони дуже швидко отримують ітерації чогось дуже близького до того, що вони хочуть. Система швидко налаштовується, щоб удосконалити успішне клієнтське рішення, адаптуючись у міру зміни загального середовища.

Також, завдяки залученню замовника не лише до етапів висування вимог і доставки, проект залишається відповідним завданням і відповідає потребам замовника на кожному кроці. Це означає, що менше повертається назад і менше часу, коли ми виконуємо роботу, і тим часом, коли замовник пропонує зміни.

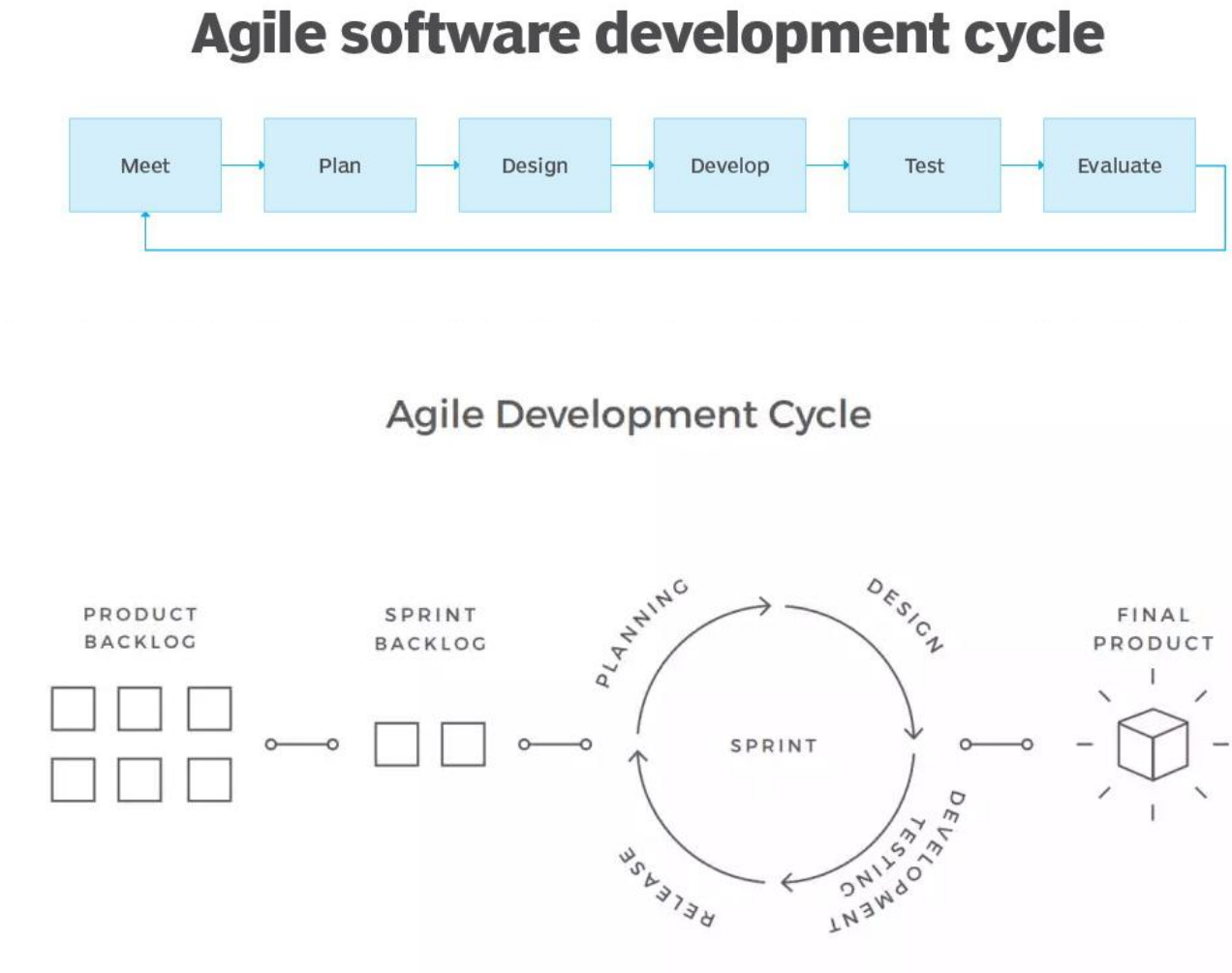

Рис. 2.1 – Життєвий цикл процесу розробки програми згідно методології Agile

Гнучкі методології використовують ітеративний підхід до розробки програмного забезпечення. На відміну від простої лінійної моделі водоспаду,

гнучкі проекти складаються з кількох менших циклів – спринтів. Кожен із них — це проект у мініатюрі: він має відставання та складається з етапів проектування, впровадження, тестування та розгортання в рамках заздалегідь визначеного обсягу робіт.

Управління конфігураціями полягає у забезпеченні доступу до артефактів проекту, розробки, відслідковуванні їх версій в системах управління версіями, керування змінами. Менеджмент проєкту полягає у розподілі відповідальностей всередині проекту, управлінні ризиками, уточненні та керуванні завданнями. Основою оточення є елементи забезпеченні команди інструментами (програмним та апаратним забезпеченням) і слідкуванні за дотриманням стандартів.

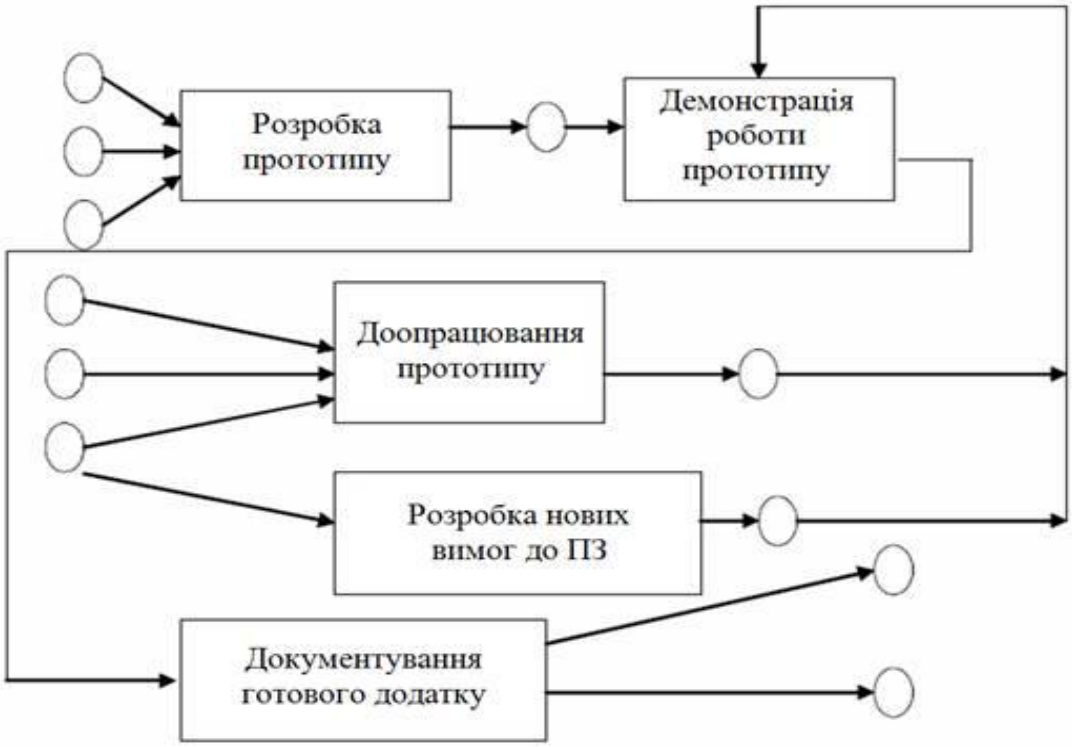

Рис. 2.2 – Ітераційний розвиток прототипу проєкту на ЖЦ розробки

У архітектурному рішенні було вказано, що дане розроблюване ПЗ маю бути робочим додатком для комп’ютера, тобто використовуватиметься науковим працівником у лабораторії з ПК. Запускатися дана програма повинна з операційної системи та працювати у її межах та її засобами. Тобто доступ до WEB та мережі може і не бути, хоча буде доцільним для обміну результатами.

2.2 Проєктування користувачів та ВВ системи

Розглянута система планується для використання в лабораторії для досліджень. Згідно вимог це робочий ПК додаток для отримання результатів обчислень на основі певної моделі. Тому достатньо припустити, що система не має складних архітектурних рішень в плані проектування багатокористувацьких можливостей, ролевості та різноцільової аудиторії. Система зосереджена на специфічну аудиторію, що займається відповідними розробками. Тобто в роботі з системою буде присутній лише один актант – науковий працівник або асистент, що і виконуватиме всі відведені в можливостях системи заявленим завдання, які і відповідають варіантам використання даної програми.

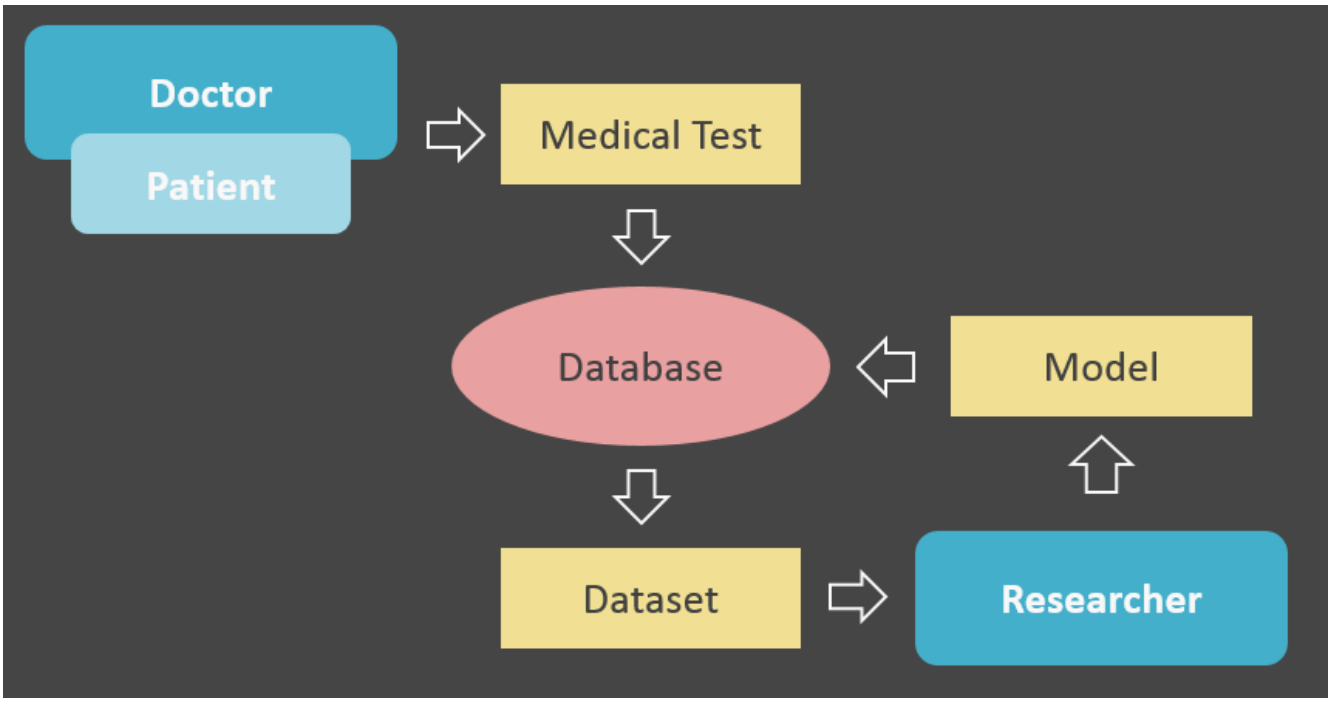

Рис. 2.3 – Data flow системи

До варіантів використання системи користувачем віднесено: зміну параметрів математичної моделі, зміну параметрів шарів накладання модельованих складових, збереження поточних пареметрів моделювання, імпорт раніше збережених результатів моделювання, вибір параметра представлення порівняння моделей.

Дані варіанти використання представлені на рисунку 2.4.

Рис. 2.4 – Діаграма варіантів використання розроблюваноїсистеми

2.3 Конструювання програмної системи

Розподіл на модулі/підсистеми найкраще робити виходячи з тих завдань, які вирішує система. Основне завдання розбивається на складові для нього підзадачі, які можуть вирішуватися/виконуватися незалежно один від одного. Кожен модуль відповідає за вирішення якоїсь підзадачі і виконувати відповідну їй функцію. Крім функціонального призначення модуль характеризується також набором даних, необхідних йому для виконання його функції, тобто:

Модуль = Функція + Дані.

Таким чином, грамотна декомпозиція ґрунтується, насамперед, на аналізі функцій системи і необхідних для виконання цих функцій даних. Це яскраво відображається навіть в структурі проекту, тому що на кожну сутність таблицю створений окремий клас, що буде утримувати дані для операцій в цій сутності (рисунок 2.4).

2.4 Вибір середовища розробки

Після вибору технології, а відповідно і мови Python, слід обрати середовище розробки. Серед доступних середовищ розробки було обрано до розгляду наступні:

- Jupyter Notebook;
- PyCharm;
- Visual Studio Code;
- Python IDLE;
- Spyder.

Для зручності взаємодії з набором бібліотек для обробки даних та візуалізації було обрано середовище виконання проєкту Jupyter Notebook, яке відрізняється відкритим вихідним кодом. Цей інструмент дуже люблять дата аналітики: у ньому є можливість очищувати та перетворювати дані, створювати гарні візуалізації та ділитися документами. Крім того, для коду на Python у редакторі є розмітка та інші круті та зручні функції.

NumPy та Ndarrays. Серед технологій які дозволяють імплементувати таку систему можна виділити NumPy, Ndarrays.

У цьому розділі представлені способи використання NumPy та ndarrays. Бібліотека NumPy велика і включає безліч підпрограм. Для наших цілей поставлених у дипломній роботі, ми розглянемо додаткові підпрограми з бібліотеки NumPy за необхідності для наших демонстрацій візуалізації.

NumPy - це основний пакет для числових обчислень у Python. Найбільш корисною функцією бібліотеки NumPy є структура даних багатовимірного контейнера, відома як ndarray. Ndarray - це багатовимірний масив (також відомий як контейнер) елементів, що мають однаковий тип даних та розмір. Ми можемо визначити розмір та тип даних елементів під час створення ndarray. Як і у випадку з іншими структурами даних, такими як списки, ми можемо отримати доступ до вмісту ndarray за допомогою індексу. Індекс в ndarrays коливається від 0 (так само, як масиви в C або списки в Python).

Ми можемо використовувати ndarrays для різних обчислень. Всі інші бібліотеки в SciPy та інших бібліотеках також розпізнають та використовують NumPy ndarrays та пов'язані з ними підпрограми для представлення своїх власних структур даних та операцій над ними.

Для створення нового ноутбука Jupyter слід встановити бібліотеку NumPy:

```
!pip3 install numpy
```

Імпортуємо його в поточну записну книжку, виконавши таку команду:

```
import numpy as np
```

Можна створити список, а потім використовувати його для створення простого ndarray наступним чином:

```
l1 = [1, 2, 3]
x = np.array(l1, dtype=np.int16)
```

Тут ми створюємо ndarray зі списку. Тип даних з елементами є 16-бітне ціле число. Ми можемо записати попередній код в один рядок таким чином:

```
x = np.array(l1, dtype=np.int16)
```

Тепер надрукуємо значення ndarray і його тип (який, як ми знаємо, є ndarray).

```
print(x)
print(type(x))
```

В результаті будемо мати:

```
[1 2 3]
<class 'numpy.ndarray'>
```

Як ми бачимо у виведених даних, він належить до класу numpy.ndarray. Таким чином, ми отримуємо, що індексування починається з 0. Наочно продемонструємо це, звернувшись до членів ndarray наступним чином:

```
print(x[0]); print(x[1]); print(x[2])
```

Звідки

```
1
2
3
```

Ми можемо задати більше одного виміру для масиву наступним чином:

```
x1 = np.array([[1, 2, 3], [4, 5, 6]], np.int16)
```

Задамо двовимірну матрицю з двома рядками і трьома стовпцями. Ми можемо отримати доступ до окремих елементів наступним чином:

```
print(x1[0, 0]); print(x1[0, 1]); print(x1[0, 2]);
```

Ми навіть можемо отримати доступ до цілих рядків:

```
print(x1[0, :])
print(x1[1, :])
```

що дає:

```
[1 2 3]
[4 5 6]
```

Ми також можемо мати ndarray з більш ніж двома вимірами. Для прикладу синтаксис для створення тривимірного (3D) масиву:

```
x2 = np.array([[[1, 2, 3], [4, 5, 6]],[[0, -1, -2], [-3, -4,
-5]]], np.int16)
```

Наукові та бізнес-додатки часто використовують багатовимірні дані. Ndarrays дуже корисні для зберігання числових даних. Скомпілюємо тепер наступні елементи та отримати елементи попередньої 3D-матриці.

```
print(x2 [0, 0, 0])
print(x2 [1, 1, 2])
print(x2[:, 1, 1])
```

Ми можемо дізнатися більше про ndarrays, посилаючись на їхні властивості. По-перше, давайте розглянемо всі властивості з демонстрацією. У наступному коді використовується та сама тривимірна матриця, яку ми використовували раніше.

```
x2 = np.array([[[1, 2, 3], [4, 5, 6]],[[0, -1, -2], [-3, -4,
-5]]], np.int16)
```

Ми можемо встановити вимірність масиву за допомогою наступного твердження:

```
print(x2.ndim)
```

Результат повертає кількість вимірів:

```
3
```

Константи NumPy. Бібліотека NumPy містить багато корисних математичних і наукових констант, які можна використовувати у своїх програмах. Наступний фрагмент коду повертає всі важливі константи:

```
print(np.inf)
print(np.NAN)
print(np.NINF)
print(np.NZERO)
print(np.PZERO)
print(np.e)
print(np.euler_gamma)
print(np.pi)
```

Звідки:

```
inf
nan
-inf
-0.0
0.0
2.718281828459045
0.5772156649015329
3.141592653589793
```

2.5 Візуалізація даних за допомогою NumPy і Matplotlib

Розглянемо кілька основних процедур створення ndarray та основи візуалізації даних за допомогою Matplotlib, які нам дозволяють реалізувати алгоритми взаємодії з даними по захворюванню для обробки.

Подальші наші дії базуються на застосуванні NumPy, розглянувши кілька процедур створення ndarray. Ми також розпочнемо роботу з бібліотекою візуалізації даних екосистеми наукових обчислень Matplotlib. Використаємо процедури створення ndarray NumPy для демонстрації візуалізації за допомогою

Matplotlib. Більш докладно викладемо положення програмування системи та її візуалізацію. В загальному нас цікавлять такі технології:

- Matplotlib
- Візуалізація за допомогою NumPy та Matplotlib
- Однолінійні графіки
- Багатолінійні графіки
- Сітки, осі та мітки
- Кольори, стилі та маркери

Matplotlib є невід'ємною частиною SciPy і використовується для візуалізації. Це розширення NumPy. Він забезпечує схожий на MATLAB інтерфейс для побудови та візуалізації. Його спочатку розробив Джон Д. Хантер як альтернативу з відкритим кодом, яку можна використовувати з Python. Ми можемо встановити його за допомогою Jupyter Notebook наступним чином:

```
!pip3 install matplotlib
```

Щоб використовувати його в блокноті для базової побудови графіків, ми повинні імпортувати його модуль pyplot наступним чином:

```
import matplotlib.pyplot as plt
```

Крім того, щоб відобразити візуалізацію Matplotlib в блокноті, ми повинні виконати наступну команду:

```
%matplotlib inline
```

Це змушує Matplotlib відображати вихідні дані у рядку, безпосередньо під коміркою коду, що створює візуалізацію. Ми завжди будемо використовувати це, коли ми повинні використовувати Matplotlib. Давайте також імпортуємо NumPy наступним чином:

```
import numpy as np
```

Створимо ndarray NumPy за допомогою процедур створення ndarray, а потім використовувати Matplotlib для їхньої візуалізації. Почнемо з підпрограм для створення ndarrays. Перша підпрограма – це arange(). Він створює рівномірно розподілені значення із заданим інтервалом. Аргумент стоп-значення є

обов'язковим. Параметри початкового значення та інтервалу мають аргументи за умовчанням, що дорівнює 0 і 1 відповідно.

Давайте подивимося на приклад.

```
x = np.arange(5)
```

У цьому прикладі стоп-значення дорівнює 5, тому створюється ndarray, що починається з 0 і 4, що закінчується. Функція повертає послідовність з напіввідкритим інтервалом, що означає, що стоп-значення не включається у вихідні дані. Оскільки ми не вказали інтервал, передбачається, що він дорівнює 1. Ми можемо побачити висновок та його тип даних так:

```
print(x)
type(x)
```

Результат такий:

```
[0 1 2 3 4]
numpy.ndarray
```

Побудуємо ці числа. Для 2D-графіки нам потрібні пари X-Y. Розглянемо простий приклад і скажемо, що y=f(x)=x, виконавши наступний оператор:

```
y=x
```

Тепер скористуємося функцією plot(), щоб візуалізувати це. Для цього потрібні вхідні значення X, Y та параметри побудови графіка. Далі ми послідовно розглянемо більше про опцій для побудови графіків.

```
plt.plot(x, y, 'o--')
plt.show()
```

Функція show() відображає графік. Як ми бачимо вище, графік візуалізується з варіантами промальовування штриховою лінією o--. Це означає, що точки представлені суцільними кружками, а лінія пунктиром, як показано на Рис. 1.

```
:  import matplotlib.pyplot as plt
   import numpy as np
   x = np.arange(5)
   y=x
   plt.plot(x, y, 'o--')
   plt.show()
```

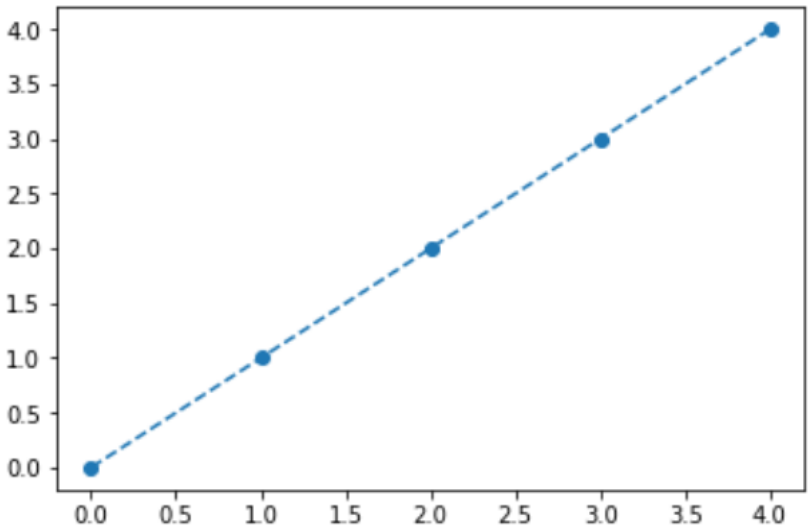

Рис. 2.5 – Візуалізація графіка функції y=f(x)=x

Приклад виклику функції arange() з аргументами start та stop наступним чином:

```
np.arange(2, 6)
```

Він повертає наступний результат (він безпосередньо друкує, і ми не зберігаємо його у змінній):

```
array([2, 3, 4, 5])
```

Функція linspace(start, stop, number) повертає рівномірно розподілені числа за вказаний інтервал. Ми повинні передати йому початкове значення, кінцеве значення та кількість значень таким чином:

```
N = 11
x = np.linspace(0, 10, N)
print(x)
```

Цей код створює 11 чисел (0-10 включно) таким чином:

```
[ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10.]
```

Ми можемо візуалізувати це так:

```
import matplotlib.pyplot as plt
import numpy as np
N = 11
x = np.linspace(0, 10, N)
y = x
plt.plot(x, y, 'o--')
plt.axis('off')
plt.show()
```

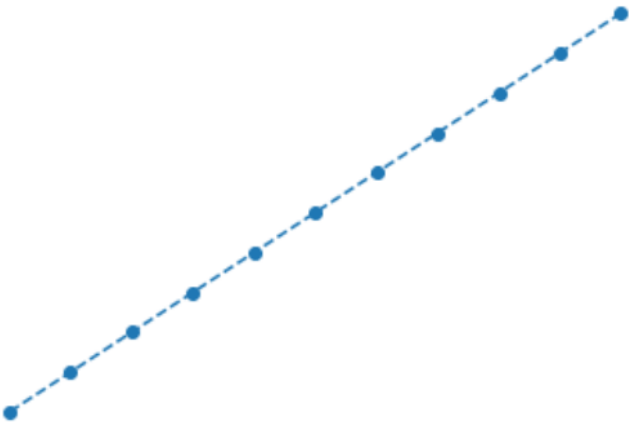

Рис. 2.6 – Виведення y = x за допомогою linspace()

Відключаємо вісь за допомогою рядка plt.axis('off'). Так само ми можемо обчислювати і візуалізувати значення в logspace таким чином:

```
import matplotlib.pyplot as plt
import numpy as np
y = np.logspace(0.1, 1, N)
print(y)
plt.plot(x, y, 'o--')
plt.show()
```

```
[ 1.25892541  1.54881662  1.90546072  2.34422882  2.8840315   3.54813389
  4.36515832  5.37031796  6.60693448  8.12830516 10.        ]
```

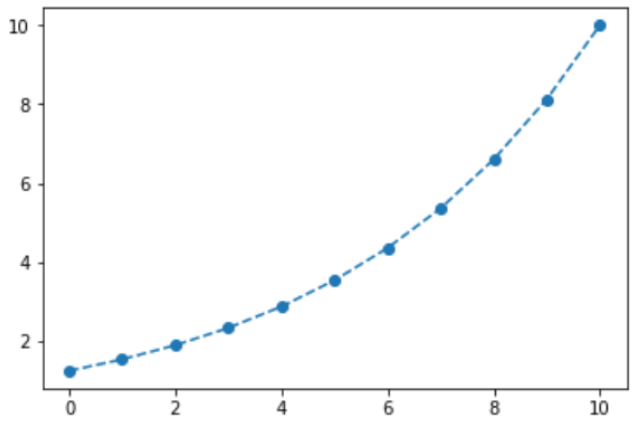

Рис. 2.7 – Виведення logspace()

Для прикладу обчислити ряд у геометричній прогресії так:

```
import matplotlib.pyplot as plt
import numpy as np
y = np.geomspace(0.1, 1000, N)
print(y)
plt.plot(x, y, 'o--')
plt.show()
```

```
[1.00000000e-01 2.51188643e-01 6.30957344e-01 1.58489319e+00
 3.98107171e+00 1.00000000e+01 2.51188643e+01 6.30957344e+01
 1.58489319e+02 3.98107171e+02 1.00000000e+03]
```

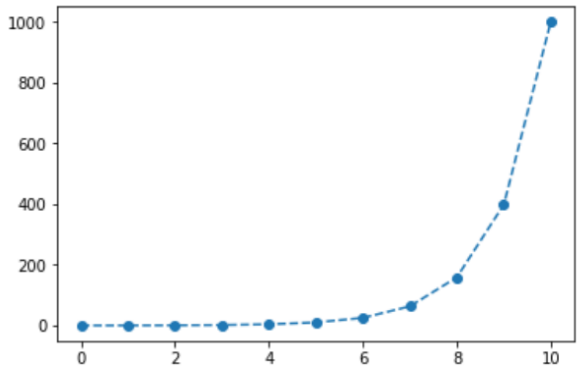

Рис. 2.8 – Виведення geomspace()

Однолінійні графіки. Розглянемо декілька актуальних способів побудови однолінійного графіка. Ми використовували функцію plot() для побудови графіків. Коли малюнку є лише одне візуалізація, використовує функцію plot(), це називається однолінійним графіком. Давайте розглянемо це докладніше кількох прикладах. Ми також можемо використовувати списки Python для візуалізації графіків таким чином:

```
import matplotlib.pyplot as plt
import numpy as np
x = [1, 4, 5, 2, 3, 6]
plt.plot(x)
plt.show()
```

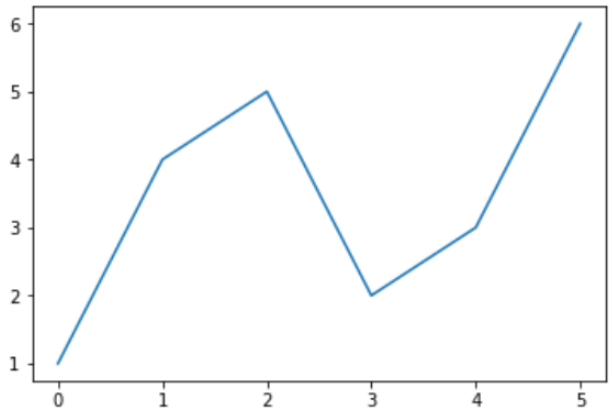

Рис. 2.9 - Приклад лінійного графіка

І тут приймаються значення осі y. Для такого прикладу маємо:

```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(10)
plt.plot(x)
plt.show()
```

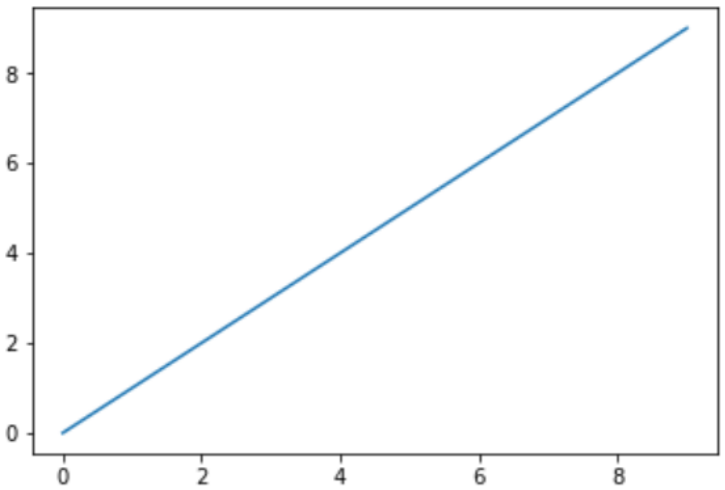

Рис. 2.10 - Простий однолінійний графік із arange()

Далі візуалізуємо квадратичний графік $y=f(x)=x^2$. Код виглядає так:

```
import matplotlib.pyplot as plt
import numpy as np
plt.plot(x, [y**2 for y in x])
plt.show()
```

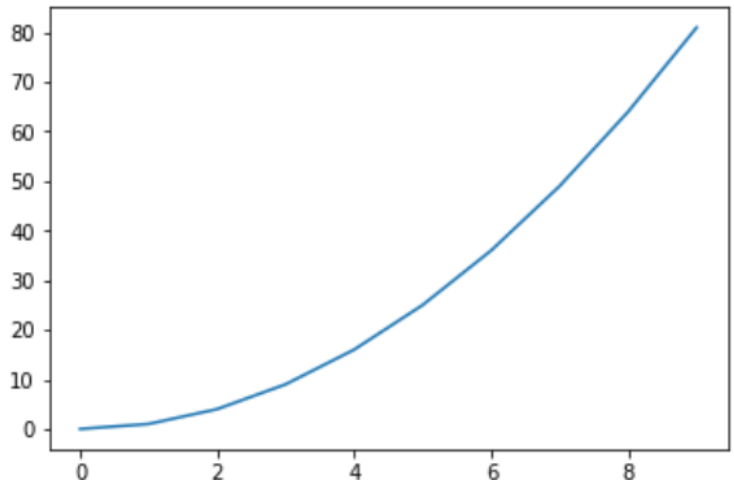

Рис. 2.11 – Побудова функції $y=f(x)=x^2$

Багаторядкові графіки. В одній візуалізації можна відобразити декілька графіків. Давайте подивимося, як ми можемо показати кілька кривих в одній візуалізації на цьому прикладі:

```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(10)
plt.plot(x, x**2)
plt.plot(x, x**3)
plt.plot(x, x*2)
plt.plot(x, 2**x)
plt.show()
```

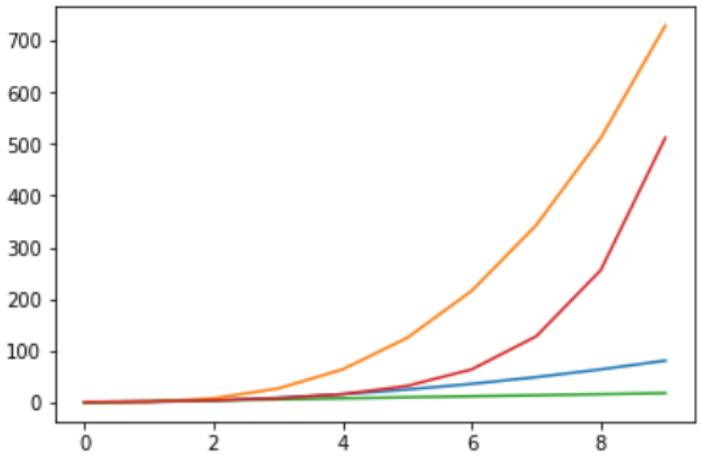

Рис. 8. Побудова мультилінійного графіку

Як ми бачимо, Matplotlib автоматично призначає кольори кривим окремо. Ми можемо написати той самий код простим способом:

```
plt.plot(x, x**2, x, x**3, x, x*2, x, 2**x)
plt.show()
```

Ось приклад використання:

```
import matplotlib.pyplot as plt
import numpy as np
x = np.array([[1, 2, 6, 3], [4, 5, 3, 2]])
plt.plot(x)
plt.show()
```

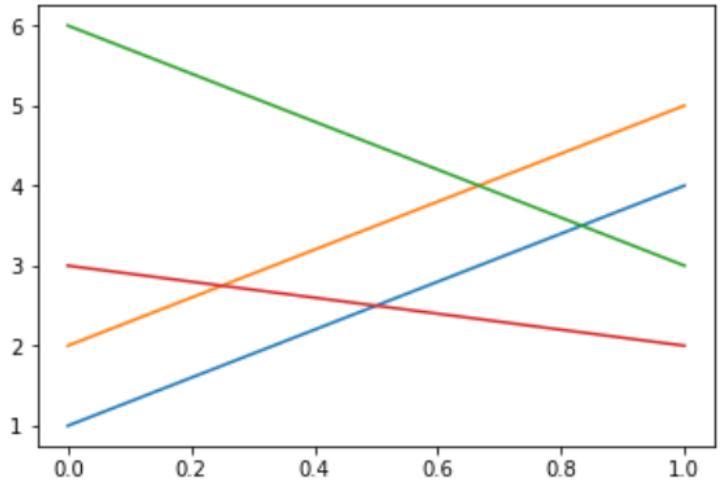

Рис. 2.12 - Приклад мультилінійного графіка

Сітка, осі та мітки. Часто буває дуже корисним увімкнути сітку у візуалізаціях. Це можна зробити за допомогою оператора plt.grid(True). Крім того необхідно оптимально керувати межами осей. Однак перед цим ми швидко навчимося зберігати візуалізацію як зображення. Розглянемо наступний код:

```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(3)
plt.plot(x, x**2, x, x**3, x, 2*x, x, 2**x)
plt.grid(True)
plt.savefig('test.png')
plt.show()
```

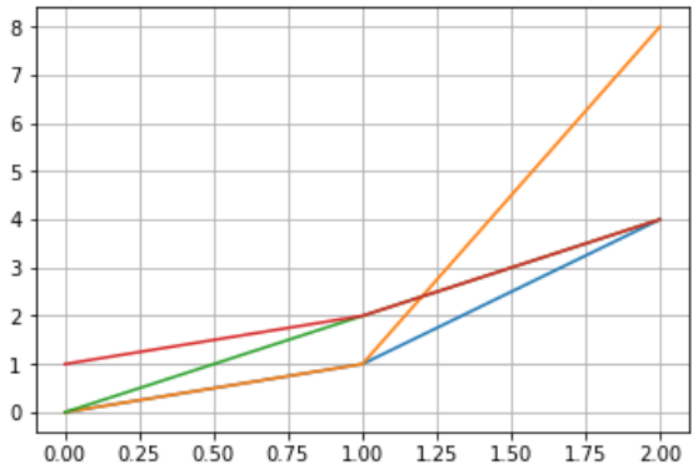

Рис. 2.13 - Мультилінійний графік з сіткою

Оператор plt.savefig('test.png') зберігає зображення в поточний каталог файлу Jupyter Notebook. Ми бачимо, що межі осей встановлені за умовчанням. Ми можемо встановити для них певні значення, як показано тут:

```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(3)
plt.plot(x, x**2, x, x**3, x, 2*x, x, 2**x)
plt.grid(True)
plt.axis([0, 2, 0, 8])
print(plt.axis())
plt.show()
```

```
(0.0, 2.0, 0.0, 8.0)
```

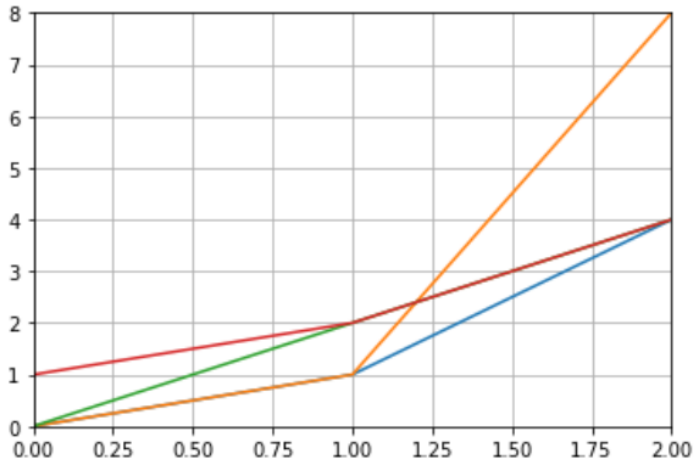

Рис. 2.14 - Вибір налаштування осей

Оператор plt.axis([0, 2, 0, 8]) встановлює значення осей. Перша пара (0, 2) відноситься до обмежень осі x, а друга пара (0, 8) відноситься до обмежень осі y. Ми можемо написати цей код з іншим синтаксисом, використовуючи функції xlim() та ylim() таким чином. Цей код видає такий самий результат, як показано далі. Заголовок та мітки для осей створюються наступним чином:

```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(3)
plt.plot(x, x**2, x, x**3, x, 2*x, x, 2**x)
plt.grid(True)
plt.xlabel('x = np.arange(3)')
plt.xlim([0, 2])
plt.ylabel('y = f(x)')
plt.ylim([0, 8])
plt.title('Simple Plot Demo')
plt.show()
```

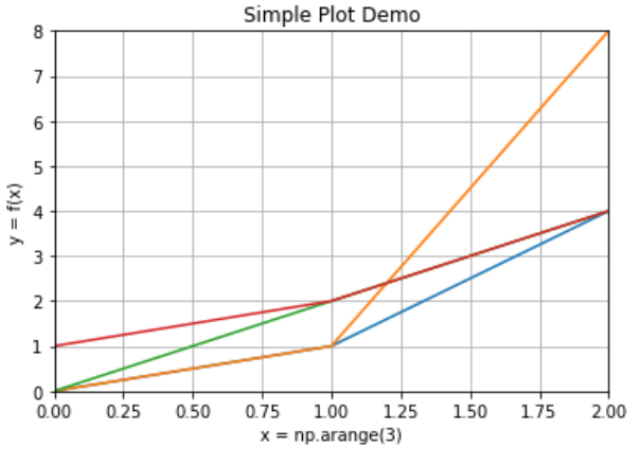

Рис. 2.15 - Підпис та мітки координатних осей

Можна передати аргумент для мітки параметра plot() функцію, а потім викличте функцію legend(), щоб створити легенду наступним чином:

```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(3)
plt.plot(x, x**2, label='x**2')
plt.plot(x, x**3, label='x**3')
plt.plot(x, 2*x, label='2*x')
plt.plot(x, 2**x, label='2**x')
plt.legend()
plt.grid(True)
plt.xlabel('x = np.arange(3)')
plt.xlim([0, 2])
plt.ylabel('y = f(x)')
plt.ylim([0, 8])
plt.title('Simple Plot Demo')
plt.show()
```

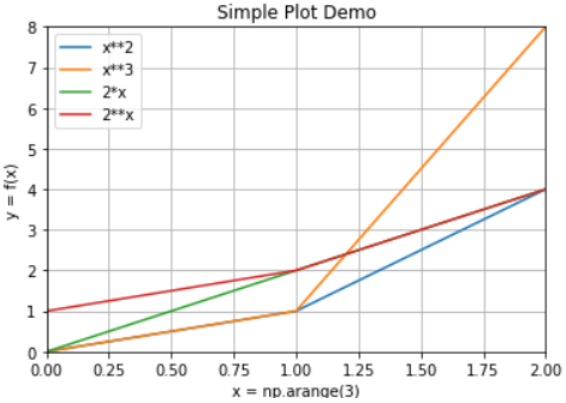

Рис. 2.16 - Вивід графіки разом з підписами

Замість того, щоб передавати рядок легенди як аргумент функції plot(), ми можемо передати список рядків як аргумент функції legend() наступним чином:

```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(3)
plt.plot(x, x**2, x, x**3, x, 2*x, x, 2**x)
plt.legend(['x**2', 'x**3', '2*x', '2**x'], loc='upper center')
plt.grid(True)
plt.xlabel('x = np.arange(3)')
plt.xlim([0, 2])
plt.ylabel('y = f(x)')
plt.ylim([0, 8])
plt.title('Simple Plot Demo')
plt.show()
```

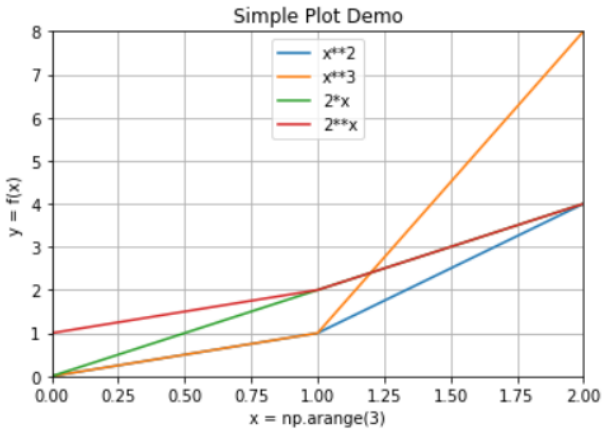

Рис. 2.17 - Вивід графіки разом з підписами посередині

```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(5)
y = x
plt.plot(x, y+1, 'g')
plt.plot(x, y+0.5, 'y')
plt.plot(x, y, 'r')
plt.plot(x, y-0.2, 'c')
plt.plot(x, y-0.4, 'k')
plt.plot(x, y-0.6, 'm')
plt.plot(x, y-0.8, 'w')
plt.plot(x, y-1, 'b')
plt.show()
```

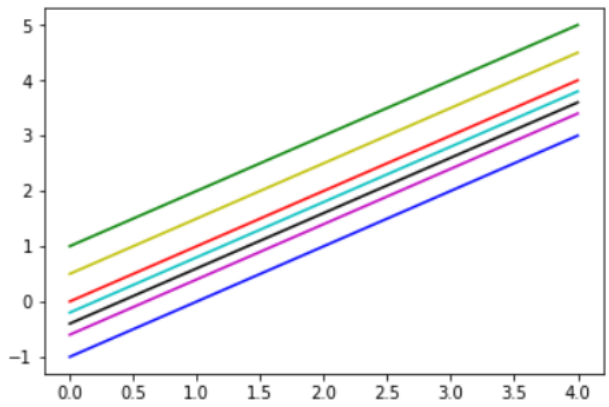

Рис. 2.18 - Приклад задання кольорів графіки

Кольори, стилі та маркери. У разі багаторядкових графіків Matplotlib автоматично призначає кольори, стилі та маркери. Спостерігали кілька прикладів їхнього налаштування. Далі ми розглянемо детально, як їх налаштувати. Почнемо із кольорів ліній. У наступному коді перераховані всі основні кольори Matplotlib (у цьому прикладі ми не налаштовуємо стилі та маркери).

Налаштовуємо стиль лінії наступним чином:

```
: import matplotlib.pyplot as plt
import numpy as np
x = np.arange(5)
y = x
plt.plot(x, y, '-', x, y+1, '--', x, y+2, '-.', x, y+3, ':')
plt.show()
```

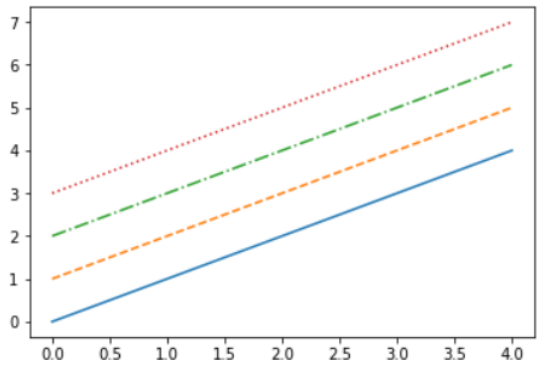

Рис. 2.19 - Різні стилі задання ліній

Усі три функції (кольори, маркери та стилі ліній) можна комбінувати, щоб налаштувати візуалізацію наступним чином:

```
import matplotlib.pyplot as plt
import numpy as np
plt.plot(x, y, 'mo--')
plt.plot(x, y+1 , 'g*-.')
plt.show()
```

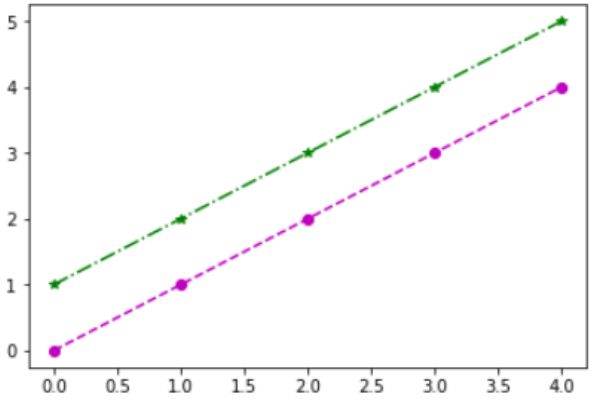

Рис. 2.20 - Поєднання різних типів подання ліній

Змінити маркери можливо таким чином:

```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(5)
y = x
plt.plot(x, y, '.')
plt.plot(x, y+0.5, ',')
plt.plot(x, y+1, 'o')
plt.plot(x, y+2, '<')
plt.plot(x, y+3, '>')
plt.plot(x, y+4, 'v')
plt.plot(x, y+5, '^')
plt.plot(x, y+6, '1')
plt.plot(x, y+7, '2')
plt.plot(x, y+8, '3')
plt.plot(x, y+9, '4')
plt.plot(x, y+10, 's')
plt.plot(x, y+11, 'p')
plt.plot(x, y+12, '*')
plt.plot(x, y+13, 'h')
plt.plot(x, y+14, 'H')
plt.plot(x, y+15, '+')
plt.plot(x, y+16, 'D')
plt.plot(x, y+17, 'd')
plt.plot(x, y+18, '|')
plt.plot(x, y+19, '_')
plt.show()
```

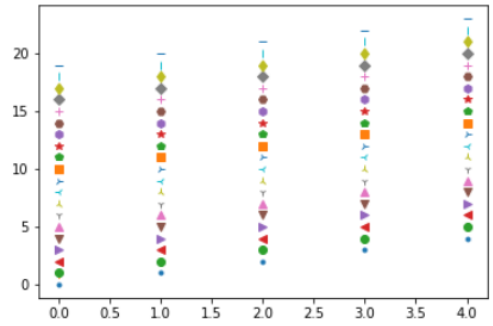

Рис. 2.21 - Маркери ліній

Це базові налаштування, але ви можете налаштувати більш детально. Наприклад, використовуйте цей код для налаштування інших деталей:

```
import matplotlib.pyplot as plt
import numpy as np
plt.plot(x, y, color='g', linestyle='--', linewidth=1.5,
marker='^', markerfacecolor='b', markeredgecolor='k',
markeredgewidth=1.5, markersize=5)
plt.grid(True)
plt.show()
```

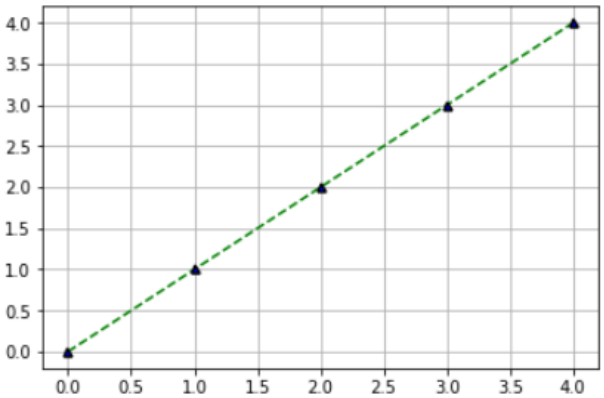

Рис. 2.22 - Застосування окремих деталей налаштування

Бажано крім того налаштувати значення по осях x та y наступним чином:

```
: import matplotlib.pyplot as plt
import numpy as np
x = y = np.arange(10)
plt.plot(x, y, 'o--')
plt.xticks(range(len(x)), ['a', 'b', 'c', 'd', 'e', 'f', 'g',
'h', 'i', 'j'])
plt.yticks(range(0, 10, 1))
plt.show()
```

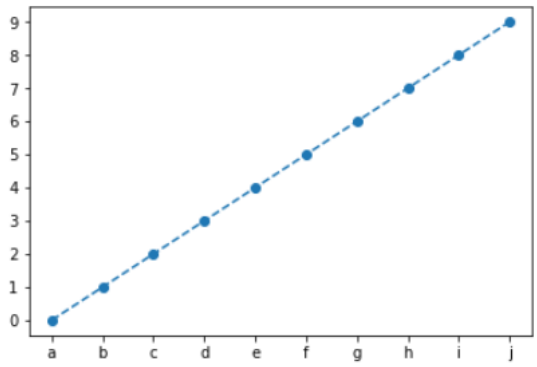

Рис. 2.23 - Налаштування позначок на осях

Таким чином нами основна увага приділялася різним параметрам, доступним для візуалізацій даних. Ми багато чого дізналися про візуалізацію даних. Основні положення, розглянуті нами вище, будуть використовуватися протягом для візуалізації даних у дипломній роботі.

Далі ми ще більше вивчимо візуалізацію даних. Ми розглянемо сукупність підходів для того, щоб візуалізувати зображення та тривимірні фігури та розглянемо основи операцій із зображеннями.

Візуалізація зображень та 3D-фігур. Розпочнемо роботу над візуалізацією за допомогою бібліотеки Matplotlib в Python 3. У цьому розділі ми продовжимо наші пригоди з Matplotlib та NumPy для візуалізації зображень та тривимірних фігур. Розглянемо сукупність засобів для візуалізації даних у таких основних положеннях:

- Візуалізація зображень
- Операції із зображеннями
- 3D-візуалізація

Далі будемо безпосередньо працювати із зображеннями та 3D-фігурами.

Візуалізація зображень. Проаналізуємо засоби для візуалізації зображення. Функція imread() Matplotlib може читати зображення у форматі .png. Щоб він міг читати файли інших форматів, необхідно встановити іншу бібліотеку обробки зображень, відому як pillow. Встановити цю бібліотеку можна, виконавши наступну команду в комірці записника:

```
!pip3 install pillow
```

3D-візуалізації за допомогою Python 2. Ми відображали візуалізації в записнику. Цей процес добре працює для 2D-візуалізації. Однак у разі 3D-візуалізації вона показуватиме їх лише під фіксованим кутом. Тому ми маємо використовувати інший метод. Найкращий спосіб працювати з цим – використовувати іншу команду, яка показує візуалізацію в окремому вікні Qt наступним чином:

```
%matplotlib qt
```

Імпортуємо модуль pyplot у Matplotlib та NumPy у ранніх осередках В подальшому ми маємо імпортувати більше функціональності з такими твердженнями:

```
from mpl_toolkits.mplot3d import Axes3D
from mpl_toolkits.mplot3d import axes3d
```

Почнемо з простого прикладу. Перенесемо принципи побудови 2D графіків, тепер зробимо ті самі речі в 3D. Побудуємо просту параметричну криву. Спочатку визначимо фігуру та вісь:

```
fig = plt.figure()
ax = fig.gca(projection='3d')
```

Для 3D-візуалізацій ми включили 3D-проекції за допомогою пари параметрів та аргументів project='3d'. Далі визначимо дані. Для цього нам потрібні координати x, y та z точок даних. Спочатку визначте полярні координати точок даних таким чином:

```
theta = np.linspace(-3 * np.pi, 3 * np.pi, 200)
z = np.linspace(-3, 3, 200)
r = z**3 + 1
```

Тепер обчислимо координати x та y наступним чином:

```
x = r * np.sin(theta)
y = r * np.cos(theta)
```

Ми використовуємо тригонометричні функції у бібліотеці NumPy для обчислення x та y. Отже, ми сконструюємо це так:

```
ax.plot(x, y, z, label='Parametric Curve')
ax.legend()
plt.show()
```

Результат відображатиметься в окремому вікні, він буде інтерактивним таким чином, що ми зможемо змінити кут огляду.

```
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
from mpl_toolkits.mplot3d import axes3d
fig = plt.figure()
ax = fig.gca(projection='3d')
theta = np.linspace(-3 * np.pi, 3 * np.pi, 200)
z = np.linspace(-3, 3, 200)
r = z**3 + 1
x = r * np.sin(theta)
y = r * np.cos(theta)
ax.plot(x, y, z, label='Parametric Curve')
ax.legend()
plt.show()
```

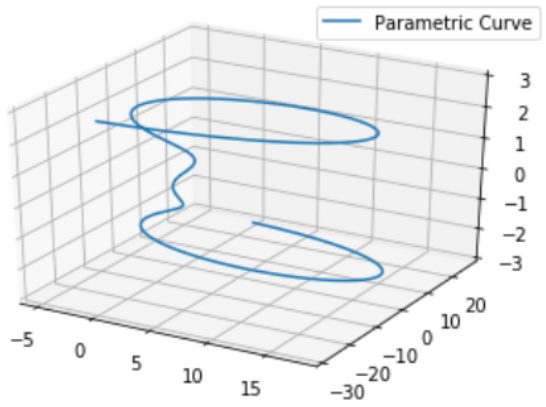

Рис. 2.24 - Просторовий графік параметрично заданої фігури

Якщо окреме вікно виводу не з'являється, а 3D візуалізація відображається в самому браузері, необхідно перезапустити ядро Jupyter і повторно виконати відповідні комірки. У цьому випадку не виконуйте комірку, що містить вбудований % matplotlib.

Далі створимо тривимірну гістограму. Тепер обчислимо дані так:

```
x = np.arange(4)
y = np.arange(4)
xx, yy = np.meshgrid(x, y)
print(xx);print(yy)
```

Функція meshgrid() створює і повертає матриці координат векторів координат наступним чином:

```
[[0 1 2 3]
 [0 1 2 3]
```

Використаємо функцію ravel(), щоб згладити обидві матриці таким чином:

```
X, Y = xx.ravel(), yy.ravel()
print(X); print(Y);
```

Це призводить до наступного результату:

```
[0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3]
[0 0 0 0 1 1 1 1 2 2 2 2 3 3 3 3]
```

Тепер обчислимо більше значень:

```
top = X + Y
bottom = np.zeros_like(top)
width = depth = 1
```

Далі створимо фігуру та осі й створимо гістограму наступним чином:

```
fig = plt.figure(figsize=(8, 3))
ax1 = fig.add_subplot(121, projection='3d')
ax2 = fig.add_subplot(122, projection='3d')
ax1.bar3d(X, Y, bottom, width, depth, top, shade=True)
ax1.set_title('Shaded')
ax2.bar3d(X, Y, bottom, width, depth, top, shade=False)
ax2.set_title('Not Shaded')
plt.show()
```

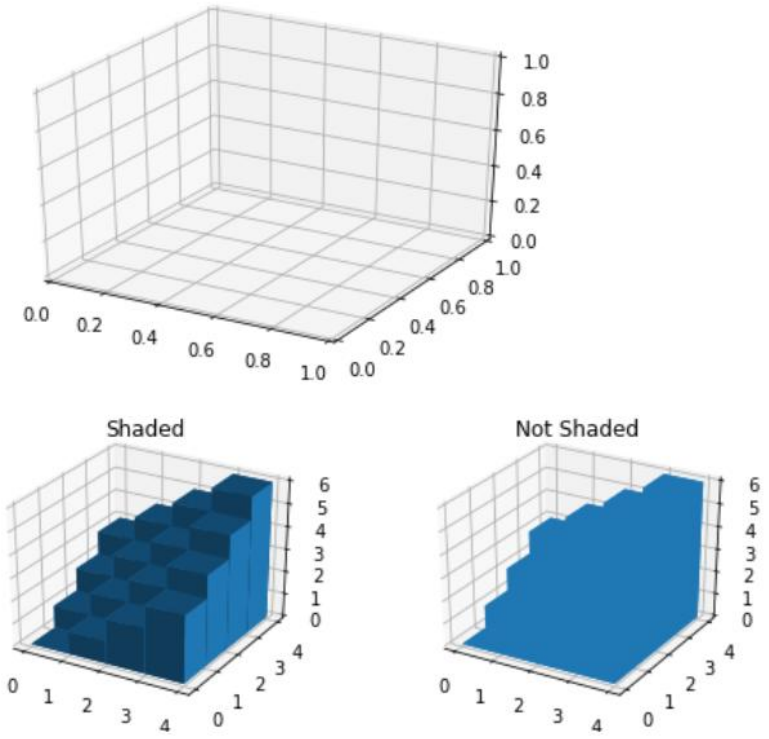

Рис. 2.25 - Візуалізація тривимірної гістограми

Далі візуалізуємо каркас наступним чином:

```
fig = plt.figure()
ax = fig.gca(projection='3d')
X, Y, Z = axes3d.get_test_data(delta=0.1)
ax.plot_wireframe(X, Y, Z)
plt.show()
```

У цьому фрагменті коду ми спочатку створюємо фігуру та вісь у 3D-режимі. Потім вбудована функція get_test_data() повертає тестові дані. Потім ми використовуємо функцію plot_wireframe() для створення каркасної моделі. Результат показано на Рис. 2.26.

```
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
from mpl_toolkits.mplot3d import axes3d
fig = plt.figure()
ax = fig.gca(projection='3d')
X, Y, Z = axes3d.get_test_data(delta=0.1)
ax.plot_wireframe(X, Y, Z)
plt.show()
```

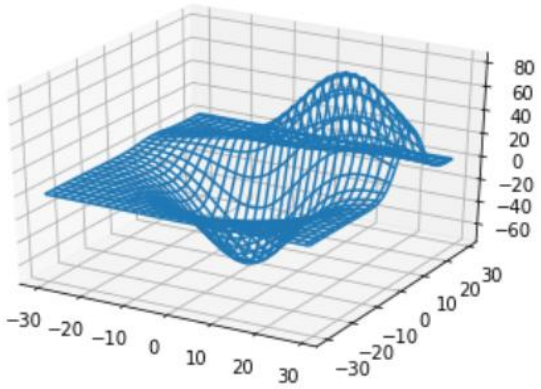

Рис. 2.26 - Візуалізація 3D-каркасу

Обчислимо та візуалізуємо поверхню. Спочатку обчислимо координати x та y, а потім обчислимо крок сітки:

```
x = np.arange(-3, 3, 0.09)
y = np.arange(-3, 3, 0.09)
X, Y = np.meshgrid(x, y)
print(X); print(Y)
```

звідки

```
[[-3.   -2.91 -2.82 ...  2.76  2.85  2.94]
 [-3.   -2.91 -2.82 ...  2.76  2.85  2.94]
 [-3.   -2.91 -2.82 ...  2.76  2.85  2.94]
 ...
 [-3.   -2.91 -2.82 ...  2.76  2.85  2.94]
```

Потім обчислимо координату z наступним чином:

```
R = np.sqrt(X**2 + Y**2)
Z = np.cos(R)
print(Z)
```

В результаті, створимо фігуру та вісь і візуалізємо 3D-поверхню:

```
fig = plt.figure()
ax = fig.gca(projection='3d')
surf = ax.plot_surface(X, Y, Z,
                       cmap=plt.cm.cool,
                       linewidth=0,
                       antialiased=False)
plt.show()
```

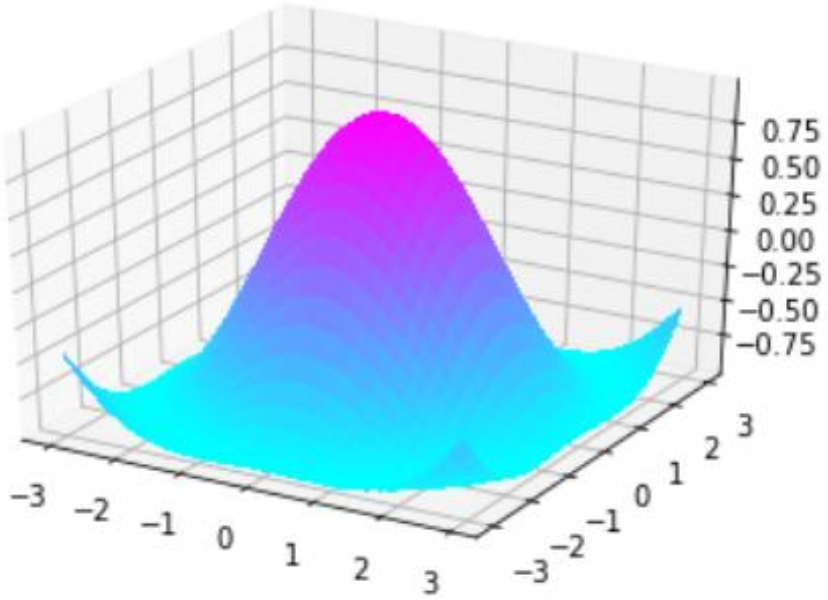

Рис. 2.27 - Побудова просторової фігури

Таким чином, ми розглянули основні методи та продемонстрували візуалізацію зображень. Встановили особливості та продемонстрували основні операції із зображеннями, такі як арифметичні операції та поділ кольорових зображень на складові канали. Ми також розглянули написання програм для 3D-візуалізації, включаючи криві, стовпці та сітки.

2.5 Реалізація основних сутностей класів та методів

Після аналізу основних варіантів використання та побудови UML діаграм було розроблено базову версію програмної системи. Дана версія включає всі інфраструктурні класи, необхідні для подальшого розвитку продукту. Також дана версія програмної системи включає класи доступу до сутностей в базі даних. Для

реалізації програмної системи було прийнято рішення покроково розробляти мікросервіси, що входять до її складу. Так, набір основних мікросервісів включає:

- Шлюз;
- Автентифікатор;
- Процесор медичних даних;
- Генератор наборів даних;
- Процесор даних моделей машинного навчання;
- Ініціатор зворотного зв’язку.

Наведені вище мікросервіси можуть в подальшому поділятись на менші складові частини чи навпаки об’єднуватись в залежності від змін вимог до програмної системи та наявних ресурсів.

3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ ОБРОБКИ ДАНИХ COVID-19

3.1 Робота з набором даних про пандемію COVID-19

У цьому розділі ми отримуємо реальні дані відомих баз знань та накопичення інформації та візуалізуємо їх, а також візуалізуємо реальні набори даних.

Отримуємо ці дані за допомогою спеціальних бібліотек у Python. Одну з таких бібліотек можна знайти за адресою https://ahmednafies.github.io/covid/. Ця бібліотека може отримувати дані як з бази Університету Джона Гопкінса, так і з World-O-Meter. Щоб встановит, створюємо новий блокнот для цього розділу та виконайте таку команду в рядку:

```
!pip3 install covid
```

Далі імпортуйте бібліотеку наступним чином:

```
from covid import Covid
```

Отримати дані можна за допомогою цього коду:

```
covid = Covid()
```

Він за замовчуванням отримує дані з Університету Джона Гопкінса. Ви також можете явно вказати джерело даних:

```
covid = Covid(source="john_hopkins")
```

Відображаємо всі дані за допомогою такої команди:

```
covid.get_data()
```

Повертається список словників, як показано на малюнку нижче:

```
In [35]: # get all data
         covid.get_data()

Out[35]: [{'country': 'North America',
           'confirmed': 3628797,
           'new_cases': 7809,
           'deaths': 178674,
           'recovered': 1648713,
           'active': 1801410,
           'critical': 18601,
           'new_deaths': 945,
           'total_tests': 0,
           'total_tests_per_million': Decimal('0'),
           'total_cases_per_million': Decimal('0'),
           'total_deaths_per_million': Decimal('0'),
```

Рис. 3.1 - Дані COVID-19, отримані з відомих джерел

Визначаємо джерело даних таким чином:

```
covid.source
```

Результат у цьому випадку показано тут:

```
'worldometers'
```

Також можна отримати статус за назвою країни наступним чином:

```
covid.get_status_by_country_name("Ukraine")
```

Також існує можливість отримати дані за ідентифікатором країни (дійсна лише для джерела даних Johns Hopkins) за допомогою цього коду:

```
covid.get_status_by_country_id(115)
```

Щоб отримати список країн, які постраждали від пандемії COVID-19, використовуйте цей синтаксис:

```
covid.list_countries()
```

Загальну кількість активних випадків можна отримати так:

```
covid.get_total_active_cases()
```

Загальну кількість підтверджених випадків можна отримати так:

```
covid.get_total_confirmed_cases()
```

Загальну кількість випадків одужання можна отримати так:

```
covid.get_total_recovered()
```

Загальну кількість смертей можна отримати так:

```
covid.get_total_deaths()
```

Запустивши ці оператори, отримуємо результат отримання статистичних даних.

3.2 Читання даних, що зберігаються у форматі CSV

Зчитуємо дані, що зберігаються у форматі значень, розділених комами (CSV), за допомогою методу `read_csv()`. Файл CSV може зберігатися за віддаленою URL-адресою або місцем на диску. Ось приклад читання файлу CSV, розміщеного на URL-адресі через Інтернет:

```
df = pd.read_csv('https://raw.githubusercontent.com/cs109/2014_
data/master/countries.csv')
df.head(5)
```

Результат зчитування даних наступний:

	Country	Region
0	Algeria	AFRICA
1	Angola	AFRICA
2	Benin	AFRICA
3	Botswana	AFRICA
4	Burkina	AFRICA

Рис 3.2 - Візуалізація даних з CSV-файла в табличній формі

Таким же чином ми можемо зчитувати дані, що зберігаються у файлі на диску, у фрейм даних.

3.3 Візуалізація даних про COVID-19

Конвертувати всі ці дані у фрейм даних pandas можна наступним чином:

```
import pandas as pd
df = pd.DataFrame(covid.get_data())
print(df)
```

Результат показаний на рисунку нижче:

```
import pandas as pd
df = pd.DataFrame(covid.get_data())
print(df)
```

```
                  country  confirmed  new_cases  deaths  recovered   active  \
0           North America    3628797       7809  178674    1648713  1801410
1           South America    2614931       1036   96832    1717350   800749
2                    Asia    2700746      20485   64867    1839062   796817
3                  Europe    2513631       8954  194782    1462217   856632
4                  Africa     511949        779   12026     248751   251172
..                    ...        ...        ...     ...        ...      ...
218  Caribbean Netherlands          7          0       0          7        0
219             St. Barth          6          0       0          6        0
220               Anguilla          3          0       0          3        0
221  Saint Pierre Miquelon          1          0       0          1        0
```

Рис. 3.3 - Дані про COVID-19 перетворено у фрейм даних Pandas

Застосуємо код для сортування даних:

```
sorted = df.sort_values(by=['confirmed'], ascending=False)
```

Ці дані містять кумулятивні дані для всіх континентів і світу. Ми можемо виключити дані для світу та континентів таким чином:

```
excluded = sorted[~sorted.country.isin(['Europe', 'South America', 'Asia', 'World', 'North America', 'Africa'])]
```

10 найкращих країн можна отримати таким чином:

```
top10 = excluded.head(10) print(top10)
```

Результат показаний на рисунку:

```
top10 = excluded.head(10)
print(top10)
```

```
   country  confirmed  new_cases  deaths  recovered   active  critical
8      USA    3097538        454  133991    1355524  1608023     15371
9   Brazil    1674655          0   66868    1117922   489865      8318
10   India     746506       3025   20684     458618   267204      8944
11  Russia     700792       6562   10667     472511   217614      2300
12    Peru     309278          0   10952     200938    97388      1265
13   Chile     301019          0    6434     268245    26340      2060
14   Spain     299210          0   28392          0        0       617
15      UK     286349          0   44391          0        0       209
16  Mexico     268008       6258   32014     163646    72348       378
17    Iran     248379       2691   12084     209463    26832      3309
```

Рис. 3.4 - Топ 10 найбільш-постраждалих від COVID-19 країн у світі згідно отриманої статистики

Тепер ми можемо витягнути дані в змінні наступним чином:

```
x = top10.country
y1 = top10.confirmed
y2 = top10.active
y3 = top10.deaths
```

```
y4 = top10.recovered
```

Тепер можемо використовувати результати для візуалізації. Імпортуєм Matplotlib і ввімкніть малювання на площині за допомогою команди:

```
%matplotlib inline
import matplotlib.pyplot as plt
```

Створюємо простий лінійний графік наступним чином:

```
plt.plot(x, yl)
plt.xticks(rotation=90)
plt.show()
```

Результат візуалізації даних продемонстровано:

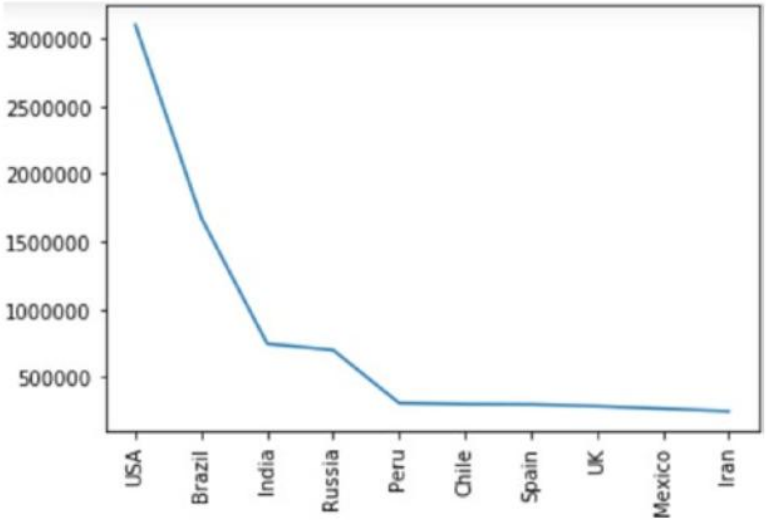

Рис. 3.5 - Топ 10 країн, які найбільше постраждали від COVID-19 (лінійний графік)

Також, ми можемо відобразити гістограму криву таким чином:

```
plt.plot(x, yl)
plt.xticks(rotation=90)
plt.show()
```

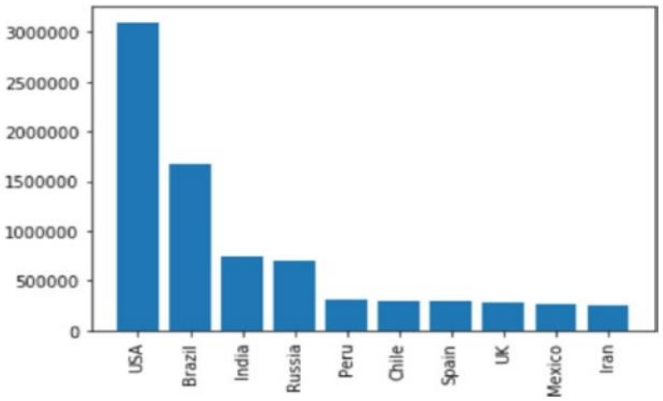

Рис. 3.6 - Візуалізація даних у вигляді гістограми

Застосуємо мультвибір та функції побудови графіків для відображення декількох трендів на кривих графіків:

```
plt.plot(x, yl, label='Confirmed')
plt.plot(x, y2, label='Active')
plt.plot(x, y3, label='Deaths')
plt.plot(x, y4, label='Recovered')
plt.legend(loc='upper right')
plt.xticks(rotation=90) plt.show()
```

Результат моделювання візуалізовано на рисунку:

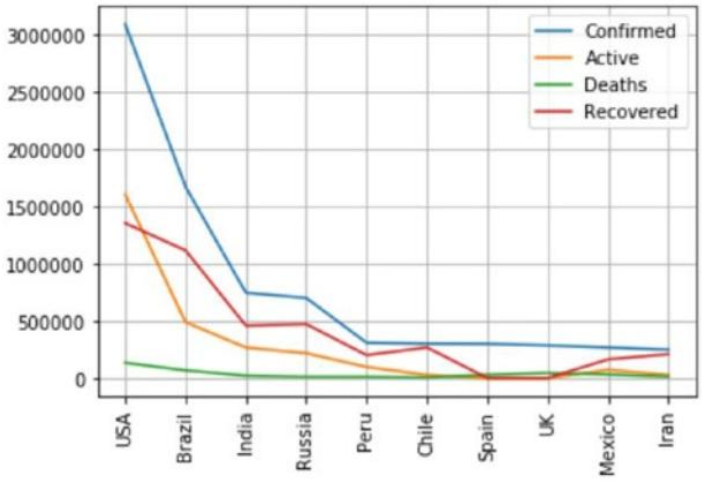

Рис. 3.7 - найбільш постраждалих від COVID-19 країн (багаторядковий графік)

Для зручності представлення відобразимо дані у вигляді вертикальних гістограм таким чином:

```
labels = ['Confirmed', 'Active', 'Deaths', 'Recovered']
plt.plot(x, y1, x, y2, x, y3, x, y4)
plt.legend(labels, loc='upper right')
plt.xticks(rotation=90)
plt.grid()
plt.show()
```

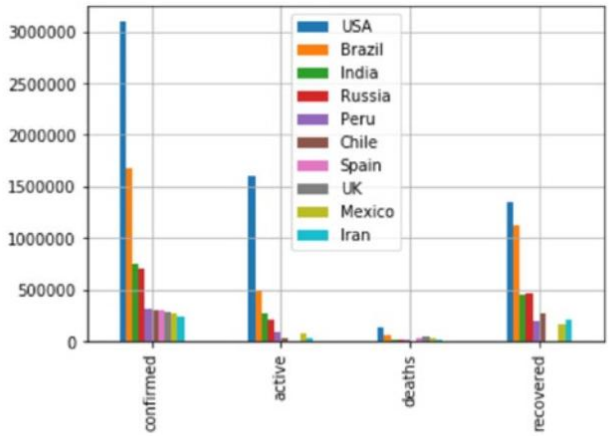

Рис. 3.8 - 10 країн, які найбільше постраждали від COVID-19 (багаторядковий графік із сітками)

Додамо демонстрацію даних для порівнння у вигляді вертикальних гістограм за допомогою такого коду:

```
df2.plot.bar(stacked=True);
plt.legend(x, loc='upper center')
plt.xticks(rotation=90)
plt.grid()
plt.show()
```

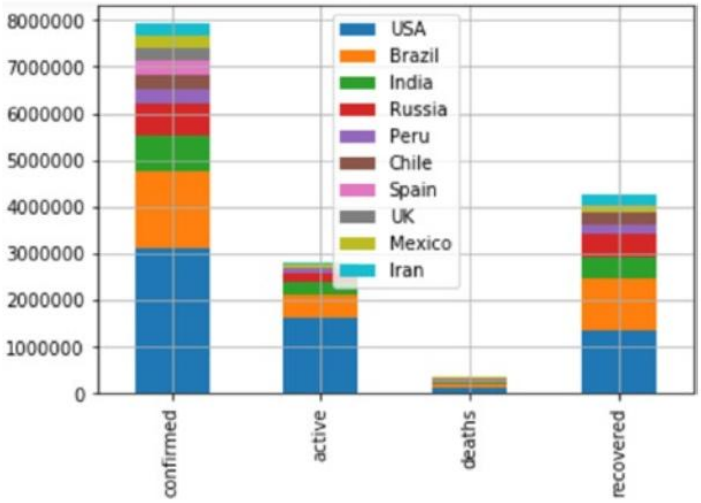

Рис. 3.9 - Статистика COVID-19 у 10 найпопулярніших країнах (гістограма з вертикальним накопиченням із сіткою)

Використаємо площинні графіки для візуалізації цих даних. За замовчуванням площинний графік, як ми знаємо, складений. Ми можемо створити його наступним чином:

```
df2.plot.area();
plt.legend(x, loc='upper right')
plt.xticks(rotation=90)
plt.grid()
plt.show()
```

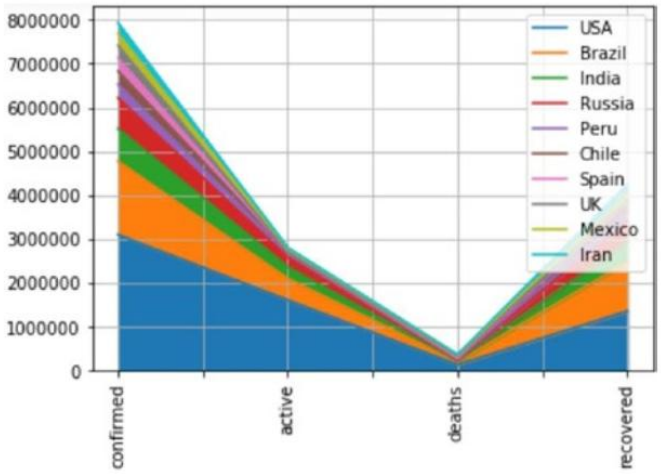

Рис. 3.10 - Статистика COVID-19 (діаграма з накопиченням)

Актуально добавити на діаграму площини, що перекривається, наступним чином:

```
df2.plot.area(stacked=False);
plt.legend(x, loc='upper right')
plt.xticks(rotation=90)
```

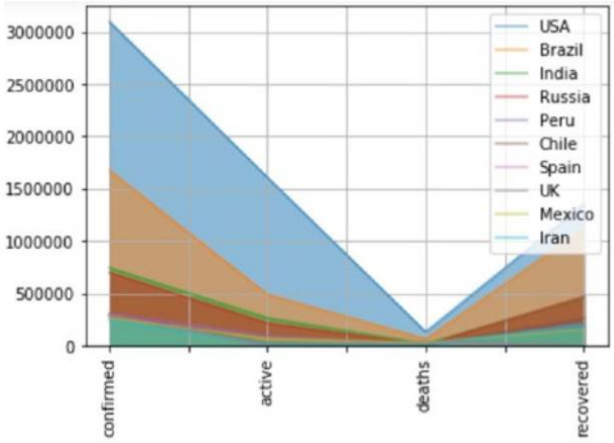

Рис. 3.11 - Статистика по COVID-19 (діаграма площин, що перекривається)

Щоб створити красиву діаграму розсіювання, де розмір точок пропорційний величині даних, використаємо наступний код:

```
factor=0.0001
plt.scatter(x, y1, s=y1*factor);
plt.scatter(x, y2, s=y2*factor);
plt.scatter(x, y3, s=y3*factor);
plt.scatter(x, y4, s=y4*factor);
plt.legend(labels, loc='upper right')
plt.xticks(rotation=90)
plt.grid()
plt.show()
```

Результат виконання даного коду показаний на рисунку:

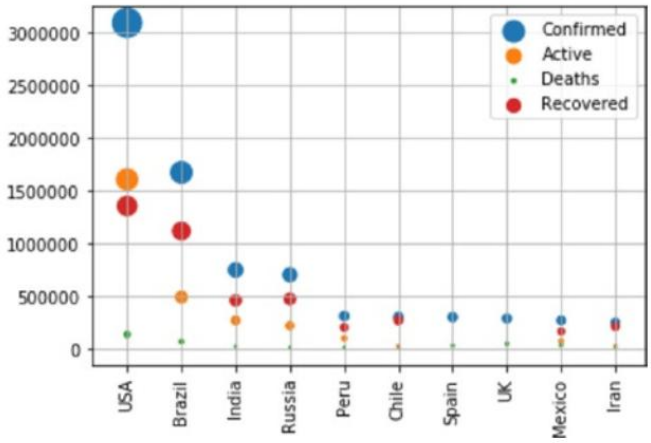

Рис 3.12 - Топ-10 статистичних даних про COVID-19 у країнах (точкова діаграма)

Генерування гарної кругової діаграми для ефектної візуалізації статистики по захворюваності здійснимо за допомогою такого коду:

```
plt.pie(y1, labels=x)
plt.title('Confirmed Cases')
plt.show()
```

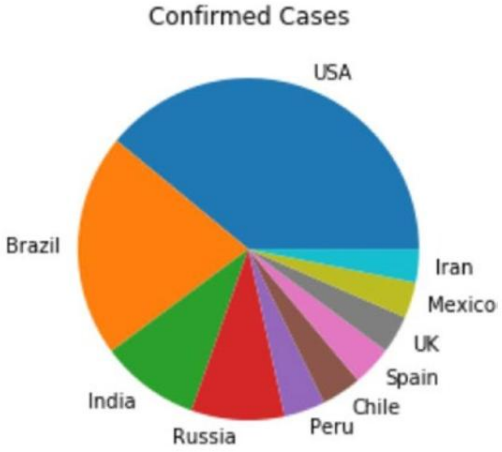

Рис 3.13 - Результат виконання побудови «кругова діаграма»

Топ-10 статистичних даних про COVID-19 у країнах

3.6 Впровадження реалізованого ПЗ

Після виконання розробки та тестування, було вирішено встановити дане ПЗ для виконання дослідів з моделями та оцінки ефективності розроблюваного продукту. Також було вирішено встановити програмну систему для проведення доопрацювання, обновлення та доробки даного реалізованого ПЗ в необхідному обсязі. Тобто у наступних реалізаціях можна буде здійснити зміну формул моделі, додати нові можливості.

Висновки до розділу.

У цьому розділі здійснено кодинг та візуалізація результатів виконання моделей представлення статистичних даних, як отримати дані про COVID-19 із різних онлайн-джерел і перетворити їх у фрейм даних pandas. Продемонстровано процес та складові, як підготувати різні типи візуалізацій для графічного представлення даних про COVID-19.

4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Охорона праці. Вимог охорони праці з ціллю збереження здоров'я при роботі за комп’ютерною технікою. Планування робочого місця

При написанні кваліфікаційної роботи та розробці data-системи для Jupyter Notebook використовувався персональний комп’ютер, тому слід зазначити, що при роботі за комп’ютерною технікою необхідно дотримуватися вимог охорони праці з ціллю збереження здоров'я.

При роботі з комп’ютером значним чином відбувається вплив на нервово-емоційний стан операторів, така робота характеризується великим навантаженням на м’язи рук при роботі з клавіатурою комп'ютера, високою інтенсивністю зорової роботи та значним розумовим перенапруженням.

Отже, раціональне планування робочого місця повинно забезпечити: зменшення втоми працівників та підвищення продуктивності праці, уникнення загального дискомфорту, якнайкраще розташування інструментів та предметів праці.

Для того, щоб виявити та проаналізувати шкідливі і небезпечні виробничі фактори необхідно почати з аналізу дотримання вимог, встановлених санітарними правилами і нормами [ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин»] для виробничих приміщень та робочих місць.

На виробництві для дійсної оцінки умов праці відбувається сертифікація робочих місць відповідно до умов праці та використовується «Гігієнічна класифікація праці» за показниками небезпечності та шкідливості факторів виробничого середовища, напруженості та тяжкості трудового процесу. На основі принципів гігієнічної класифікації умови праці поділяють на чотири класи:

- Перший клас - оптимальні умови праці - це умови, за яких створюються передумови для підтримки високого рівня працездатності;

- Другий клас - допустимі умови праці, що характеризуються такими рівнями факторів виробничого середовища та трудового процесу, які не перевищують встановлених норм;

- Третій клас - шкідливі умови праці, що характеризуються такими рівнями шкідливих виробничих факторів, які перевищують нормативи і можуть мати несприятливий вплив на організм працівника і /або його нащадків;

- Четвертий клас – небезпечні/екстремальні умови праці.

На підставі сертифікації робочого місця необхідно охарактеризувати інтенсивність робіт за такими напрямами:

- відповідність обладнання нормативно-технічним вимогам;
- документації, а також характер і обсяг виконуваних робіт;
- відповідність площі та обсягу займаного робочого місця чинним нормам;
- спеціалізоване устаткування робочого місця (засоби захисту пристроїв та їх технічний стан);
- відповідність технологічного процесу, інструментів, устаткування, засобів контролю вимогам стандартів безпеки і нормам охорони праці.

Обладнання та організація робочого місця ВДТ ЕОМ повинні задовольняти відповідність конструкції всіх елементів робочого місця та їх відносного розташування ергономічним вимогам з урахуванням особливостей та характеру трудової діяльності [ДСТУ 7299:2013 «Дизайн і ергономіка. Робоче місце оператора. Взаємне розташування елементів робочого місця. Загальні вимоги ергономіки»].

Раціональне планування робочого місця повинно забезпечити: найкраще розміщення інструментів та предметів праці, уникати загального дискомфорту, зменшити втому працівників та підвищити продуктивність праці.

Заходи щодо усунення ризику ураження електричним струмом зводяться до правильного розміщення обладнання та електричних кабелів. Також інші заходи для забезпечення електробезпеки збігаються із загальними заходами пожежної та електробезпеки.

В якості заходів профілактики для того, щоб забезпечити пожежну безпеку необхідно у приміщеннях використовувати приховану електромережу, ввімкнення та вимкнення живлення виконувати обладнанням за допомогою стандартних вимикачів, надійні розетки з пожежобезпечних матеріалів. Необхідно регулярно чистити внутрішні частини комп'ютерів та інше обладнання від пилу, комп'ютери розміщувати на окремих неспарюваних столах. Для запобігання іскроутворення необхідно рідше вставляти і виймати вилки з розеток. Щодо розташування робочих місць, то вони мають бути розміщені на відстані, що не є меншою 1,5 м від стіни з вікнами, та від інших стін на відстані близько 1 м, між бічними поверхнями ВДТ - 1,2 м; від задньої площини одного ВДТ до іншого екрану - 2,5 м.

Конструкція робочої поверхні користувача ВДТ повинна забезпечувати підтримку оптимального робочого положення. Сприятливе робоче положення в процесі роботи за комп’ютером забезпечується налаштуванням висоти робочого столу, підставки для ніг і стільця.

Важливою є форма спинки стільця, яка повинна повторювати форму спини працівника. Висота стільця має бути такою, щоб користувач не відчував тиску на стегна чи куприк. Бажано обладнати крісло підлокітниками. Потрібно встановити їх так, щоб не довелося тягнутися до клавіатури.

Порівняно з вікнами робоче місце доцільно розмістити таким чином, щоб на нього природне світло потрапляло з бокової частини, переважно зліва. Робочі зони потрібно розташовувати так, щоб пряме світло не потрапляло в очі. Рекомендовано розміщувати джерела світла по обидва боки екрану, паралельно напрямку погляду. Для того, щоб уникнути відблисків на екрані, клавіатурі в напрямку очей користувача, від загальних освітлювальних приладів або сонячного світла, потрібно використовувати спеціалізовані фільтри для екранів, захисні навіси, антипроблискові сітки та жалюзі на вікнах.

Екран ВДТ повинен бути розташовуваний на оптимальній відстані від очей користувача, яка становить від 600 до 700 мм, але не ближче 600 мм, враховуючи розмір буквено-цифрових символів та знаків.

При обладнанні робочого місця ВДТ лазерним принтером параметри лазерного випромінювання повинні відповідати вимогам [ДСанПІН 3.3.2.007-98].

Дотримання всіх необхідних заходів з охорони праці забезпечує комфортні умови праці та відсутність шкоди для здоров'я, що сприяє підвищенню продуктивності праці, а також меншому виснаженню при роботі за персональним комп'ютером.

4.2 Безпека в надзвичайних ситуаціях. Пожежна безпека при улаштуванні електроустановок

У процесі написання кваліфікаційної роботи та розробки data--системи для Jupyter Notebook при роботі за персональним комп'ютером існують такі види небезпеки:

- порушення опорно-рухового апарату через тривалі не динамічні навантаження при роботі з ПК;
- незадовільні ергономічні характеристики робочого місця внаслідок нераціонального планування робочого місця, може призвести до уражень електричним струмом, механічних травм, та розладів опорно-рухового апарату;
- ризик ураження електричним струмом через недотримання правил електробезпеки або несправність електроприладів;
- негативний вплив недостатнього освітлення робочої зони на зір і продуктивність праці користувача ЕОМ, внаслідок несправності освітлювальних приладів або неправильної конструкції системи освітлення;
- негативний вплив незадовільних параметрів повітряного середовища робочої зони на здоров'я працівника, в результаті неправильної конструкції вентиляційної системи або її несправності;
- нервово-психічні перевантаження через постійні контакти з колегами по роботі, клієнтами, керівництвом при вирішенні робочих питань, які можуть

носити конфліктний характер і призвести до емоційної нестабільності та дискомфорту, внутрішнього подразнення, та захворювань нервової системи;

- негативний вплив підвищеного рівня шуму на психо-емоційний стан працівника, що пов’язано із використанням застарілого периферійного обладнання, освітлювальних приладів, копіювального обладнання, кондиціонерів;

- ризик пожежі внаслідок несправності електрообладнання, недотримання або порушення правил пожежної безпеки обслуговуючим персоналом, що може призвести до пожежі та неправильні дії персоналу в надзвичайних ситуаціях.

Відповідно до «Правил улаштування електроустановок» реалізовані такі групи заходів електробезпеки: конструктивні заходи забезпечують захист від випадкового контакту з струмопровідними частинами за допомогою їх ізоляції та захисних оболонок. Оскільки згідно з [НПАОП 40.1-1.32-01 "Правила встановлення електроустановок. Електричне обладнання спеціальних установок"] офісні приміщення в основному відносяться до класу пожежонебезпечної зони П-IIа (приміщення, в яких розташовані тверді горючі речовини), тому ступінь забезпечується захист ізоляції обладнання IP44.

Приміщення, в яких працюють розробники програмного забезпечення, класифікуються як приміщення без підвищеного ризику ураження електричним струмом. Обладнання, яке використовується в цих кімнатах, є споживачем електроенергії, що живиться від мережі змінного струму 220 В від мережі із заземленою нейтраллю, і належить до електричних установок до 1000 В закритої конструкції.

Іншим важливим завданням безпеки в надзвичайних ситуаціях є створення заходів пожежної безпеки. Закон України «Про пожежну безпеку» визначає загальні правові, соціальні і економічні основи забезпечення пожежної безпеки на території України, регулює відносини між державними органами, юридичними та фізичними особами у цій галузі, незалежно від виду їх діяльності та форм права власності.

Пожежна безпека - це стан об’єкта, при якому можливість виникнення та розвитку пожежі та вплив її небезпечних факторів на людей виключається з

регульованою ймовірністю, а також забезпечується захист матеріальних цінностей. Для забезпечення пожежної безпеки в закладах проводиться протипожежна профілактика, що включає комплекс організаційно-технічних заходів, спрямованих на забезпечення безпеки людей, запобігання пожежі, обмеження її поширення, а також створення умов для успішного подолання пожежі.

Для ліквідації пожежі на початковій стадії її розвитку персонал об’єктів повинен використовувати первинне обладнання для пожежогасіння. До них відносяться:

- вогнегасники;
- протипожежне обладнання (покривала з негорючих теплоізоляційних тканин, пожежні відра, ящики з піском, лопати, ломи, сокири і тп);
- автоматичні системи пожежогасіння.

Первинне обладнання пожежогасіння, може розташовуватися як окремо, так і у складі протипожежних щитів. Це залежить від агрегатного стану та особливостей горіння різних горючих речовин та матеріалів пожежі [ДБН В. 1.1.7-2016 «Пожежна безпека об’єктів будівництва. Загальні вимоги»] поділяються на відповідні класи. Офісні приміщення містять дерев'яні меблі, електронне обладнання і паперові матеріали. Клас пожежі в офісному приміщенні [ДБН В.1.1-7:2016 «Пожежна безпека об`єктів будівництва»] - пожежі твердих речовин, переважно органічного походження, горіння яких супроводжується тлінням (деревина, пластик, папір) і визначається як клас А.

Категорія приміщень [ДСТУ Б В.1.1-36:2016 «Норми визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою»] - визначається як категорія D. Визначення типу та розрахунок кількості первинних засобів пожежогасіння обладнання [ДБН В.1.1-7:2016 «Пожежна безпека об`єктів будівництва. Загальні вимоги»]. Крім того, адміністративні приміщення повинні бути оснащені протипожежними датчиками, які реагують на підвищення температури, на дим та полум’я. До прикладу, такі моделі датчиків, як ДТЛ, ІТМ.

Для запобігання пожежі надзвичайно важливо правильно оцінити пожежонебезпеку будівлі, виявити небезпеку та обґрунтувати методи та засоби протипожежної охорони та захисту.

Однією з умов забезпечення пожежної безпеки є усунення можливих джерел займання.

Несправності електропроводки, електричних розеток та вимикачів можуть стати причиною запалення у лабораторії. Необхідно регулярно здійснювати плановий огляд – виявляти та усувати існуючі несправності, щоб уникнути виникнення пожежі, з цих причин.

Іншою причиною запалення у лабораторних приміщеннях може бути несправність електроприладів. Щоб усунути виникнення пожежі, із цих причин, необхідно своєчасно проводити ремонт електроприладів та не використовувати несправні електроприлади.

При обігріванні приміщень електронагрівальними приладами з відкритими нагрівальними елементами може призвести до запалення у лабораторії. З огляду на те, що в приміщенні є папір, який є легкозаймистим матеріалом - відкриті нагрівальні поверхні можуть спричинити пожежу. В лабораторії рекомендується не використовувати відкриті нагрівальні прилади задля запобігання пожежі.

Коротке замикання в електропроводці може призвести до пожежі. Електропроводка повинна бути прихованою, щоб зменшити ймовірність пожежі внаслідок короткого замикання.

Ще однією з причин запалення є влучення блискавки у будівлю. Влітку під час грози може потрапити блискавка в будинок, що призведе до можливої пожежі. Щоб уникнути цього, рекомендується встановити громовідвід на даху будинку.

Недотримання заходів пожежної безпеки та куріння в приміщенні також можуть спричинити пожежу. Для ліквідації пожежі в результаті паління в лабораторії необхідно категорично заборонити куріння і дозволяти це лише в суворо відведеному місці.

Перед початком роботи, з метою запобігання пожежі, необхідно провести інструктаж з протипожежних заходів з працівниками, що працюють у лабораторії,

де необхідно ознайомити працівників з правилами пожежної безпеки, а також навчити їх користуватися первинним обладнанням для гасіння пожежі.

Перше, що необхідно зробити під час виникнення пожежі це відключити електропостачання, зателефонувати до пожежної охорони, евакуювати людей з приміщень відповідно до плана евакуації, та розпочати гасіння полум’я вогнегасниками. Якщо є незначне джерело полум’я, тоді потрібно, використавши підручні засоби, зупинити доступ повітря до джерела займання.

Під час роботи комп’ютерів забороняється ремонтувати їх на робочому місці, не допускається працювати на пошкодженому обладнанні або захаращувати робочі місця матеріалами, які не використовуються для поточних робіт.

Ремонт, будь-яке технічне обслуговування та налагодження комп'ютера, а також інші операції з цього приводу слід проводити лише тоді, коли живлення повністю вимкнено. У тих випадках, коли ремонт та інші процедури неможливо виконати при відключенні електроживлення, необхідно, щоб обладнання та периферійні пристрої були заземлені, ремонт виконували двоє або більше робітників з використанням інструментів з ізольованими ручками, а також килимки-діелектрики на підлогу.

Режим праці та відпочинку працівників електронно-обчислювальної техніки визначається [ДСанНіП 3.3.2-007-98]. Під час робочого дня кожні 40-50 хвилин роботи потрібно робити перерви для відпочинку, які тривають близько 3-5-хвилин. Загальна тривалість роботи на день не повинна перевищувати 4 години, а на тиждень - 20 годин.

Таким чином, при дотриманні правил безпеки в надзвичайних ситуаціях, можна запобігти виникненню критичних результатів.

Програмний продукт data-система для Jupyter Notebook, що описаний у кваліфікаційній роботі розроблений з урахуванням вимог техніки безпеки та пожежної безпеки.

ВИСНОВКИ

У кваліфікаційній роботі представлено постановку завдання щодо виконання проєкту, аналіз предметної області та сучасних напрацювань науки про дані дані та артефакти розробки, проектування та розробка алгоритму обробки даних медичного спрямування. А саме висвітлено обробку статистичних накопичених людством за період пандемії даних по захворюваності на COVID-19 з можливістю візуалізувати програмним способом.

Опираючись на актуальні питання та проблеми областей машинного навчання та медицини сформульовано вимоги до програмної системи.

За результатами виконання роботи отримано програмну систему, яка буде отримувати дані від сервісів або зовнішніх служб (або ж від інших систем), обробляти їх та надавати інформацію у відповідь.

В ході виконання проєкту було розглянуто існуючі рішення в області збору та публікації медичних даних для розробки моделей машинного навчання.

Робота також містить презентацію графічного інтерфейсу програми й з докладним опис графічних результатів моделювання, отриманих за її допомогою.

ПЕРЕЛІК ПОСИЛАНЬ

1. Репозиторій машинного навчання [Електронний ресурс] – Режим доступу: http://archive.ics.uci.edu/ml/about.html
2. Урядовий портал Австралії [Електронний ресурс] – Режим доступу: https://data.gov.au/about
3. Національний сервіс даних Австралії [Електронний ресурс] – Режим доступу: https://www.ands.org.au/about-us
4. Маніфест гнучкої розробки [Електронний ресурс] – Режим доступу: http://www.agilemanifesto.org/.
5. «Hunt J. Introduction. In: Advanced Guide to Python 3 Programming. Undergraduate Topics in Computer Science: Springer Nature Switzerland AG, 2019. 204 p.
6. Unpingco J. Python Programming for Data Analysis: Springer Cham, 2021. 263 p.
7. Igual L., Seguí S. Introduction to Data Science. A Python Approach to Concepts, Techniques and Applications: Springer International Publishing Switzerland, 2017. 218 p.
8. Suzuki J. Statistical Learning with Math and Python: Springer Singapore, 2021. 256 p.
9. Kent D. L., Hubbard S. Data Structures and Algorithms with Python: Springer Cham, 2015. 330 p.
10. Linge S., Langtangen H. P. Programming for Computations - Python: Springer Cham, 2020. 332 p.

ДОДАТКИ

Додаток А. Лістинги коду Python для візуалізації даних на графіках:

```
plt.figure(figsize=(25,10))
ax=plt.axes()

ax.set_facecolor('black')
ax.grid(linewidth=0.4, color='#8f8f8f')

plt.xticks(rotation='vertical',
           size='20',
           color='white')#ticks of X

plt.yticks(size='20',color='white')

ax.set_xlabel('\nDistrict',size=25,
              color='#4bb4f2')
ax.set_ylabel('No. of cases\n',size=25,
              color='#4bb4f2')

plt.tick_params(size=20,color='white')

ax.set_title('Maharashtra District wise breakdown\n',
             size=50,color='#28a9ff')
plt.bar(x,co,label='re')
plt.bar(x,re,label='re',color='green')
plt.bar(x,de,label='re',color='red')

for i,j in zip(x,co):
    ax.annotate(str(int(j)),
                xy=(i,j+3),
                color='white',
                size='15')

plt.legend(['Confirmed','Recovered','Deceased'],
           fontsize=20)
```

Додаток Б. Дані, отримані з відомих баз знань про стан захворюваності в різних країнах:

Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	1/30/20	1/31/20
	Afghanistan	33.0	65.0	0	0	0	0	0	0	0	0	0	0
	Albania	41.1533	20.1683	0	0	0	0	0	0	0	0	0	0
	Algeria	28.0339	1.6596	0	0	0	0	0	0	0	0	0	0
	Andorra	42.5063	1.5218	0	0	0	0	0	0	0	0	0	0
	Angola	-11.2027	17.8739	0	0	0	0	0	0	0	0	0	0
	Antigua and Barbuda	17.0608	-61.7964	0	0	0	0	0	0	0	0	0	0
	Argentina	-38.4161	-63.6167	0	0	0	0	0	0	0	0	0	0
	Armenia	40.0691	45.0382	0	0	0	0	0	0	0	0	0	0
Australian Capital Territory	Australia	-35.4735	149.0124	0	0	0	0	0	0	0	0	0	0
New South Wales	Australia	-33.8688	151.2093	0	0	0	0	3	4	4	4	4	4
Northern Territory	Australia	-12.4634	130.8456	0	0	0	0	0	0	0	0	0	0
Queensland	Australia	-28.0167	153.4	0	0	0	0	0	0	0	1	3	2
South Australia	Australia	-34.9285	138.6007	0	0	0	0	0	0	0	0	0	0
Tasmania	Australia	-41.4545	145.9707	0	0	0	0	0	0	0	0	0	0
Victoria	Australia	-37.8136	144.9631	0	0	0	0	1	1	1	1	2	3
Western Australia	Australia	-31.9505	115.8605	0	0	0	0	0	0	0	0	0	0
	Austria	47.5162	14.5501	0	0	0	0	0	0	0	0	0	0
	Azerbaijan	40.1431	47.5769	0	0	0	0	0	0	0	0	0	0
	Bahamas	25.0343	-77.3963	0	0	0	0	0	0	0	0	0	0
Macau	China	22.1667	113.55	1	2	2	2	5	6	7	7	7	7	7
Ningxia	China	37.2692	106.1655	1	1	2	3	4	7	11	12	17	21	26
Qinghai	China	35.7452	95.9956	0	0	0	1	1	6	6	6	8	8	9
Shaanxi	China	35.1917	108.8701	0	3	5	15	22	35	46	56	63	87	10
Shandong	China	36.3427	118.1498	2	6	15	27	46	75	95	130	158	184	20
Shanghai	China	31.201999999999998	121.4491	9	16	20	33	40	53	66	96	112	135	16
Shanxi	China	37.5777	112.2922	1	1	1	6	9	13	27	27	35	39	47
Sichuan	China	30.6171	102.7103	5	8	15	28	44	69	90	108	142	177	20
Tianjin	China	39.3054	117.323	4	4	8	10	14	23	24	27	31	32	41
Tibet	China	31.6927	88.0924	0	0	0	0	0	0	0	0	1	1	1
Xinjiang	China	41.1129	85.2401	0	2	2	3	4	5	10	13	14	17	18
Yunnan	China	24.974	101.48700000000001	1	2	5	11	16	26	44	55	70	83	93
Zhejiang	China	29.1832	120.0934	10	27	43	62	104	128	173	296	428	538	59
	Colombia	4.5709	-74.2973	0	0	0	0	0	0	0	0	0	0	0
	Congo (Brazzaville)	-4.0383	21.7587	0	0	0	0	0	0	0	0	0	0	0
	Congo (Kinshasa)	-4.0383	21.7587	0	0	0	0	0	0	0	0	0	0	0
	Costa Rica	9.7489	-83.7534	0	0	0	0	0	0	0	0	0	0	0
	Cote d'Ivoire	7.54	-5.5471	0	0	0	0	0	0	0	0	0	0	0

Додаток С. Диск з даними проєкту та запискою.