МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет комп'ютерних наук, фізики та математики
Кафедра інформатики, програмної інженерії та економічної кібернетики

МЕТОДИ ВІЗУАЛІЗАЦІЇ ДАНИХ ІЗ ЗАСТОСУВАННЯМ МОДУЛЯ MATPLOTLIB МОВИ ПРОГРАМУВАННЯ PYTHON

Кваліфікаційна робота (проєкт)

на здобуття ступеня вищої освіти "бакалавр"

Виконав : студент 4 курсу 431 групи
Спеціальності 122 Комп'ютерні науки
Освітньо-професійної програми:
Комп'ютерні науки
Коржиневський Дмитро Сергійович
Керівник доктор технічних наук,
професор Бабічев С.А.
Рецензент кандидатка педагогічних наук, доцентка Куриленко Н.В.

Херсон-2021

ЗМІСТ

ВСТУП

Актуальність теми. Людина XXI століття за місяць обробляє таку кількість інформацію, яку людина XVI століття отримувала за все життя. Візуалізація даних - це спосіб ефективного представлення інформації, оскільки ми витрачаємо набагато менше часу на читання даних з графіка, ніж вивчаємо ту саму інформацію в тексті. Людина засвоює 80% побаченої інформації, а всього 20% почутої.

Наш мозок постійно досліджує те, що ми бачимо, і шукає закономірності. Ця система є основою знань. Розуміння того, як працює ця функція, допоможе перетворити багато елементів, що розташовані у таблиці, в нові знання та історію. Зараз, у період розповсюдження вірусу COVID-19 по всій планеті, статистику захворюваності, смертності та одужання відображають за допомогою графіків та діаграм. Взагалі, методи візуалізація даних мають велике значення для розуміння закономірностей, що містять дані, що досліджуються. Їх використовують в науці, в бізнесі, в медицині, інженерії, тощо.

Об'єктом дослідження є методи візуалізації даних різноманітної природи.

Предметом дослідження є засоби створення графіків та діаграм на основі застосування модуля Matplotlib мови програмування Python.

Мета дослідження: опанування методів та засобів створення графіків та діаграм за допомогою бібліотеки Matplotlib мови програмування Python.

Завдання роботи: розглянути різновид графіків та діаграм, засвоїти принципи побудови графіків на основі застосування модуля Matplotlib мови програмування Python, описати середовище розробки та реалізувати візуалізацію даних різної природи із застосуванням функції бібліотеки Matplotlib.

Робота складається з вступу, трьох розділів, висновку та списку використаних джерел.

РОЗДІЛ 1
АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ВІЗУАЛІЗАЦІЇ ДАНИХ.

Існує чотири основних типи візуалізації: порівняння, композиція, розподіл і відношення. На рис. 1.1 представлено пояснення вибору правильного типу діаграми .

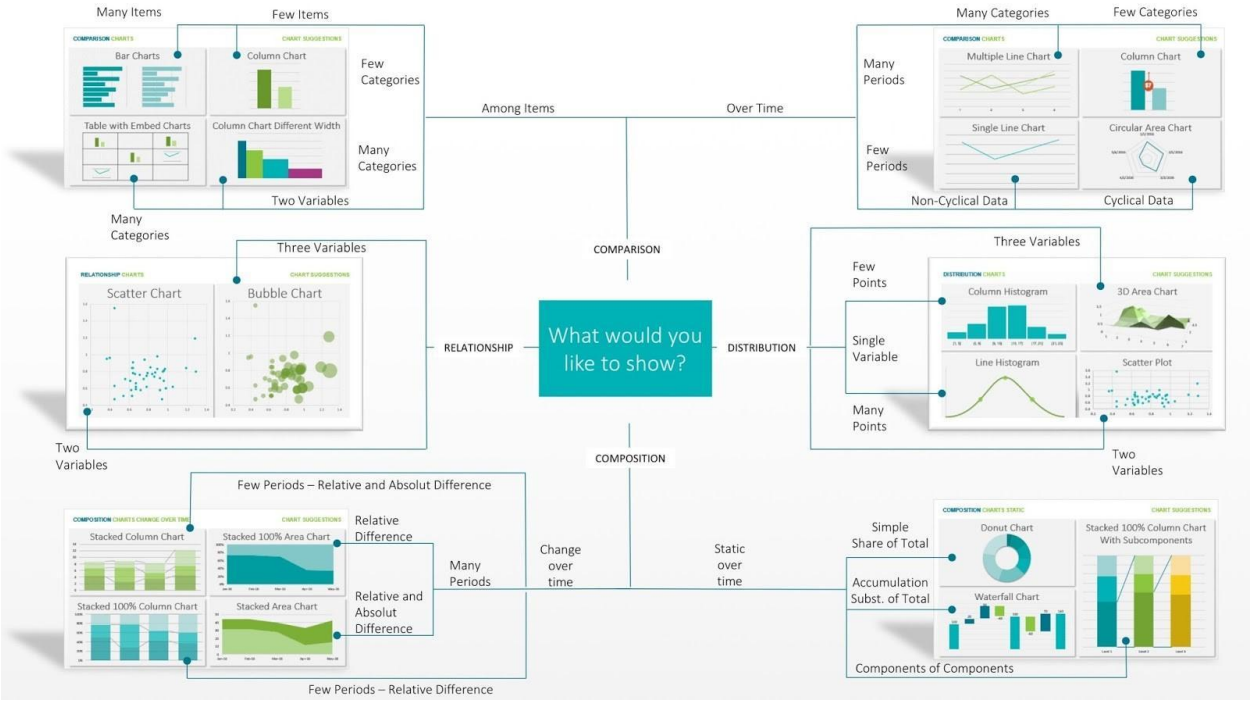

Рисунок 1.1 – Вибір правильного типу діаграми

У кожного типу діаграми є свої різновиди:

1.1 Точкова діаграма

Діаграма розсіювання - це тип діаграми для відображення значень, як правило, для двох змінних для набору даних з використанням декартових координат (рис. 1.2).

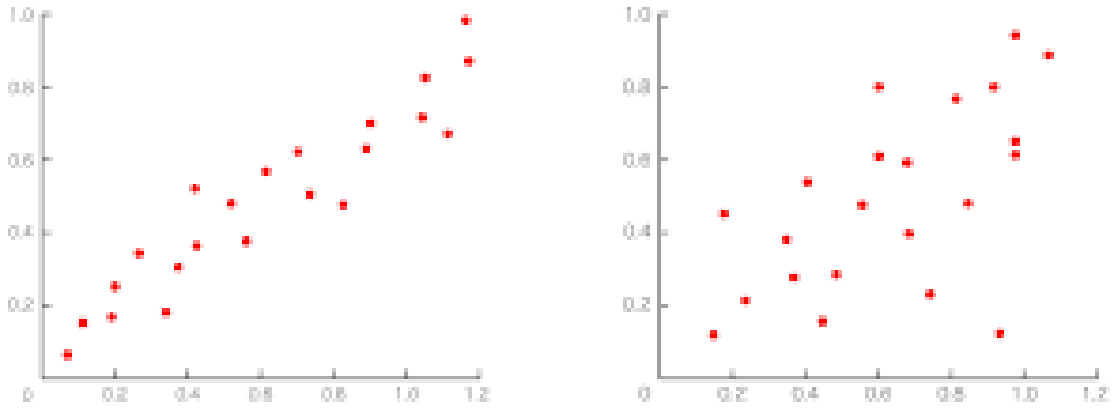

Рисунок 1.2 – Діаграма розсіювання

Дані відображаються, як набір точок, кожна з яких має значення однієї змінної, яка визначає розташування на горизонтальній осі і значення іншої змінної, що визначає положення по вертикальній осі. Цей тип графіка дозволяє нам проводити дослідницький аналіз відповідних залежностей змінних для того, щоб визначити послідовність кроків обробки даних . Слід зазначити, що точки, які відповідають одним і тим же об'єктам, можуть бути позначені як кольором, так і розміром. У другому випадку ми отримуємо бульбашкову діаграму, яка є різновидом діаграми розсіювання, в якій точки даних замінюються бульбашками, та додатковий вимір даних представлений у розмірі бульбашок. Діаграми розсіювання допомагають вирішувати проблеми, показуючи взаємозв'язок між змінними.

У разі використання більш ніж двох змінних (дані з великими розмірами), ми можемо відобразити ці дані за допомогою матриці діаграми розсіювання (рис. 1.3). [3]

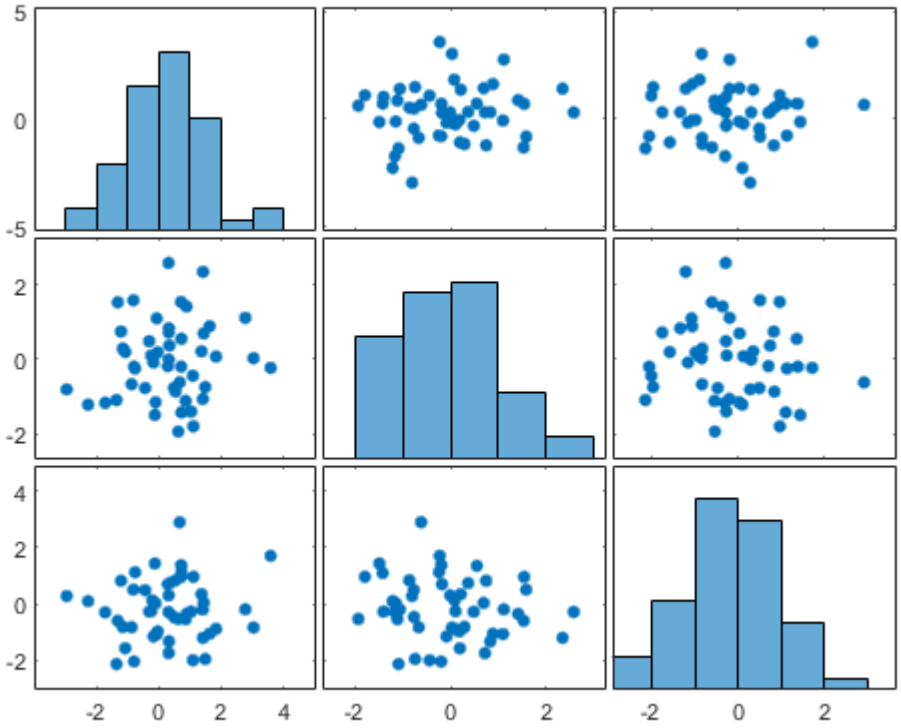

Рисунок 1.3 – Матриця діаграми розсіювання

Матриця діаграми розсіювання показує всі графіки попарного розсіювання змінних на одному зображенні з безліччю графіків розсіювання у форматі матриці.

1.2 Лінійний графік

Лінійний графік (рис. 1.4) являє собою графічне відображення функціонального взаємозв'язку відклику Y при різних значеннях незалежної змінної X. У випадку, якщо змінна X є часовою змінною, маємо часовий ряд. Приклади відображення часових рядів представлені на (рис. 1.5).

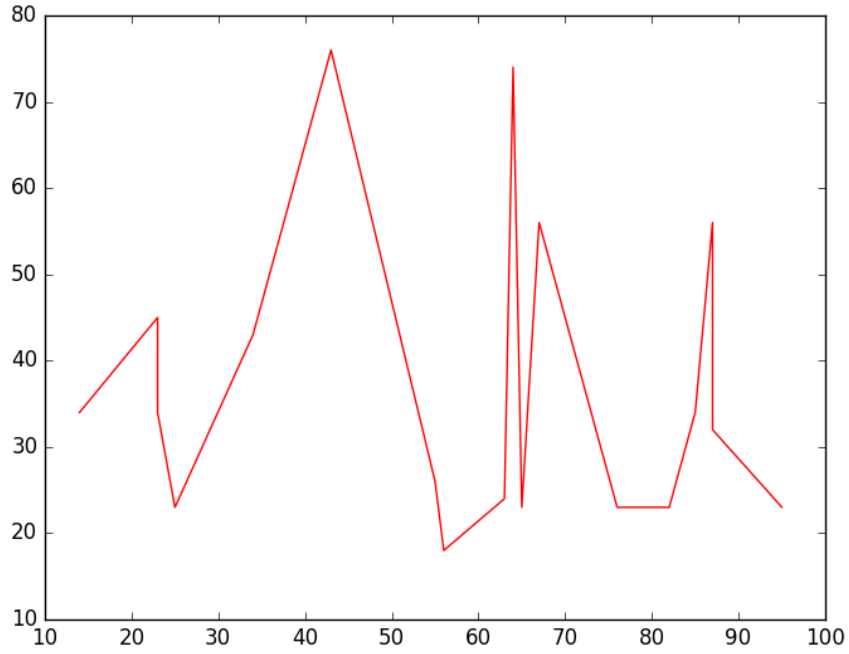

Рисунок 1.4 – Лінійний графік

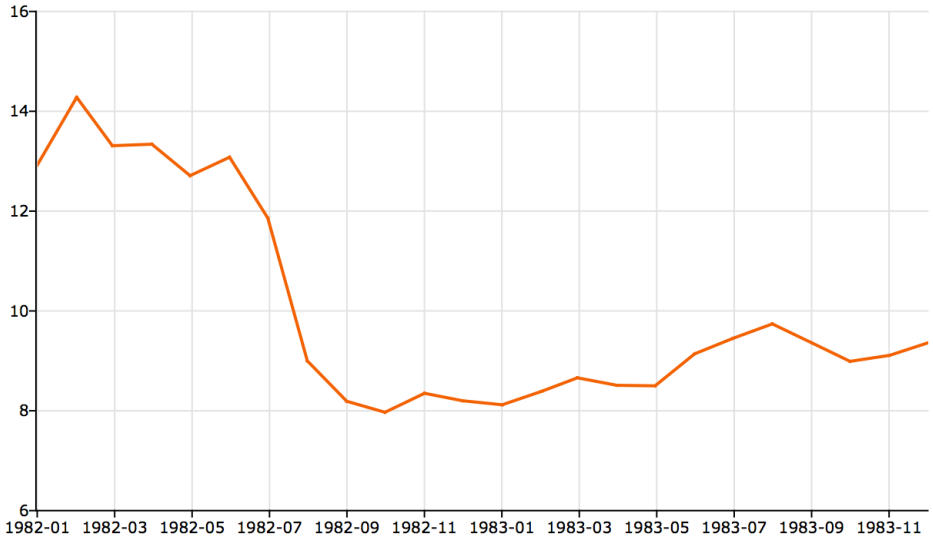

Рисунок 1.5 – Лінійний графік часового ряду

Візуальний аналіз часових рядів дозволяє оцінити динаміку зміни відповідних процесів з метою їх подальшого прогнозування.

1.3 Гістограма та стовпчаста діаграма

Стовпчаста діаграма (рис. 1.6) - це діаграма, яка дозволяє відображати категоричні дані даних прямокутними смугами з довжиною, пропорційною значенням, які вони представляють. Смуги можуть бути нанесені вертикально або горизонтально (рис. 1.6).

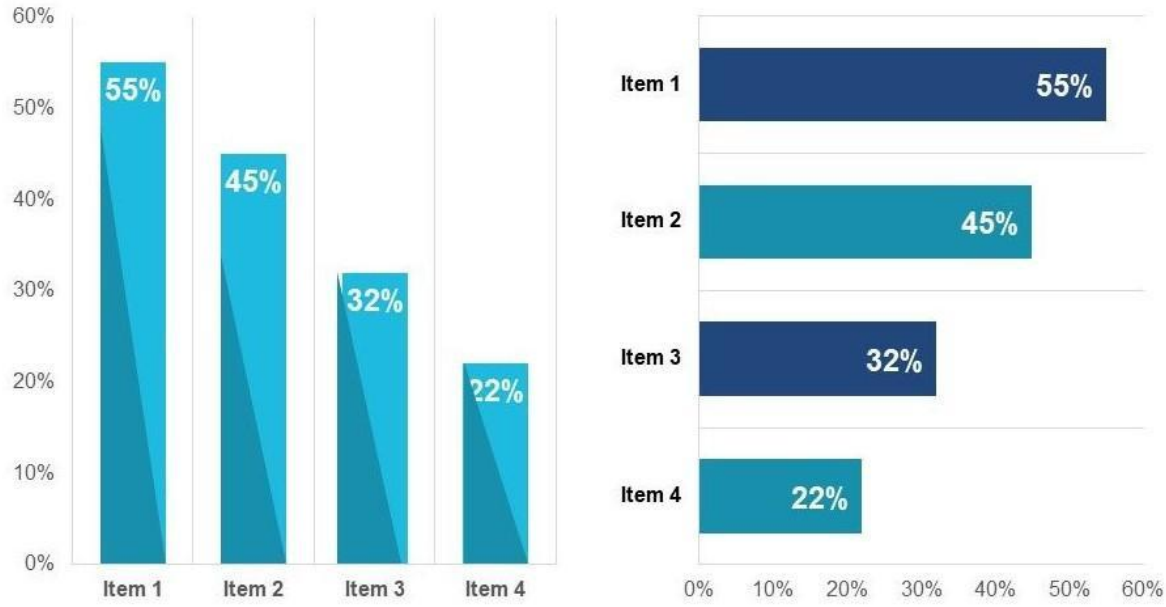

Рисунок 1.6 – Вертикальна та горизонтальна стовпчаста діаграма

Гістограма показує порівняння між дискретними категоріями. Одна вісь гістограми показує конкретні категорії, що порівнюються, а інша вісь, являє собою виміряне значення. На відміну від стовпчастих діаграм, гістограми відображають статистичну інформацію, яка використовує прямокутники, щоб показати частоту елементів даних в послідовні числові інтервали рівного розміру (рис. 1.7). [8]

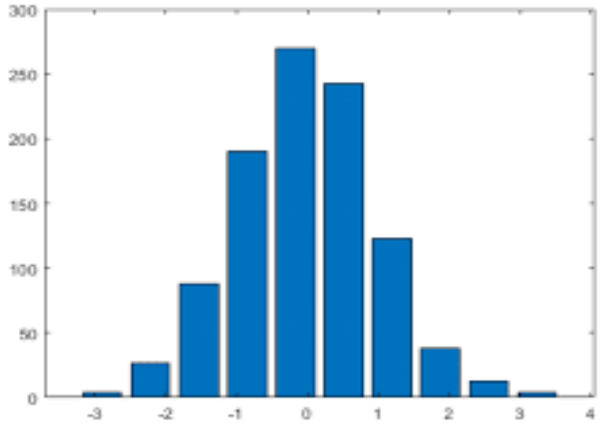

Рисунок 1.7 – Гістограма

У деяких випадках гістограми представляють собою стовпці, згруповані в декілька стовпців, показуючи значення більш ніж одної вимірюваної змінної (рис. 1.8).

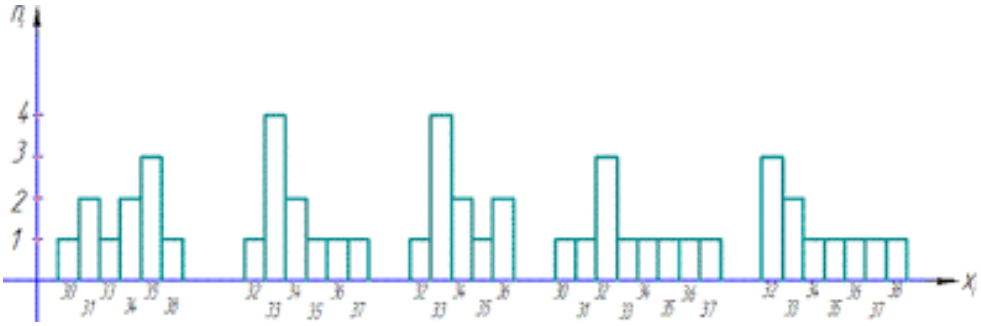

Рисунок 1.8 – Гістограма зі згрупованими стовпцями

Створення гістограми передбачає сортування аналізованої змінної по зростанню, потім вони діляться на кластери з деякими інтервалами. В результаті отримана гістограма виглядає, як набір прямокутних смуг, ширина яких відповідає відповідним значенням інтервалу, а довжина смуг пропорційна частоті елементів даних у відповідних інтервалах. [8]

1.4 Кругова діаграма

Кругова діаграма (секторна діаграма) є альтернативою стовпчастій діаграмі. Це особливий вид діаграми, в якій використовуються «секторні зрізи» для відображення відносних обсягів даних. Кругова діаграма розподілена на сектори, де кожен з них показує відносний розмір відповідної величини. На Рис. 1.9 зображені приклади кругових діаграм для аналізу даних з різними кольорами секторів та використанням різних параметрів.

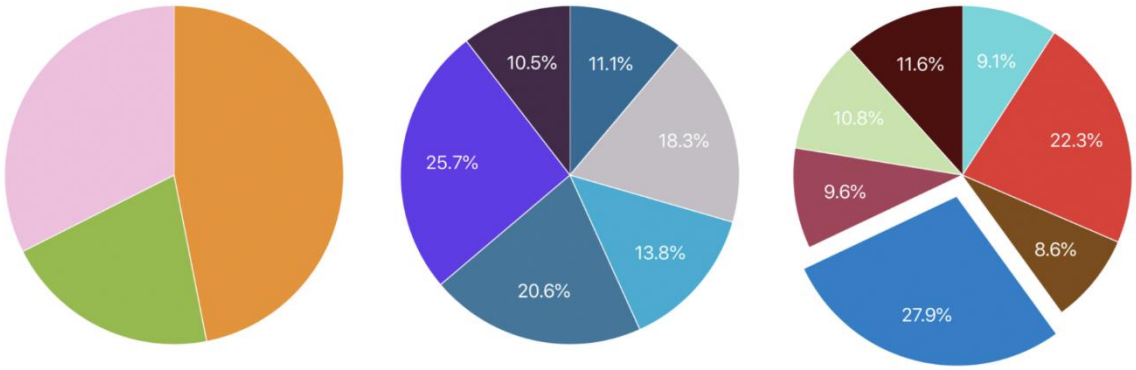

Рисунок 1.9 – Кругові діаграми

Проте, вагомим недоліком такої діаграми є те, що вона не підходить для роботи з великими обсягами інформації. Якщо буде велика кількість секторів, вони будуть малого розміру та їх буде важко позначати різними кольорами або текстурами. Людям, які будуть розглядувати таку діаграму важко буде її розуміти. [3]

1.5 Коробковий графік

Коробковий графік - це графічне представлення статистичних даних на основі мінімуму, першого квартилю Q1 (25-й процентиль), медіани, третього квартилю Q3 (75-й процентиль) і максимуму. На рис. 1.10 зображено загальний приклад коробкового графіку з максимумом, третім квартилем, медіаною, першим квартилем, мінімальними значеннями та викидами.

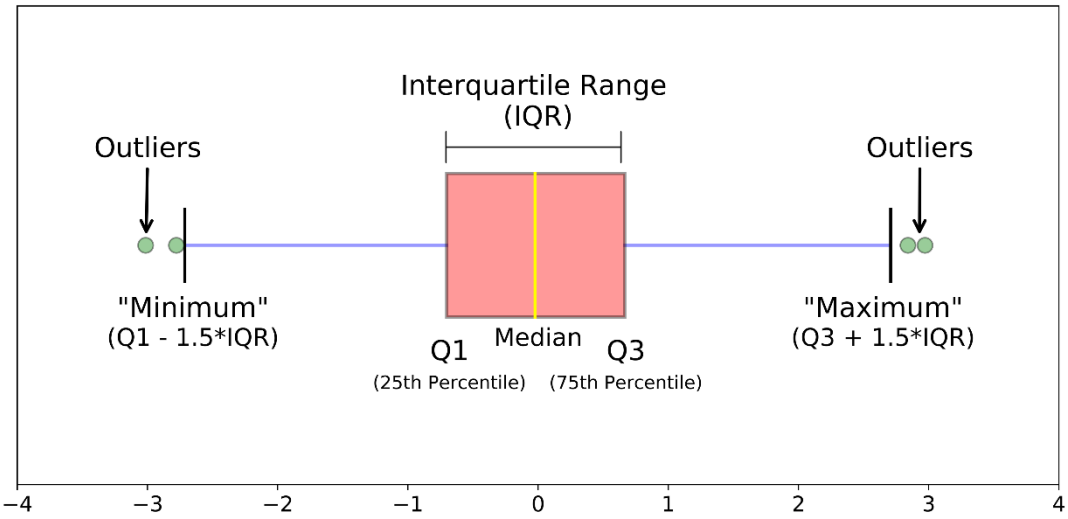

Рисунок 1.10 – Коробковий графік

Викиди представляють значення, що виходять за межі 1,5 межквартільного інтервалу (1,5 * IQR). Коробчаті діаграми є чудовим інструментом для порівняння розподілу даних для різних змінних. [3]

1.6 Графік щільності

Графік щільності представляє собою числовий розподіл змінних. Він використовує оцінку щільності ядра, щоб показати щільність ймовірності функції змінної. Цей графік є різновидом гістограми, яка використовує згладжування ядра для побудови графіків значень, дозволяючи більш плавний розподіл, згладжуючи шум. Піки графіку щільності допомагають відобразити, на якому інтервалі зосереджено значення змінної. На рис. 1.11 представлений графік щільності у випадку нормального розподілу даних. Як можна побачити, максимальне значення графіку щільності відповідає медіані даних у цьому випадку. У загальному випадку, більша щільність розподілу даних відповідає більшому значенню щільності функції. [3]

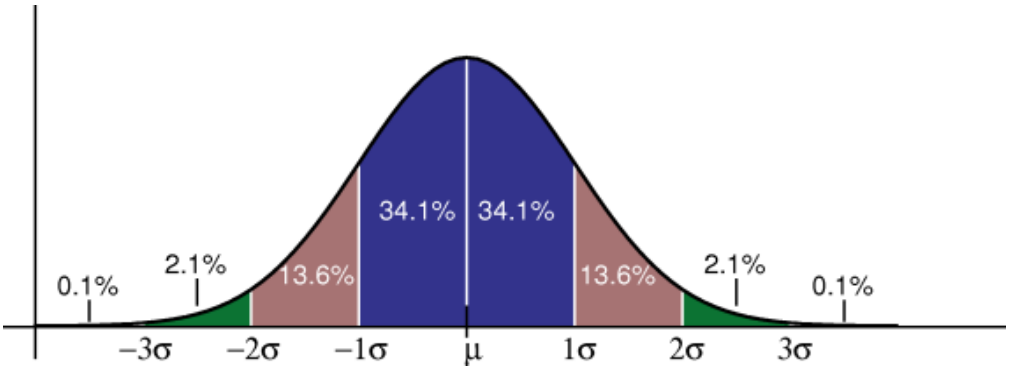

Рисунок 1.11 – Графік щільності у випадку нормального розподілу даних

1.7 Скрипковий графік

Скрипковий графік(Рис.1.12) використовується для візуалізації розподілу даних і їх щільності ймовірностей.

Скрипковий графік - це поєднання коробкового графіку і графіка щільності, розгорнутих і розташованих по обидві сторони для відображення форми розподілу даних. Товста чорна смуга в центрі являє собою міжквартільний діапазон, тонка чорна лінія яка виходить із міжквартільного діапазону являє довірчі інтервали з 95% -ною вірогідностю, а біла точка - це медіана.

Коробкові графіки мають обмеження у відображенні даних, оскільки їх візуальна простота приховує ряд істотних деталей щодо того, яким чином розподіляються значення даних. Наприклад, за допомогою коробкового графіку неможна побачити, який це розподіл: бімодальний або мультимодальний. У той же час скрипковий графік дозволяє відобразити більше інформації, але при цьому він у порівнянні з коробковим графіком не такий згладженний. [3]

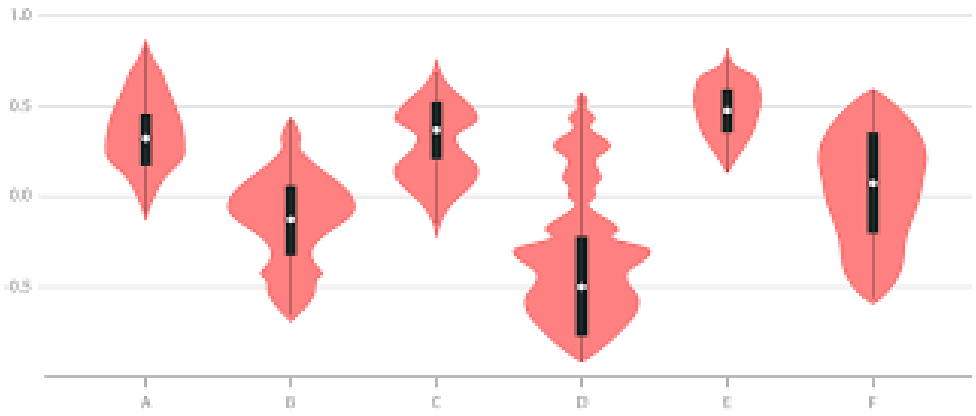

Рисунок 1.12 – Скрипковий графік

1.8 Теплова карта

Теплова карта (рис. 1.13) - це один з методів відображення тривимірних даних. Перші дві змінні x і y визначають осі координат X і Y, а третя змінна z визначає колір у відповідному сегменті. Таким чином, проводиться аналіз, розподіл кольорів дозволяє зрозуміти характер третьої змінної, значення z - розподіл у двовимірному просторі змінних x та y. [11]

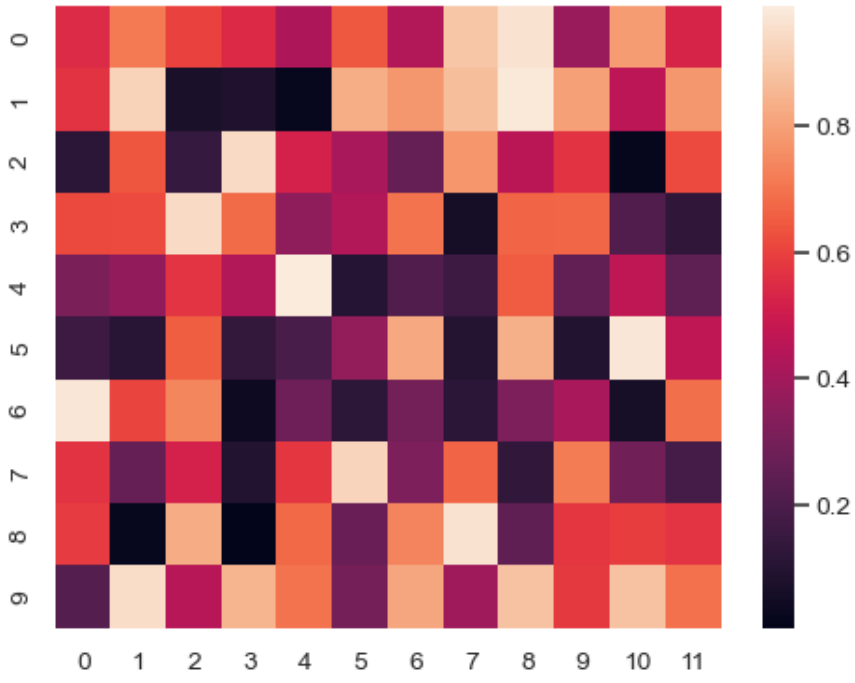

Рисунок 1.13 – Теплова карта

Дендрограма на тепловій карті дозволяє оцінити рівень близькості відповідних змінних на основі аналізу кластерної структури. На (рис. 1.14) зображена дендрограма разом з тепловою картою.

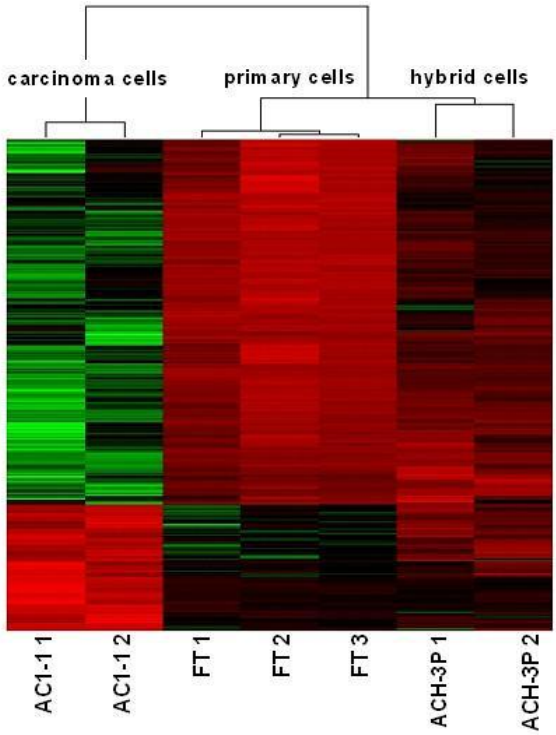

Рисунок 1.14 – Дендрограма з тепловою картою

1.9 Діаграма у паралельних координатах

Діагарама у паралельних координатах - тип візуалізації даних, що використовується для побудови багатовимірних графіків (рис. 1.15).

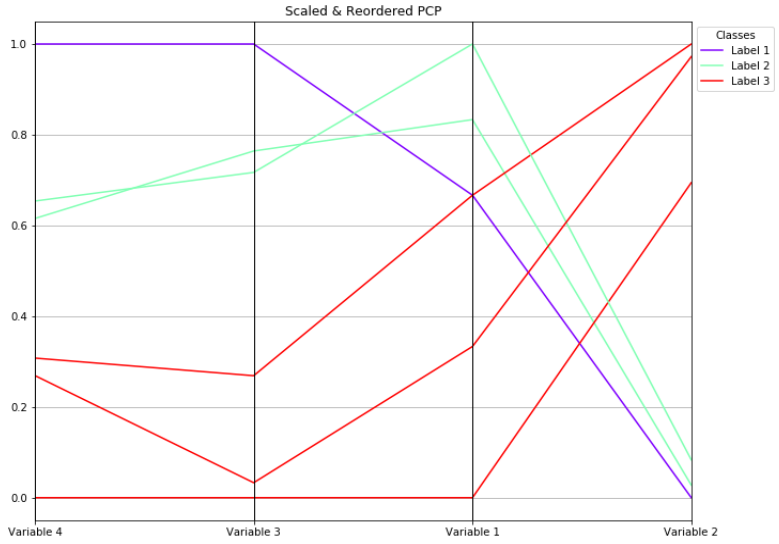

Рисунок 1.15 – Діаграма у паралельних координатах

Графіки у паралельних координат ідеально підходять для порівняння багатьох змінних разом та показує взаємозв'язок між ними. У цьому типі графіку, кожна зі змінних має свою вісь і всі вісі розміщені паралельно одна одній.

Кожна вісь може мати різний масштаб, оскільки кожна змінна працює з різною одиницею вимірювання, або дані, розподілені вздовж вісей можна нормалізувати, щоб усі шкали були рівномірними. Значення побудовані як ряд ліній, що з'єднуються по всіх вісях. Це означає, що кожна із ліній - це сукупність точок, розміщених на кожній вісі і з'єднаних між собою. [3]

РОЗДІЛ 2
СТРУКТУРА ТА ОСОБЛИВОСТІ ЗАСТОСУВАННЯ МОДУЛЯ MATPLOTLIB МОВИ ПРОГРАМУВАННЯ PYTHON

2.1 Бібліотека Matplotlib

Бібліотека Matplotlib - це двовимірна графічна бібліотека для мови програмування Python, яку можна використовувати для створення високоякісних малюнків у різних форматах. Matplotlib - пакетний модуль для Python. [1]

Matplotlib у свою чергу складається з великої кількості модулів. Модулі упаковані різними класами та функціями, які є ієрархічно пов'язаними.

Створення малюнку в Matplotlib схоже з роботою художника. Йому потрібно взяти основу (полотно або папір), інструменти (пензлі чи олівці), скласти концепцію картини, і нарешті намалювати малюнок деталь за деталлю.

Сама бібліотека виступає тут як інтерпретатор. Користувач повинен контролювати дії художника- Matplotlib, визначаючи, що саме малювати і за допомогою яких інструментів. Таким чином, користувач бібліотеки Matplotlib виступає в ролі адміністратора.

Оскільки Matplotlib організована ієрархічно, і функції найвищого рівня є найпростішими для розуміння людиною, пізнання Matplotlib починаються з інтерфейсу matplotlib.pyplot (найвищого рівня). Отже, щоб намалювати гістограму за допомогою цього модуля, потрібно викликати одну команду: matplotlib.pyplot.hist (arr).

Користувачеві не потрібно думати про те, як саме бібліотека намалювала цю діаграму. Якби ми самі намалювали гістограму, ми зрозуміли б, що вона складається з фігур, які повторюються: прямокутників. А щоб

намалювати прямокутник, потрібно знати хоча б координату одного кута та ширину або довжину.

Цей приклад показує ієрархію малюнків, коли остаточна діаграма (високий рівень) складається з простих геометричних фігур (нижній рівень, середній), створених кількома загальними методами малювання (нижній рівень). Якби кожен малюнок потрібно було створити з нуля, це було б дуже довго і нудно.

Інтерфейс matplotlib.pyplot - це набір команд і функцій, які роблять синтаксис для побудови команд matploltib подібним до тих, що використовуються в середовищі MATLAB (c). Спочатку Matplotlib планувався як безкоштовна альтернатива MATLAB (c), де як креслення, так і чисельний аналіз будуть доступні в одному середовищі. Так з’явився pylab у Matplotlib, який поєднує модулі pyplot та numpy в одному просторі імен.

У той же час для більш серйозних завдань (вбудовування Matplotlib у графічний інтерфейс користувача) потрібний більший контроль над процесом та більша гнучкість, ніж модулі pyplot та numpy. Потрібен доступ до можливостей бібліотеки нижчого рівня, яка реалізована в об’єктно-орієнтованому стилі. Об’єктно-орієнтований стиль помітно складніший для початківців і вимагає знання конкретної роботи в класах та їх методів, але надає більше можливостей для взаємодії з бібліотекою Matplotlib. [2]

2.2 Ієрархічна структура малюнка в Matplotlib

Малюнок(Figure) - це об’єкт верхнього рівня, який містить одну або кілька областей малювання (Axes), елементи малювання Artisits (заголовки, легенда тощо) та полотно. Figure може мати кілька областей малювання(Axes), але дана Axe може належати лише одній Figure.

Область малювання(Axes) є об'єктом середнього рівня, який є, напевно, головним об'єктом роботи з графікою Matplotlib в об'єктно-орієнтованої

стилі. Це те, що асоціюється зі словом "plot", це частина зображення з простором даних. Кожна область малювання Axes містить дві (або три в разі тривимірних даних) координатних осі (Axis об'єктів), які впорядковують відображення даних.

Координатна вісь(Axis) є об'єктом середнього рівня, які визначають область зміни даних, на них наносяться поділи (ticks) і підписи до поділок (ticklabels). Розташування поділів визначається об'єктом Locator, а підписи поділок обробляє об'єкт Formatter. Конфігурація координатних осей полягає в комбінуванні різних властивостей об'єктів Locator і Formatter.

Елементи малюнка Artists є як би червоною лінією для всіх ієрархічних рівнів. Практично все, що відображається на малюнку є елементом малюнка (Artist), навіть об'єкти Figure, Axes і Axis. Елементи малюнка Artists включають в себе такі прості об'єкти як текст (Text), плоска лінія (Line2D), фігура (Patch) та інші.

Коли відбувається відображення малюнка (figure rendering), всі елементи малюнка Artists наносяться на основу-полотно (Canvas). Велика частина з них пов'язується з областю малювання Axes. Також елемент малюнка не може спільно використовуватися декількома областями Axes або бути переміщений з одного на інший. [2]

2.3 Інтерфейс прикладного програмування Matplotlib API

У Matplotlib функції логічно розділені між декількома об'єктами, причому кожен з них сам має досить складну структуру. Можна виділити три рівні інтерфейсу прикладного програмування (matplotlib API):

1) matplotlib.backend_bases.FigureCanvas - абстрактний базовий клас, який дозволяє малювати і візуалізувати результати команд;

2) matplotlib.backend_bases.Renderer - об'єкт (абстрактний клас), який знає як малювати на FigureCanvas;

3) matplotlib.artist.Artist - об'єкт, який знає, як використовувати візуалізатор (renderer), щоб малювати на полотні (canvas).

FigureCanvas і Renderer обробляють деталі, необхідні для взаємодії із засобами призначеного для користувача інтерфейсу, так як це робить WxPython або мову малювання PostScript. Artist обробляє всі конструкції високого рівня такі як представлення і розташування малюнка, тексту і ліній.
Artists - об'єкт дає ті можливості високого рівня для створення малюнків.
Існує два типи об'єктів-класів Artists:

- Примітиви (primitives);
- Контейнери (containers).

Примітиви є стандартними графічними об'єктами: плоска лінія (Line2D), прямокутник (Rectangle), текст (Text), зображення (AxesImage), тощо. Контейнери - це об'єкти-сховища, на які можна наносити графічні примітиви. До контейнерів відносяться: малюнок (Figure), область малювання (Axes), координатна вісь (Axis), ділення (Ticks). Саме за допомогою звернень до різних контейнерів класу Artists, об'єднаних логічно в єдину структуру, буде здійснюватися налаштування малюнків в matplotlib.
Всього існує 4 види Artists контейнерів:

1. Figure container - це контейнер найвищого рівня. На ньому розташовуються всі інші контейнери і графічні примітиви.
2. Axes container - дуже важливий контейнер, так як саме з ним найчастіше працює користувач. Примірники Axes - це області, розташовані в контейнері Figure, для яких можна задавати координатну систему. На ньому розташовуються всі інші контейнери, крім Figure, і графічних примітивів. Це області на малюнку, на яких розташовуються

графіки та діаграми, в які вставляються зображення і т.д. Мультивіконні малюнки складаються з набору областей Axes.

3. Axis container - контейнер обслуговує екземпляри Axes. Він відповідає за створення координатних осей, на які будуть наноситися поділи осей, підписи поділок і ліній допоміжної сітки. Його спеціалізація - це розташування поділок і ліній, їх позиціювання та форматування підписів поділів, їх відображення.

4. Tick container - контейнер нижчого рівня. Його спеціалізація - задавати характеристики (колір, товщину ліній) ліній сітки, поділів і їх підписів (розміри і типи шрифтів).

При створенні малюнку в Matplotlib спочатку створюється екземпляр класу Figure (Figure instance), на якому виділяється одна або кілька областей Axes (або примірників Subplot), і використовуються допоміжні методи екземпляра класу Axes (Axes instance) для створення графічних примітивів (primitives). Якщо автоматично підібрані характеристики координатної сітки, поділів і їх підписів не влаштовують користувача, то вони налаштовуються за допомогою примірників контейнерів Axis і Tick, які завжди присутні на створеній області малювання Axes. [2]

2.4 Інтерфейс Pyplot

Інтерфейс Pyplot дозволяє користувачеві зосередитися на виборі готових рішень і налаштування базових параметрів малюнка. Це його головна перевага, тому вивчення matplotlib найкраще починати саме з інтерфейсу Pyplot. Існує стандарт виклику Pyplot в Python:

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

Малюнки в Matplotlib створюються шляхом послідовного виклику команд: або в інтерактивному режимі (в консолі), або в скрипті (текстовий файл з python-кодом). Графічні елементи (точки, лінії, фігури і т.д.) нашаровуються одна на іншу послідовно. При цьому наступні перекривають попередні, якщо вони займають спільні ділянки на малюнку (регулюється параметром zorder).

У Matplotlib працює правило "поточної області" ("current axes"), яке означає, що всі графічні елементи наносяться на поточну область малювання. Незважаючи на те, що областей малювання може бути кілька, одна з них завжди є поточною.

Найголовнішим об'єктом у Matplotlib є малюнок Figure. Тому при створенні наукової графіки потрібно починати саме зі створення малюнка. Створити малюнок у Matplotlib означає вказати форму, розміри і властивості основи-полотна (canvas), на якому буде створюватися майбутній графік.

Створити малюнок figure дозволяє метод plt.figure(). Після виклику будь-якої графічної команди, тобто функції, яка створює будь-якої графічний об'єкт, наприклад, plt.scatter () або plt.plot (), завжди існує хоча б одна область для малювання (за замовчуванням прямокутної форми). [2]

Щоб результат малювання, відобразилося на екрані, можна скористатися командою plt.show (). Будуть показані всі малюнки (figures), які були створені.

2.5 Елементи малюнка Artists

Весь простір малюнка Figure можна використовувати для нанесення інших елементів малюнку, наприклад, контейнерів Axes, графічних примітивів у вигляді ліній, фігур, тексту і так далі. У будь-якому випадку кожен малюнок можна структурно представити таким чином:

1) Область малювання Axes

- Тема області малювання -> plt.title ();

2) Вісь абсцис Xaxis

- Підпис осі абсцис OX -> plt.xlabel ();

3) Вісь абсцис Yaxis

- Підпис осі абсцис OY -> plt.ylabel ();

4) Легенда -> plt.legend ()

5) Колірна шкала -> plt.colorbar ()

- Підпис горизонтальній осі абсцис OY -> cbar.ax.set_xlabel ();

- Підпис вертикальної осі абсцис OY -> cbar.ax.set_ylabel ();

6) Поділу на осі абсцис OX -> plt.xticks ()

7) Поділу на осі ординат OY -> plt.yticks ()

Для кожного з перерахованих рівнів-контейнерів є можливість нанести заголовок (title) або підпис (label). Підписи до малюнку полегшують розуміння того, в яких одиницях представлені дані на графіку або діаграмі.

Також часто на малюнок наносяться лінії допоміжної сітки (grid). У Pyplot вона викликається командою plt.grid(). Допоміжна сітка пов'язана з поділами координатних осей (ticks), які визначаються автоматично виходячи зі значень вибірки. Варто сказати, що у Matplotlib існують головні ділення (major ticks) і допоміжні (minor ticks) для кожної координатної осі. За замовчуванням малюються тільки головні поділи і пов'язані з ними лінії сітки grid.

Якщо на малюнку присутній так званий "mappable object", то на малюнку може бути намальована колірна шкала (colorbar). Для шкали також можна робити підписи вздовж різних сторін. При цьому сама колірна шкала може бути розташована як на поточній області малювання axes, так і на самостійній області малювання. [2]

2.6 Властивості графічних елементів

Різноманіття і зручність створення графіки у Matplotlib забезпечується не тільки за рахунок створених графічних команд, але і за рахунок багатого арсеналу по конфігурації типових форм. Ця установка включає в себе роботу з кольором, формою, типом лінії або маркера, товщиною ліній, ступенем прозорості елементів, розміром і типом шрифту, та іншими властивостями.

Параметри, які визначають ці властивості в різних графічних командах, зазвичай мають однаковий синтаксис. Стандартним способом завдання властивостей будь-якого створюваного об'єкта або методу є передача по ключу: ключ = значення. Найбільш часто зустрічаються назви параметрів зміни властивостей графічних об'єктів перераховані нижче:

- color / colors / c - колір;
- linewidth / linewidths - товщина лінії;
- linestyle - тип лінії;
- alpha - ступінь прозорості (від повністю прозорого 0 до непрозорого 1);
- fontsize - розмір шрифту;
- marker - тип маркера;
- s - розмір маркера в методі plt.scatter (тільки цифри);
- rotation - поворот рядки на X градусів.

При створенні функцій або методів класів, особливо в разі, параметри часто передаються у вигляді об'єднань послідовностей: кортежу або словника. Для цього існують спеціальні символи-приставки: "*" або "**" відповідно. Це особливо корисно у випадках, коли функція або метод може приймати змінне число параметрів. Для передачі кортежу використовується змінна args, а в разі зі словником - kwargs. Якщо перед змінною args вказано символ "*", то всі додаткові аргументи, передані функції або методу, збережуться в args у вигляді кортежу. Якщо перед args буде вказано символ

"**", то всі додаткові параметри будуть розглядатися як пари "ключ - значення" в словнику.

У функціях або методах описуються властивості таких графічних об'єктів як лінія, текст, прямокутник, параметри часто об'єднуються у вигляді послідовностей * args, або словників ** kwargs. Так зручніше при створенні класів і їх методів. [2]

РОЗДІЛ 3
ПРИКЛАДИ ЗАСТОСУВАННЯ МОДУЛЯ MATPLOTLIB ДЛЯ ВІЗУАЛІЗАЦІЇ ДАНИХ

Для застосування модуля matplotlib, я вибрав командну оболонку для інтерактивних обчислень Jupyter Notebook. Jupyter Notebook використовується для роботи з даними, статистичним моделюванням і машинним навчанням. Для початку я встановив Anaconda(дистрибутив Python). В ньому є панель навігації і вже встановлений Jupyter Notebook. [7]

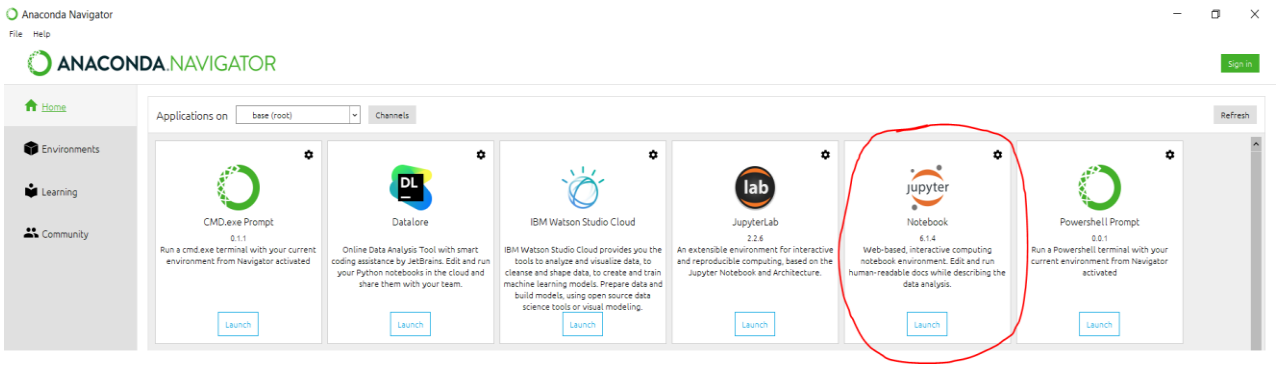

Рисунок 2.1 – Anaconda Navigator.

Далі треба відкрити панель інструментів Jupiter Notebook та створити блокнот.

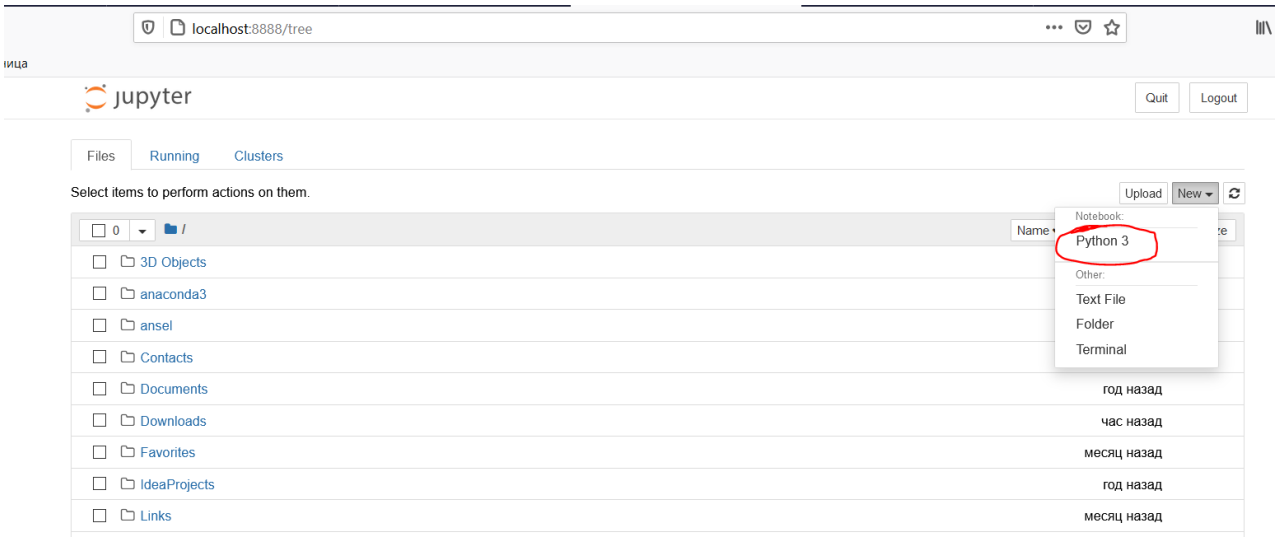

Рисунок 2.2 – створення нового блокноту.

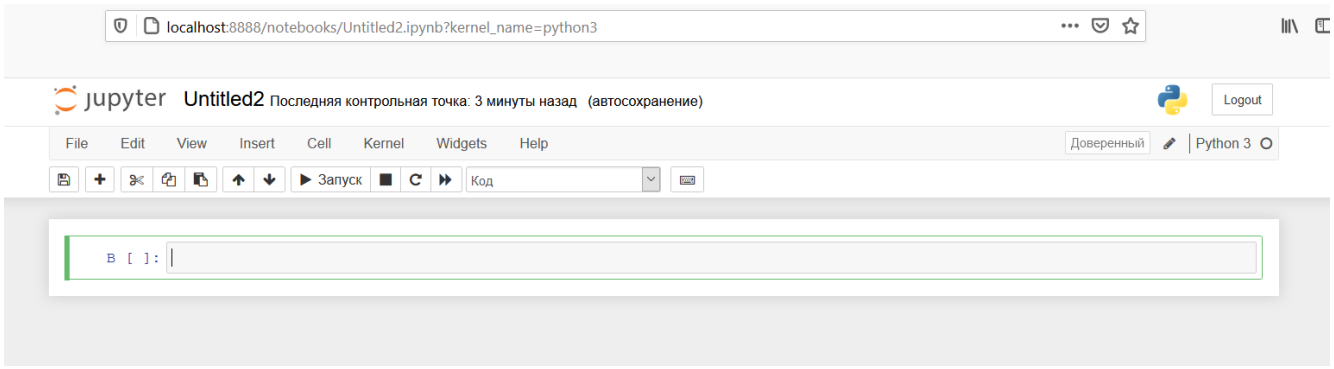

Рисунок 2.3 – інтерфейс блокноту.

Стовпчикову діаграму можна застосовувати в багатьох галузях, наприклад можна зобразити продаж книг за пів року, умовно з березня по серпень.

Спочатку для того, щоб код виповнювався, імпортуються модулі pyplot та numpty. На другому кроці створюється послідовність за допомогою функції np.arange() та масив(values1) зі значеннями для осі X. За допомогою функції plt.title(), задається назва для діаграми. За допомогою функції plt.bar задаються характеристики для стовпців, а за допомогою plt.xticks() – підписуються стовпці. Функція plt.legend() показує легенду діаграми.

```
В [15]: import matplotlib.pyplot as plt
        import numpy as np

        index = np.arange(6)
        values1 = [95,120,156,177,199,180]
        plt.title('Продаж книг за 6 місяців')
        plt.bar(index, values1,alpha=0.7,color='red', label='Кількість екземлярів')
        plt.xticks(index,['Березень','Квітень','Травень','Червень','Липень','Серпень'])
        plt.legend(loc=2)
        plt.show()
```

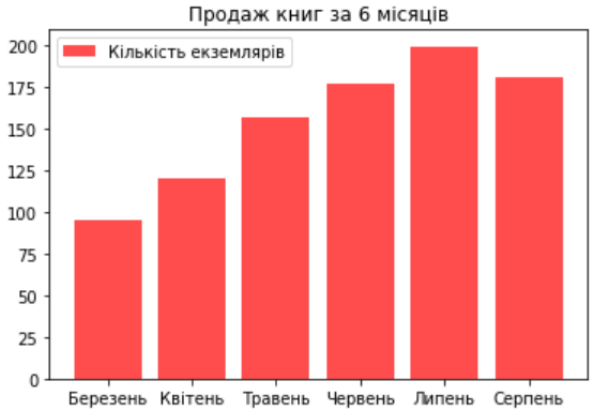

Рисунок 2.4 – Стовпчикова діаграма продажу книг за 6 місяців.

Також можна зобразити топ продаж телефонів за 2020 рік. Були створені масиви за назвами брендів виробників(label), масив з кількістю(у мільйонах) проданих телефонів (values) та позначено кожний сектор відповідним кольором за допомогою масиву colors. За допомогою explode відділився сектор з найбільшим значенням, щоб краще було видно, яка компанія продала більше всього телефонів. Функція plt.pie “малює” саму діаграму, задаються характеристики для відображення діаграми. Функція plt.axis(‘equal’) написана для того, щоб діаграма була ідеально круглою.

```
import matplotlib.pyplot as plt
import numpy as np

labels = ['SAMSUNG','APPLE','XIAOMI','HUAWEI','OPPO']
values = [79.8,41.7,46.2,50.9,31]
colors = ['yellow','green','red','blue','orange']
explode = [0.2,0,0,0,0]
plt.title('Кількість проданих телефонів в 2020 році')
plt.pie(values,labels=labels,colors=colors,explode=explode,shadow=True,autopct='%1.1f%%',startangle=90)
plt.axis('equal')
plt.show()
```

Рисунок 2.5 – Кругова діаграма продажу телефонів за 2020 рік.

На прикладі багаторядної стовпчикової діаграми можна показати скільки, наприклад умовна компанія продавала продукцію різного призначення протягом кожного місяця. Для перетворення звичайного багаторядної стовпчикової діаграми в складену, додається іменований аргумент left в кожну функцію barh(). Кожен об'єкт Series повинен бути присвоєний відповідному аргументу left. Результатом буде створена горизонтальна стовпчикова діаграма.

Рисунок 2.6 – Умовна складена горизонтальна стовпчикова діаграма.

Також був створений лінійний графік для аналізу даних з застосуванням бібліотеки pandas. Ядром цієї бібліотеки являється Dataframe – структура для роботи з різними вимірами. В цій структурі відразу зображено 3 масиви, але відображаються вони, як 3 різні об’єкти. Тобто на цьому графіку умовно зображена зміна значення трьох об’єктів протягом умовного часу.

В [26]:
```
import matplotlib.pyplot as plt
import pandas as pd

data = {'obj1':[1,3,4,3,5,8,8,6,9,9],
        'obj2':[2,4,5,2,4,5,8,4,7,8],
        'obj3':[3,2,3,1,3,3,5,7,10,7]}
df = pd.DataFrame(data)
x = np.arange(10)
plt.axis([0,10,0,10])
plt.plot(x,df)
plt.legend(data, loc=2)
plt.show()
```

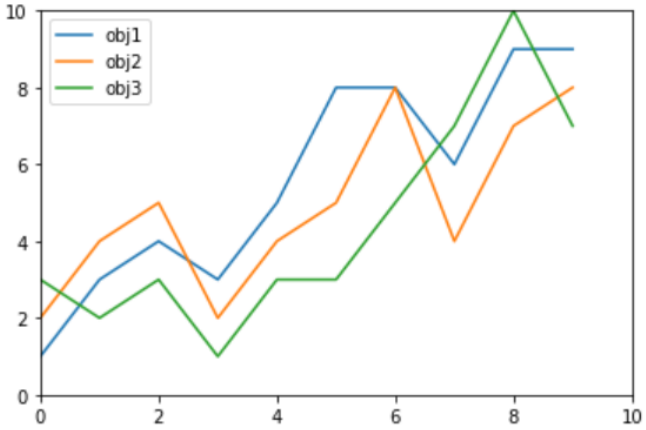

Рисунок 2.7 – Лінійний графік.

ВИСНОВКИ

В даній кваліфікаційній роботі було проаналізовано існуючі методи візуалізації даних. Розібрали для чого потрібні різні діаграми та графіки Можна сказати, що кожний метод кращий по своєму, або краще реалізовується в деяких сферах науки.

Було ознайомлення з принципами побудови цих графіків та діаграм на мові програмування Python. Також пройшов розбір структури бібліотеки Matplotlib. Ознайомилися з ієрархією структури малюнку, його елементами, властивостями графічних елементів та інтерфейсом Matplotlib.

В останньому розділі розібрано приклади діаграм, їх можливе призначення. Встановлення Jupyter Notebook та побудова діаграм та графіків. Діаграми були реалізовані за допомогою бібліотеки Matplotlib, тобто виконано практичну частину кваліфікаційної роботи.

Взагалі візуалізація даних завжди була та буде актуальною, так як люди таку інформацію сприймають набагато краще ніж якусь іншу.

СПИСКИ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Matplotlib. – 2020 – URL : https://uk.wikipedia.org/wiki/Matplotlib [1]

2. Библиотека matplotlib. Pyplot интерфейс. – 2017 – URL : https://nbviewer.jupyter.org/github/whitehorn/Scientific_graphics_in_python/blob/master/P1%20Chapter%201%20Pyplot.ipynb [2]

3. Deepayan S. Lattice: Multivariate Data Visualization with R. Springer, 2008. [3]

4. Python – Modules. Python Basic Tutorial. – 2021 – URL : https://www.tutorialspoint.com/python/python_modules.htm [4]

5. Tutorials matplotlib. – 2021 – URL: https://matplotlib.org/stable/tutorials/index.html [5]

6. Особенности Jupyter Notebook, о которых вы (может быть) не слышали. – 2016 – URL : https://habr.com/ru/company/wunderfund/blog/316826/ [6]

7. Как настроить Jupyter Notebook для Python 3. – 2017 – URL : https://tproger.ru/translations/jupyter-notebook-python-3/ [7]

8. Визуализация данных. Столбчатые и круговые диаграммы. – 2019 – URL : https://devpractice.ru/matplotlib-lesson-4-3-bar-pie/#p21 [8]

9. Добавление легенды. – 2020 – URL : https://pyprog.pro/mpl/mpl_adding_a_legend.html [9]

10. Wickham H. ggplot2: Elegant Graphics for Data Analysis. Springer, 2009. [10]

11. Creating annotated heatmaps. – 2021 – URL : https://matplotlib.org/stable/gallery/images_contours_and_fields/image_annotated_heatmap.html [11]

12. John D. Hunter, Matplotlib: A 2D Graphics Environment. May/June 2007. [12]

13. Sandro Tosi, Matplotlib for Python Developers. November 2009. [13]

14. Fabio Nelli, Python Data Analytics: Data Analysis and Science using pandas, matplotlib and the Python Programming Language. August 2015. [14]

15. Wes McKinney, Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. October 2012. [15]

16. Построение графиков в Python при помощи Matplotlib. – 2020 – URL : https://python-scripts.com/matplotlib [16]
17. 50 оттенков matplotlib — The Master Plots. – 2019 – URL : https://habr.com/ru/post/468295/ [17]
18. Python Plotting With Matplotlib (Guide). – 2018 – URL : https://realpython.com/python-matplotlib-guide/ [18]
19. Введение в визуализацию данных с Matplotlib. – 2020 – URL : https://nagornyy.me/courses/data-science/intro-to-matplotlib/ [19]
20. Jake VanderPlas, Python Data Science Handbook: Essential Tools for Working with Data. November 2016. [20]

ДОДАТОК А