ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Створення онлайн-сервісу для аналізу і візуалізації даних з різних джерел»

на здобуття освітнього ступеня бакалавра
зі спеціальності 123 – Комп'ютерна інженерія
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерна інженерія
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

	Владислав МАРУСЯК
(підпис)	*Ім'я, ПРІЗВИЩЕ здобувача*

Виконав:	здобувач вищої освіти гр.КІД-41 Владислав МАРУСЯК Ім'я, ПРІЗВИЩЕ
Керівник: *науковий ступінь, вчене звання*	Дмитро РОЗМАЇТИЙ Ім'я, ПРІЗВИЩЕ
Рецензент: *науковий ступінь, вчене звання*	 Ім'я, ПРІЗВИЩЕ

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Комп'ютерної інженерії
Ступінь вищої освіти бакалавр
Спеціальність 123 – Комп'ютерна інженерія
Освітньо-професійна програма Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедрою Наталія ЛАЩЕВСЬКА
_____Ім'я, ПРІЗВИЩЕ
«____» _____2024р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Марусяк Владислав Васильович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Створення онлайн-сервісу для аналізу і візуалізації даних з різних джерел

керівник кваліфікаційної роботи Дмитро РОЗМАЇТИЙ,
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024р. № 36

2. Строк подання кваліфікаційної роботи «03» червня 2024р.

3. Вихідні дані до кваліфікаційної роботи: розробка та впровадження програмного забезпечення, обробки та аналізу великих обсягів даних з різних джерел з метою отримання актуальної та корисної інформації

---

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Технології, методології та безпека в сфері збору великих даних
2. Обробка та аналіз великих даних
3. Розробка веб-сервісу

5. Перелік ілюстративного матеріалу: *презентація*
6. Дата видачі завдання «27» лютого 2024р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Збір даних	26.02.2024-22.03.2024	
2	Обробка матеріалу	23.03.2024-05.04.2024	
3	Розробка теоретичної частини	06.04.2024-10.04.2024	
4	Технології та методології в сфері збору великих даних	11.04.2024-30.04.2024	
5	Процеси обробки даних	01.05.2024-03.05.2024	
6	Розробка сервісу збору та аналізу даних	04.05.2024-08.05.2024	
7	Передзахист	14.05.2024-17.05.2024	
8	Здача в деканат	25.05.2024-01.06.2024	

Здобувач(ка) вищої освіти _____ Владислав МАРУСЯК
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Дмитро РОЗМАЇТИЙ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 53 стор., 13 рисю., 2 табл., 27 джерел.

Мета роботи – розробка та впровадження сервісу для ефективного збору та аналізу великих обсягів даних з різних джерел з метою отримання цінної інформації для прийняття управлінських рішень та виявлення нових можливостей

Об'єкт дослідження – процес збору, збереження та аналізу великих обсягів даних з різних джерел, таких як інтернет, датчики, соціальні мережі та інші джерела, що можуть бути важливими для певної діяльності чи бізнесу.

Предмет дослідження – розробка та впровадження програмного забезпечення для автоматизованого збору, обробки та аналізу великих обсягів даних з різних джерел з метою отримання актуальної та корисної інформації.

Короткий зміст роботи: Робота спрямована на розробку сервісу, який забезпечить можливість автоматизованого збору та аналізу великих обсягів даних з різних джерел, включаючи структуровані та неструктуровані дані. Досліджуються методи обробки, збереження та аналізу даних, а також розробляються алгоритми для виявлення закономірностей та трендів у великих масивах інформації. Результатом роботи є створення потужного інструменту для прийняття управлінських рішень та виявлення нових можливостей для розвитку бізнесу на основі аналізу великих даних.

КЛЮЧОВІ СЛОВА: Великі дані, збір даних, аналіз даних, машинне навчання, штучний інтелект, інтеграція даних, системи збору даних, обробка неструктурованих даних, автоматизація аналізу даних, прогностичний аналіз, хмарні технології, візуалізація даних.

ABSTRACT

Text part of the master's qualification work: 53 pages, 13 pictures, 2 table, 27 sources.

The purpose of the work – to develop and implement a service for efficient collection and analysis of large volumes of data from various sources in order to obtain valuable information for managerial decision-making and identifying new opportunities.

Object of research – the process of collecting, storing, and analyzing large volumes of data from various sources such as the internet, sensors, social networks, and other sources that may be important for a particular activity or business.

Subject of research – development and implementation of software for automated collection, processing, and analysis of large volumes of data from various sources in order to obtain relevant and useful information.

Summary of the work: The work aims to develop a service that will provide the capability for automated collection and analysis of large volumes of data from various sources, including structured and unstructured data. Methods of data processing, storage, and analysis are investigated, and algorithms are developed to identify patterns and trends in large information datasets. The result of the work is the creation of a powerful tool for managerial decision-making and identifying new opportunities for business development based on the analysis of big data.

KEYWORDS: Big data, data collection, data analysis, machine learning, artificial intelligence, data integration, data acquisition systems, unstructured data processing, data analysis automation, predictive analysis, cloud technologies, data visualization.

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

ПОДАННЯ
ГОЛОВІ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ
ЩОДО ЗАХИСТУ КВАЛІФІКАЦІЙНОЇ РОБОТИ
на здобуття освітнього ступеня бакалавра

Направляється здобувач Марусяк В.В. до захисту кваліфікаційної роботи
(прізвище та ініціали)
за спеціальністю 123 – Комп'ютерна інженерія
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерна інженерія
(назва)
на тему: «Створення онлайн-сервісу для аналізу і візуалізації даних з різних джерел».

Кваліфікаційна робота і рецензія додаються.

Директор ННІ _____ Андрій БОНДАРЧУК
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Висновок керівника кваліфікаційної роботи

Здобувач Владислав Марусяк показав хорошу теоретичну та практичну підготовку, знання матеріалу, вміння вирішувати самостійно питання і робити висновки. Зміст роботи відповідає завданню. Перелік використаних джерел свідчить про вміння розбиратись в наукових питаннях та застосовувати їх при дослідженнях. Роботу виконував уважно, сумлінно, акуратно.

Все це дозволяє оцінити виконану кваліфікаційну роботу здобувача Владислав Марусяк на оцінку «добре/відмінно» та присвоїти йому(їй) кваліфікацію фахівець з інформаційних технологій.

Керівник кваліфікаційної роботи _____ Дмитро РОЗМАЇТИЙ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

«____» _____20___року

Висновок кафедри про кваліфікаційну роботу

Кваліфікаційна робота розглянута. Здобувач *Марусяк В.В.* допускається до захисту даної роботи в Екзаменаційній комісії.

Завідувач кафедрою КІ _____ Наталія ЛАЩЕВСЬКА
(назва) *(підпис)* *(Ім'я, ПРІЗВИЩЕ)*

ВІДГУК РЕЦЕНЗЕНТА

на кваліфікаційну бакалаврську роботу

здобувача вищої освіти Марусяка Владислава Васильовича
(прізвище, ім'я, по батькові)
на тему «Створення онлайн-сервісу для аналізу і візуалізації даних з різних джерел»

Актуальність.

Робота спрямована на розробку сервісу, який забезпечить можливість автоматизованого збору та аналізу великих обсягів даних з різних джерел, включаючи структуровані та неструктуровані дані. Досліджуються методи обробки, збереження та аналізу даних, а також розробляються алгоритми для виявлення закономірностей та трендів у великих масивах інформації. Результатом роботи є створення потужного інструменту для прийняття управлінських рішень та виявлення нових можливостей для розвитку бізнесу на основі аналізу великих даних.

Позитивні сторони.

1. Зміст кваліфікаційної роботи відповідає завданню. Проект, який виконав Марусяк В.В., показав високий рівень знань і готовність його до майбутньої роботи з фаху
2. Достатньо успішно викладені технічні питання
3. Текст викладений грамотно, ясно, послідовно. Графічний матеріал оформлений якісно. Широко використовується науково-технічна література.

Недоліки.

1. Деякі розділи були занадто короткими або надмірно довгими, що порушило баланс роботи.
2. В деяких розділах наявне повторення інформації.

Відзначені зауваження не впливають на загальну позитивну оцінку кваліфікаційної бакалаврської роботи.

Висновок: *кваліфікаційна бакалаврська робота заслуговує оцінку "_____", а здобувач Владислав Марусяк заслуговує присвоєння кваліфікації: фахівець з інформаційних технологій.*

Рецензент:

науковий ступінь, вчене звання	*підпис*	*Ім'я, ПРІЗВИЩЕ*

ЗМІСТ

ВСТУП……..…….....9
РОЗДІЛ 1 Технології та методології в сфері збору великих даних 12
1.1 Переваги розробки сервісу збору даних……...……..12
1.2 Сучасний стан технологій у сфері збору даних.....16
1.3 Методологічні підходи до аналізу великих даних.....20
1.4 Проблеми безпеки та конфіденційності, етичні аспекти.....25
РОЗДІЛ 2 Процеси обробки даних …….……27
2.1 Процес збору та аналізу великих даних.....27
2.2 Очистка даних.....36
2.3 Інтеграція даних: процес і способи реалізації.....41
РОЗДІЛ 3 Розробка сервісу збору та аналізу даних 43
3.1 Цілі та очікувані функції сервісу збору даних.....42
3.2 Вибір інструментів.....42
3.2.1 Вибір фреймворку.....42
3.2.2 Вибір серверу.....48
3.2.3 Вибір апаратного забезпечення 49
3.3 Розробка збору даних.....50
3.3.1 Збір даних із SQL 52
3.3.2 Збір даних із файлової системи 54
3.3.3 Збір даних із хмарного сховища 57

3.4 Розробка процесу аналізу та візуалізації даних.....…… 57
3.5 Створення веб-ресурсу 65
ВИСНОВКИ…… .. …..... 69
ПЕРЕЛІК ПОСИЛАНЬ……..72
ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація) ……..….80

ВСТУП

Середовище, в якому ми сьогодні живемо, стало надзвичайно цифровим за останні декілька років. Воно надзвичайно швидко розвивається і в ньому термін «дані» став ключовим активом, що стимулює інновації, прийняття рішень і конкурентоспроможність у різних секторах. Експоненціальне зростання обсягу, швидкості та різноманітності даних, отриманих із різних джерел, таких як соціальні медіа, датчики та записи транзакцій, відкрило безмежні можливості для організацій і суспільств. Моя робота має на меті дослідити актуальність збору й аналізу великих даних, які він пропонує, а також, розробити подібний сервіс який реалізує даний функціонал.

Слід почати з того, що збір великих даних надає організаціям безліч цінної інформації про поведінку споживачів, ринкові тенденції та операційну ефективність. Використовуючи величезні обсяги структурованих і неструктурованих даних, компанії можуть отримати повне розуміння своєї цільової аудиторії, уподобань і моделей покупок. Ці знання дають змогу компаніям адаптувати свої продукти та послуги відповідно до мінливих потреб клієнтів, підвищувати задоволеність клієнтів і сприяти зростанню доходів.

Крім того, аналітика великих даних дозволяє організаціям отримувати практичну інформацію зі складних наборів даних, полегшуючи процеси прийняття рішень на основі цих даних. Наприклад, такі гіганти роздрібної торгівлі, як Amazon, використовують аналітику великих даних для персоналізації рекомендацій, оптимізації ланцюгів постачання та прогнозування попиту з надзвичайною точністю, отримуючи таким чином конкурентну перевагу на ринку.

Застосовуючи передові методи аналітики, такі як машинне навчання та прогнозне моделювання, подібні компанії можуть ідентифікувати закономірності, кореляції та тенденції, приховані в даних. Це дозволяє їм приймати обґрунтовані стратегічні рішення, оптимізувати розподіл ресурсів і ефективніше зменшувати ризики.

Незалежно від того, чи йдеться про оптимізацію логістики ланцюга поставок, персоналізацію маркетингових кампаній або виявлення шахрайських дій, уявлення, отримані за допомогою аналітики великих даних, дозволяють організаціям залишатися попереду в сучасному конкурентному середовищі.

Також, актуальність збору та аналізу даних виходить за межі сфери бізнесу, впливаючи на різні аспекти суспільства, включаючи охорону здоров'я, освіту та управління. Наприклад, у сфері охорони здоров'я аналітика великих даних може революціонізувати допомогу пацієнтам, забезпечивши персоналізовані плани лікування, раннє виявлення захворювань і системи підтримки прийняття клінічних рішень. Аналізуючи величезні сховища медичних записів, геномних даних і даних датчиків у реальному часі, постачальники медичних послуг можуть підвищити точність діагностики, покращити результати пацієнтів і зменшити витрати на охорону здоров'я.

Подібним чином у сфері освіти аналітика великих даних має величезний потенціал для оптимізації процесу навчання, виявлення учнів із групи ризику та адаптації навчальних стратегій до індивідуальних потреб. Аналізуючи дані про успішність студентів, показники залученості та моделі навчання, викладачі можуть отримати уявлення про ефективність методів навчання, активно втручатися, щоб підтримати студентів із труднощами, і сприяти більш інклюзивному та персоналізованому навчальному середовищу.

Крім того, уряди та політики можуть використовувати аналітику великих даних для вдосконалення державних послуг, оптимізації міського планування та вирішення соціально-економічних проблем. Аналізуючи дані з різних джерел, таких як записи перепису населення, транспортні системи та канали соціальних мереж, політики можуть отримати уявлення про демографічні тенденції, уподобання громадян і потреби громади. Це дозволяє їм формулювати політику, що ґрунтується на фактах, ефективно розподіляти ресурси та сприяти економічному зростанню та соціальному розвитку.

Однак, незважаючи на те, що актуальність збору та аналізу великих даних є незаперечною, це також викликає значні проблеми етики, конфіденційності та безпеки, які необхідно вирішити. Оскільки організації накопичують величезні обсяги даних про окремих осіб, зростає потреба в забезпеченні прозорості, згоди та заходів захисту даних, щоб захистити права на конфіденційність і зменшити ризик витоку даних і неправомірного використання.

Неможливо переоцінити важливість збору та аналізу великих даних у сучасному світі, який керується даними. Від надання компаніям можливостей приймати обґрунтовані рішення та стимулювання інновацій до революції в охороні здоров’я, освіті та управлінні, аналітика великих даних пропонує трансформаційний потенціал у різних секторах. Проте вкрай важливо знайти баланс між використанням потужності великих даних і вирішенням проблем етики, конфіденційності та безпеки, пов’язаних із їх збором і аналізом. Тільки таким чином ми зможемо повністю використовувати потенціал великих даних для створення більш процвітаючого, інклюзивного та сталого майбутнього для всіх.

ТЕХНОЛОГІЇ ТА МЕТОДОЛОГІЇ В СФЕРІ ЗБОРУ ВЕЛИКИХ ДАНИХ

1.1 **Переваги розробки сервісу збору даних**

Створення сервісу для аналізу та візуалізації великих даних має вирішальне значення в сучасному світі, що керується даними.

Збільшення обсягів даних. Обсяг даних, які генеруються в усьому світі, продовжує зростати через поширення цифрових пристроїв, датчиків Інтернету речей, платформ соціальних мереж тощо. [1] Організаціям потрібні інструменти, щоб зрозуміти цей обсяг даних.

Використання аналітики великих даних забезпечує багато конкурентних переваг. Це допомагає компаніям отримати глибше розуміння, націлити маркетингові зусилля, оптимізувати взаємодію з клієнтами, підвищити ефективність, зменшити витрати та стимулювати інновації продуктів. Прийняття рішень на основі даних є кращим перед традиційними методами, оскільки воно надає об’єктивне розуміння, інформацію в режимі реального часу, прогнозну аналітику та можливості керування ефективністю. Цей підхід дозволяє компаніям приймати обґрунтовані рішення, які швидко реагують на зміни ринку, і передбачати майбутні тенденції, дозволяючи вам щоб відстежувати прогрес у досягненні цілей. Аналіз великих даних може виявити нові можливості, прогалини на ринку та нові тенденції, сприяючи інноваціям і сприяючи розробці нових продуктів, послуг і бізнес-моделей. Це також може допомогти визначити потенційні джерела доходу, які раніше не були враховані.

Аналіз великих даних дає цінну інформацію про тенденції, закономірності та кореляції, надаючи можливість компаніям, урядам і організаціям приймати обґрунтовані рішення на основі даних, а не інтуїції чи припущень. Аналізуючи великі набори даних, організації можуть виявити неефективність, оптимізувати

процеси та оптимізувати розподіл ресурсів, що призведе до підвищення ефективності та продуктивності.

Завдяки аналізу великих даних організації можуть визначити можливості економії коштів, оптимізувати використання ресурсів і мінімізувати втрати. Це може призвести до значного скорочення витрат у різних сферах діяльності, сприяючи підвищенню прибутковості та стійкості. [2]

Кібербезпека. Аналітика великих даних з різноманітних джерел є потужним інструментом для управління ризиками та виявлення потенційного шахрайства, що дозволяє організаціям ефективно виявляти, запобігати та пом'якшувати шахрайство, мінімізуючи фінансові втрати та захищаючи свою репутацію. Використовуючи передові методи аналітики та можливості моніторингу в реальному часі, організації можуть бути на крок попереду шахраїв і захистити свої активи та інтереси.

Аналітика великих даних дозволяє організаціям аналізувати величезні обсяги транзакційних даних, поведінку користувачів та іншу відповідну інформацію, щоб виявити незвичайні шаблони або аномалії, які можуть вказувати на потенційне шахрайство. Встановлюючи базові показники та порівнюючи поточні дані з історичними моделями, алгоритми можуть позначати підозрілу діяльність для подальшого розслідування. Однією з ключових переваг аналітики великих даних у виявленні шахрайства є його здатність забезпечити моніторинг транзакцій і дій у режимі реального часу. [3] Обробляючи потоки даних у режимі реального часу, організації можуть виявляти шахрайську поведінку, щойно вона виникає, що дозволяє їм негайно вживати заходів для запобігання втратам і зменшення ризиків.

Аналітика великих даних дозволяє організаціям проводити аналіз поведінки, щоб зрозуміти типові моделі законної поведінки користувачів. Порівнюючи індивідуальну поведінку з установленими нормами та профілями, організації можуть виявити відхилення, які можуть вказувати на шахрайську діяльність, наприклад незвичайні моделі витрат або спроби доступу з незнайомих місць.

У складних схемах шахрайства, що включають багато учасників і транзакції, методи мережевого аналізу можуть бути використані для виявлення прихованих відносин і зв’язків між суб’єктами. [4] Аналізуючи зв’язки між особами, рахунками та транзакціями, організації можуть виявити організовані мережі шахрайства та вжити відповідних заходів, щоб порушити їхню роботу. Запобігання та пом’якшення шахрайства: крім виявлення, аналітика великих даних також може допомогти організаціям запобігти та пом’якшити шахрайство шляхом впровадження профілактичних заходів, таких як багатофакторна автентифікація, системи виявлення аномалій та динамічні моделі оцінки шахрайства. Ці заходи можуть стримувати шахраїв і ускладнювати їм успішне здійснення своєї діяльності. Відповідність вимогам і нормативним вимогам. Впроваджуючи надійні системи виявлення шахрайства та проводячи ретельний аналіз даних, організації можуть продемонструвати свою відданість дотриманню нормативних стандартів, уникаючи потенційних штрафів і шкоди репутації.

Методи та методології, які використовуються для виявлення шахрайства за допомогою аналітики великих даних, застосовуються в різних галузях, включаючи фінанси, страхування, охорону здоров’я, електронну комерцію та телекомунікації. Кожна галузь стикається з унікальними проблемами шахрайства, і аналітику великих даних можна налаштувати відповідно до конкретних ризиків і сценаріїв шахрайства.

Наукові дослідження та відкриття. Аналіз великих даних прискорює наукові дослідження та відкриття, дозволяючи дослідникам обробляти та аналізувати величезні обсяги даних, отриманих у результаті експериментів, моделювання та спостережень.

Це сприяє міждисциплінарній співпраці, перевірці гіпотез і дослідженню складних явищ, що призводить до проривів у різних галузях. Збір великих обсягів даних дозволяє науковцям виявляти та аналізувати взаємозв'язки та закономірності, які раніше залишалися непоміченими через обмеженість обсягів досліджуваних даних.

Наприклад, у медичних дослідженнях аналіз великих медичних баз даних може допомогти виявити нові причинно-наслідкові зв'язки між факторами ризику та захворюваннями. [5]

Збір великих обсягів даних дозволяє створювати та вдосконалювати моделі прогнозування. Завдяки цьому науковці можуть передбачити розвиток подій у різних сферах: від кліматичних змін до ринкових тенденцій. Наприклад, використання данних з супутників та датчиків дозволяє точніше прогнозувати природні катастрофи.

Також, збір великих обсягів даних робить можливим виявлення нових знань та трендів у різних галузях. Аналізуючи великі масиви даних, науковці можуть виявити нові закономірності або тенденції, що стають основою для подальших досліджень. Наприклад, аналіз даних з соціальних мереж дозволяє виявляти нові тренди у поведінці споживачів та суспільних процесах.

Аналітика великих даних також може стимулювати інновації, відкриваючи нові можливості, виявляючи нові тенденції та висвітлюючи сфери, де продукти чи послуги потребують вдосконалення.

Розширення клієнтських послуг у некомерційних сферах. Аналітика великих даних відіграє вирішальну роль у сфері охорони здоров’я та державних послуг, сприяючи прогнозному моделюванню, нагляду за захворюваннями, розподілу ресурсів та оптимізації догляду за пацієнтами. [6] Це дає змогу постачальникам медичних послуг і державним установам надавати кращі послуги, покращувати результати та посилювати ініціативи в галузі охорони здоров’я.

Статистика на основі даних для розробки політики. Уряди та політики можуть використовувати аналітику великих даних, щоб приймати більш обґрунтовані рішення, розробляти політику, що ґрунтується на фактах, і ефективніше вирішувати суспільні проблеми.

Аналізуючи великі набори даних, пов’язані з демографічними показниками, економікою та охороною здоров’я, політики можуть сформулювати ефективні та стійкі стратегії.

Аналітика великих даних надає клієнтам персоналізований досвід. Незалежно від того, чи йдеться про рекомендації продукту, персоналізовані маркетингові повідомлення чи спрощені послуги, аналітика даних дозволяє компаніям краще задовольняти потреби окремих клієнтів. Також, аналітика великих даних дозволяє організаціям ефективно виявляти та зменшувати ризики. Незалежно від того, чи йдеться про виявлення шахрайських дій, прогнозування збоїв обладнання чи аналіз ринкових ризиків, аналітика даних може допомогти організаціям завчасно керувати ризиками

1.2 **Сучасний стан технологій у сфері збору даних**

Сучасний стан технологій в сфері збору і аналізу великих даних відіграють ключову роль у спрощенні та поліпшенні процесів обробки та використання даних. За останній час відбулися наступні основні напрямки технологічних змін:

*Хмарні обчислення***.** Платформи хмарних обчислень, такі як Amazon Web Services (AWS), Microsoft Azure, та Google Cloud Platform, надають потужні інфраструктурні можливості для зберігання, обробки та аналізу великих обсягів даних. [7] Вони дозволяють компаніям та дослідникам легко масштабувати ресурси в залежності від потреб проекту.

Хмарні обчислення використовують розподілені ресурси, що доступні через Інтернет, для обробки та зберігання даних. Вони дозволяють користувачам отримувати доступ до обчислювальної потужності без необхідності володіти та підтримувати власні сервери та обладнання. Хмарні обчислення забезпечують масштабованість, гнучкість та доступність ресурсів для обробки великих обсягів даних. Вони дозволяють швидко масштабувати обчислювальні потужності залежно від потреб, що особливо важливо при обробці великих даних.

Технології розподілених систем, такі як Apache Hadoop та Apache Spark, також дозволяють ефективно обробляти великі обсяги даних на розподілених кластерах серверів. [8] Ці системи забезпечують високу швидкість обробки та відмінну масштабованість. Розподілені системи - це системи, які складаються з групи взаємопов'язаних комп'ютерів, які працюють разом для виконання обчислювальних завдань. Вони часто використовують паралельне обчислення для швидкого та ефективного аналізу великих обсягів даних.

Розподілені системи дозволяють розподілити завдання обробки великих даних між багатьма комп'ютерами, що прискорює час обробки та забезпечує високу доступність.

Ці технології надають потужні інструменти для розподіленої обробки та аналізу великих обсягів даних. Вони дозволяють працювати з різними типами даних та використовувати різноманітні методи аналізу, включаючи машинне навчання та глибинне навчання.

Таблиця 1.1 – Порівняльна таблиця інструментів для обробки потокових даних

Характеристика	**Apache Hadoop**	**Apache Spark**	**Apache Flink**
Мова програмування	Java, Python, інші	Scala, Java, Python, R, SQL	Java, Scala, Python
Тип обробки даних	Batch processing	Batch, Streaming, Interactive	Batch, Streaming
Продуктивність	Низька до середньої (залежно від конфігурації кластера)	Висока (швидше ніж Hadoop, особливо для ітеративних алгоритмів)	Висока (рівень латентності мінімальний, оптимізований для стрімінгу)
Інтеграція з екосистемою Hadoop	Широка, добре інтегрована	Підтримка Hadoop через HDFS та YARN	Є підтримка, але не така широка як у Hadoop та Spark

Продовження таблиці 1.1 – Порівняльна таблиця інструментів для обробки потокових даних

Масштабованість	Добре масштабований, але вимагає більшої кількості ресурсів	Добре масштабується, може працювати з великими обсягами даних	Добре масштабується, ефективно використовує ресурси
Оптимізація	Менш ефективна оптимізація через MapReduce	Широкий спектр оптимізацій, включаючи інший двигун обробки даних	Високопродуктивна оптимізація, підтримка декларативного програмування
Підтримка стрімінгу	Немає вбудованої підтримки стрімінгу	Є підтримка стрімінгу через Spark Streaming та Structured Streaming	Офіційна підтримка стрімінгу, призначена для цього

Розвиток інструментів для машинного навчання та глибинного навчання, таких як TensorFlow, PyTorch та Scikit-learn, дозволяє виявляти складні залежності та патерни в великих наборах даних. [9]

Машинне навчання та інтерактивні візуалізації даних є корисними інструментами в аналізі великих обсягів даних з різних причин. Машинне навчання дозволяє автоматизувати процес аналізу та виявлення складних патернів у великих даних, що може бути важко або навіть неможливо здійснити вручну. Відкриті бібліотеки, такі як TensorFlow та PyTorch, надають інструменти для розробки, тренування та оцінки моделей машинного навчання без значних зусиль з програмування.

Ось кілька основних інструментів у області машинного навчання:

TensorFlow. Розроблений компанією Google, TensorFlow є одним з найпопулярніших відкритих фреймворків для машинного навчання. Він надає широкий спектр інструментів для створення та тренування різних моделей машинного навчання.

PyTorch. Це ще один популярний відкритий фреймворк для машинного навчання, який надає зручний інтерфейс та динамічний граф обчислень, що робить розробку моделей швидше та простіше.

Scikit-learn. Це простий та ефективний інструментарій для машинного навчання в Python. Scikit-learn містить реалізації багатьох класичних алгоритмів машинного навчання та інструменти для підготовки даних та оцінки моделей.

Ці інструменти широко використовуються в області аналізу великих даних для розв'язання різноманітних завдань, від прогнозування до класифікації та кластеризації.

Також важливими інструменти у сфері збору та аналізу даних є *інтерактивні візуалізаційні інструменти*, такі як Tableau, Power BI, та Plotly, дозволяють користувачам взаємодіяти з великими обсягами даних шляхом побудови графіків, діаграм та інших візуальних елементів. [10]

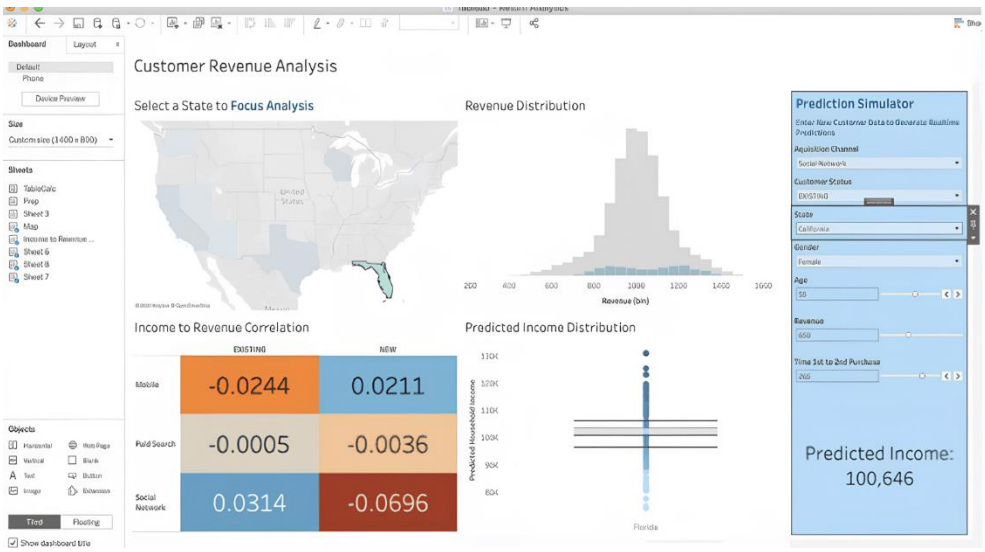


Рисунок 1.1 – Використання інструменту Tableau для візуалізації даних клієнтського прибутку

1.3 **Методологічні підходи до аналізу великих даних**

Методологічні підходи до аналізу великих даних розвиваються швидко, щоб відповісти на складні виклики, пов'язані з обробкою та інтерпретацією великих обсягів інформації. Методологічні підходи дозволяють систематизувати процес збору та аналізу даних, що робить його більш ефективним та продуктивним. Використання стандартизованих методів дозволяє знизити вплив суб'єктивних факторів на результати дослідження. Методологічні підходи, такі як машинне навчання та глибинне навчання, відкривають нові можливості для аналізу даних та виявлення раніше невидимих залежностей та патернів.

Деякі методології дозволяють автоматизувати процеси збору, обробки та аналізу даних, що робить їх більш швидкими та менш витратними з точки зору ресурсів. Вони також допомагають організаціям розробляти стратегії збору та використання даних, що підтримують досягнення їхніх цілей та завдань.

Проте, методологічні підходи слід використовувати з обачливістю та урахуванням їхніх недоліків. Деякі методології можуть бути обмежені застосуванням лише до певного типу даних або визначених умов. Також, деякі методології можуть бути складними у реалізації та вимагати спеціалізованих знань та навичок для їхнього використання. Деякі методи аналізу даних можуть бути чутливими до якості та обсягу даних, що вимагає ретельного планування та збору даних. Багато методологій потребують попередньої обробки даних, що може бути часо- та ресурсозатратним процесом. У деяких випадках результати аналізу можуть бути неоднозначними або потребувати інтерпретації, що може призвести до неправильних висновків.

Самі важливі методології збору даних на сьогодні є машинне навчання (Machine Learning), глибинне навчання (Deep Learning), аналіз тексту (Text

Analysis), графові аналітика (Graph Analytics), аналіз вимірюваних даних (Sensor Data Analysis).

Машинне навчання - це підмножина штучного інтелекту, яка досліджує розробку алгоритмів, які можуть навчатися та вдосконалювати свої здатності на основі досвіду, тобто даних. Основна ідея машинного навчання полягає в тому, щоб створити моделі, які можуть робити прогнози або приймати рішення, використовуючи великий обсяг даних без явного програмування.

До основних методів машинного навчання відносяться:

– *Навчання з учителем (Supervised Learning)*. Моделі навчаються на позначених (мітках) даних, де кожен вхід має відповідний вихід. Цей тип навчання використовується для задач, таких як класифікація та регресія.

– *Навчання без учителя (Unsupervised Learning)*. Моделі навчаються на непозначених даних, зазвичай для виявлення прихованих структур або патернів у даних. До прикладів задач без учителя відносять кластеризацію та зниження розмірності.

– *Укріплене навчання (Reinforcement Learning)*. Моделі навчаються приймати послідовні рішення шляхом взаємодії з динамічним середовищем та отриманням винагороди чи покарання відповідно до їхніх дій. Цей підхід часто використовується у вирішенні задач управління та навчання агентів.

Машинне навчання використовується в різних областях, від комп'ютерного зору та обробки природної мови до медичної діагностики та фінансового аналізу. Це могутній інструмент, який дозволяє автоматизувати прийняття рішень та виявляти складні залежності у великих обсягах даних.

У машинному навчанні *дані* є основним ресурсом. Це може бути будь-яка інформація, яка характеризує об'єкти чи явища, які ми хочемо вивчати. Дані можуть бути у формі таблиць, текстових документів, зображень, аудіофайлів тощо. Також важливе поняття для машинного навчання є *ознаки*. Ознаки - це конкретні атрибути або характеристики об'єктів у навчальному наборі даних, які використовуються для прогнозування чи класифікації. Наприклад, якщо ми

аналізуємо зображення, ознаками можуть бути пікселі зображення, або властивості звуку, якщо ми аналізуємо аудіо.

Щоб машинне навчання могло аналізувати дані, необхіним є *навчальний набір* (Training Dataset): Це підмножина даних, яка використовується для навчання моделі. Вона містить пари вхідних ознак та відповідних цільових значень, які модель намагається вивчити.

Щоб зрозуміти, наскільки прогнози моделі відрізняються від фактичних значень цільової змінної, у машиному навчанні використовується *функція втрат* (Loss Function): Це функція, яка вимірює помилку моделі, порівнюючи її прогнозовані значення з реальними. Мета моделювання полягає в мінімізації цієї функції, щоб зробити прогнози якомога більш точними.

Машинне навчання також використовує *оптимізацію*. Це процес пошуку оптимальних параметрів моделі, які мінімізують функцію втрат. Оптимізація включає в себе такі методи, як стохастичний градієнтний спуск, методи адама тощо.

Після навчання моделі важливо перевірити її ефективність на даних, яких вона ще не бачила. Для цього використовуються окремі набори даних для валідації та тестування. [13]

Глибинне навчання – це підмножина машинного навчання, яка використовує нейронні мережі з багатьма шарами для виявлення складних патернів та вирішення завдань, які потребують великої обробки даних. Основна відмінність глибинного навчання від інших методів машинного навчання полягає у його здатності автоматично відбирати корисні ознаки або риси з даних, не потребуючи ручного втручання.

До основних архітектур глибинного навчання відносяться:

– *Згорткові нейронні мережі (Convolutional Neural Networks, CNNs)*. Ці мережі ефективно працюють з вимірювальними даними, такими як зображення або відео, виявляючи в них просторові залежності.

– *Рекурентні нейронні мережі (Recurrent Neural Networks, RNNs).* Ці мережі добре справляються з послідовними даними, такими як мовлення або текст, враховуючи контекст та залежності між елементами послідовності.

– *Варіації нейронних мереж (Variational Autoencoders, Generative Adversarial Networks та інші)*. Ці архітектури використовуються для генерації нових даних або відтворення вхідних даних, що робить їх корисними для генерації нових зображень, текстів тощо.

Глибинне навчання зазвичай вимагає великої кількості даних та обчислювальних ресурсів, але воно дозволяє досягати вражаючих результатів у багатьох областях, таких як комп'ютерне зору, обробка природної мови, медична діагностика та багато інших.

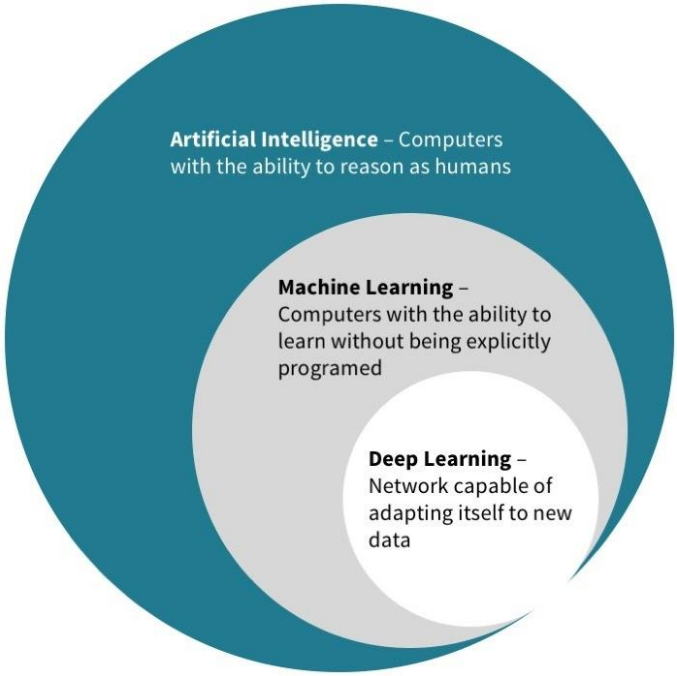


Рисунок 1.2 – Порівняння машинного навчання та глибинного навчання

Проблеми безпеки та конфіденційності, етичні аспекти

Проблеми безпеки та конфіденційності у сфері збору та аналізу великих даних є серйозними та постійно еволюціонують, оскільки обсяги даних постійно зростають, а також з'являються нові загрози та вразливості. Однією з основних проблем є забезпечення захисту особистої інформації та даних користувачів, які можуть бути зібрані та оброблені великими даними.

Однією з головних загроз є можливість несанкціонованого доступу до великих обсягів даних. Це може стати на шляху хакерів, кіберзлочинців, а також навіть державних акторів, які можуть використовувати зібрані дані для кібершпигунства або кібератак. Такий доступ може призвести до витоку конфіденційної інформації, порушення приватності користувачів або навіть фінансових втрат.

Крім того, існує загроза внутрішнього порушення безпеки даних, коли самі працівники організацій, які мають доступ до великих обсягів даних, можуть використовувати цей доступ для власної користі або для шкоди компанії. Це може бути спричинено недостатньою контрольованістю доступу до даних або недостатніми процедурами перевірки персоналу.

Додатково, існує ризик зловживання даними з метою маніпулювання або впливу на людей або групи. Це може включати в себе розповсюдження фейкових новин або маніпуляцію результатами досліджень на основі аналізу даних. Така маніпуляція може мати серйозні наслідки для суспільства та демократичних процесів.

Ще однією проблемою є збереження конфіденційності даних. За допомогою великих даних можна створити досить точні профілі користувачів на основі їхньої активності в мережі, покупок, медичної інформації та інших даних. Це відкриває шлях для можливого зловживання цією інформацією, такого як направлене рекламування, дискримінація, або навіть зловживання владою.

Крім того, існує ризик порушення конфіденційності через недоліки у захисті даних, помилкової анонімізації даних та можливості деанонімізації, що може призвести до розкриття особистих даних.

Ще однією проблемою є відсутність стандартів безпеки та конфіденційності для обробки великих даних. У багатьох випадках, підходи до захисту даних є розрізненими та неуніфікованими, що може створювати ризики для даних, особливо при їх обміні між різними організаціями. Недостатній контроль над доступом до даних, недостатнє шифрування та інші проблеми можуть призвести до витоку конфіденційної інформації.

Еволюція технологій також створює нові проблеми безпеки та конфіденційності. Наприклад, розвиток штучного інтелекту та автоматизованих систем може вести до виникнення нових методів атак та зловживань, таких як атаки через зловживання моделями машинного навчання або зловживання штучною генерацією контенту для маніпулювання даними.

Також, одним із найбільш важливих питань є приватність даних. Зі збільшенням обсягу зібраних даних і розвитком технологій, які дозволяють аналізувати ці дані, наростає ризик порушення приватності. Підвищується ймовірність недозволеного доступу до особистої інформації, що може призвести до негативних наслідків для особистого життя та свободи громадян.

Третій етичний аспект стосується відповідального використання даних. Підприємства та установи, що збирають великі обсяги даних, мають бути відповідальними за їхнє використання. Це означає розумне та етичне використання даних, а також захист прав та інтересів осіб, які є джерелом цих даних. [16]

Крім того, важливим етичним аспектом є прозорість та відкритість у зборі та використанні даних. Людям має бути надана можливість зрозуміти, як їхні дані збираються, обробляються та використовуються. Це вимагає відповідних правил і політик, а також механізмів контролю за їх виконанням.

2 ПРОЦЕСИ ОБРОБКИ ДАНИХ

2.1 Процес збору та аналізу великих даних.

У нашу цифрову епоху поширення інформації досягло найвищого рівня, створюючи як можливості, так і виклики для різних галузей. Від роздрібної торгівлі до охорони здоров’я, від фінансів до розваг, велика кількість даних представляє неймовірну кількість можливостей. Але з великим обсягом виникає також великі труднощі. Навігація в цьому "морі" даних вимагає не лише складних методів збору, але й потужних аналітичних інструментів для визначення значущих закономірностей і тенденцій.

Збір даних є одним із ключових етапів аналізу та обробки інформації з різних джерел. Цей процес може бути виконаний як автоматизовано, за допомогою спеціалізованих програмних засобів, так і вручну, коли дані збираються та введені людиною. [17]

Автоматизований збір даних передбачає використання програмних засобів для автоматичного збору, обробки та інтеграції інформації з різних джерел. Основні переваги автоматизованого збору даних включають:

- *Ефективність*. Програмні засоби можуть автоматично збирати великі обсяги даних з різних джерел швидко та ефективно.
- *Точність*. Автоматизовані процеси мають меншу ймовірність помилок порівняно з ручним введенням даних, що сприяє підвищенню точності результатів.
- *Скорочення часу*. Автоматизований збір даних дозволяє значно скоротити час, потрібний для отримання необхідної інформації.

Проте, автоматизований збір даних також може мати свої обмеження та викликати деякі проблеми, такі як складність налаштування програмних засобів,

можливість отримання неповної або неточної інформації через технічні помилки або обмеження доступу до деяких джерел даних через захищений доступ.

Щоб реалізувати механізм автоматичного збору даних необхідно задіяти спеціалізовані програмні скрипти або використати програмні інтерфейси (API) для доступу до даних. Наприклад, для отримання даних з веб-сайтів можуть використовуватися бібліотеки для парсингу HTML або API для доступу до вмісту сайту.

Отримані дані зазвичай потребують обробки перед тим, як їх можна буде використовувати для аналізу. Це може включати видалення непотрібних або дубльованих записів, перетворення форматів даних, корекцію помилок та інші маніпуляції. Для цього також можуть використовуватися програмні засоби, які дозволяють автоматизувати цей процес.

Після обробки дані з різних джерел потрібно об'єднати в один інтегрований набір даних. Цей процес, відомий як інтеграція даних, включає вирівнювання схем даних, вирішення конфліктів та об'єднання даних з різних джерел в один набір даних. Це може вимагати використання спеціалізованих інструментів для обробки та інтеграції даних.

Останнім кроком є збереження інтегрованого набору даних для подальшого використання. Дані можуть бути збережені у базі даних, файловій системі або хмарному сховищі в залежності від вимог проєкту та доступності ресурсів.

У випадках, коли автоматизований збір даних неможливий або неефективний, використовуються методи збору даних вручну. *Збір даних вручну* - це процес, при якому людина вручну збирає, аналізує та вводить інформацію з різних джерел без значної автоматизації.

Після ідентифікації джерел даних необхідно здійснити збір інформації вручну. Це може включати перегляд веб-сайтів, ручне копіювання даних з документів, проведення інтерв'ю або опитування людей тощо. Особливо важливою є уважність та точність при зборі даних вручну, оскільки будь-яка помилка може призвести до неточних результатів.

Після збору даних вони потребують обробки перед тим, як їх можна буде використати для подальшого аналізу. Це може включати видалення дублікатів, корекцію помилок, перетворення форматів даних та інші маніпуляції. Обробка даних вручну може бути часо- та ресурсоємкою, а також вимагати спеціалізованих навичок та знань.

У випадку збору даних вручну, інтеграція даних може бути складнішою, оскільки дані можуть бути отримані з різних джерел і мати різний формат. Цей процес може вимагати вручного злиття та вирівнювання даних, а також вирішення конфліктів та невідповідностей.

Основна різниця між автоматизованим та вручним збором даних полягає у рівні автоматизації та втручання людини в процес. В автоматизованому зборі даних використовуються програмні засоби для автоматичного збору, обробки та інтеграції інформації, тоді як вручний збір даних вимагає активної участі людини на кожному етапі процесу. Автоматизований збір даних зазвичай є швидшим, ефективнішим та менш помилковим, але він може бути обмежений у випадках, коли доступ до даних обмежений або коли потрібно здійснювати складні маніпуляції з даними. Вручний збір даних набагато гнучкіший та може бути використаний у випадках, коли автоматизований збір неможливий або неефективний, але він може бути часо- та ресурсоємким, а також менш точним через можливість людських помилок.

Джерела збору інформації можуть бути дуже різноманітними і залежать від конкретного контексту, області аналізу та потреб дослідження. Типові джерела збору інформації включають:

Пристрої IoT. Інтернет речей (IoT, Internet of Things) – це концепція мережі, яка складається з фізичних пристроїв, що співпрацюють між собою та обладнані датчиками, а також програмного забезпечення, яке автоматично обмінює даними між фізичним середовищем та комп'ютерними системами за допомогою стандартних протоколів зв'язку. Окрім датчиків, у мережі можуть бути виконавчі пристрої, які вбудовані в фізичні об'єкти та з'єднані між собою через дротові або

бездротові з'єднання. Ці пристрої збирають і передають дані з різних джерел, таких як датчики, інтелектуальні пристрої, переносні пристрої тощо.

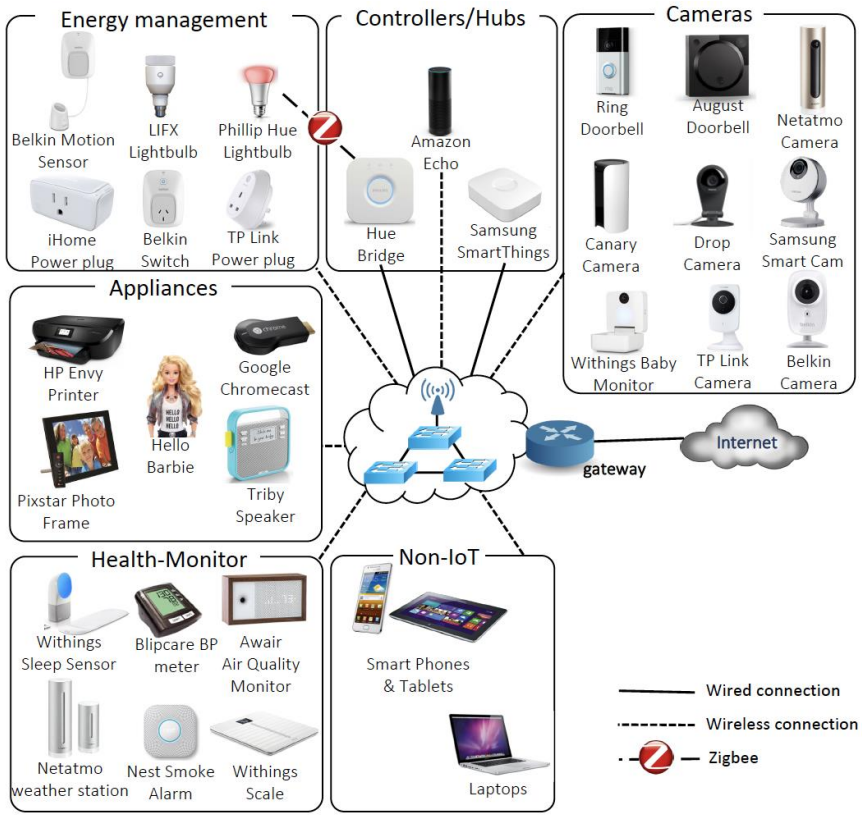


Рисунок 2.1 – Схема роботи концепції Інтернету речей

Загалом, Інтернет речей дозволяє отримувати величезний обсяг даних в реальному часі, що допомагає підприємствам та організаціям приймати обґрунтовані рішення на основі аналізу цих даних. [19]

Соціальні медіа. Платформи, такі як Facebook, Twitter і Instagram, щодня генерують мільйони постів, коментарів, лайків і ретвітів. Кожна взаємодія користувачів створює величезний обсяг даних. Ці дані можуть включати текстові повідомлення, фотографії, відео, місцезнаходження, дати та часи публікацій, дані про користувачів (вік, стать, інтереси) тощо.

Аналіз даних з соціальних медіа дозволяє розуміти тренди, настрої громадської думки, виявляти впливових користувачів, прогнозувати події та взаємодіювати з аудиторією.

Веб-дані. Інформація збирається з веб-сайтів за допомогою різних інструментів, таких як веб-аналітика, cookie-файлів, API тощо. Дані можуть

використовуватися для аналізу потоку відвідувань, визначення ефективності маркетингових кампаній, розуміння поведінки користувачів на сайті, виявлення трендів і змін у попиті тощо. Аналіз веб-даних допомагає підприємствам оптимізувати свої веб-ресурси, привертати більше трафіку, підвищувати конверсію та забезпечувати кращий користувацький досвід.

Дані про транзакції. Ці дані можуть бути отримані з різних джерел, таких як платіжні системи, банківські операції, онлайн-магазини, торгові платформи тощо. Інформація про суми транзакцій, час та місце здійснення, категорії товарів або послуг, інформація про клієнтів тощо. Аналіз цих даних дозволяє підприємствам розуміти покупкові звички клієнтів, прогнозувати попит на товари і послуги, виявляти шаблони шахрайства та підвищувати ефективність маркетингових стратегій.

Усі ці типи даних допомагають підприємствам розуміти своїх клієнтів краще, приймати обгрунтовані рішення та покращувати свою діяльність. Аналіз великих даних стає невід'ємною частиною стратегій розвитку бізнесу в епоху цифрової трансформації.

Дані машин і датчиків. Машини та датчики в таких галузях, як виробництво, охорона здоров’я та транспорт, генерують дані про продуктивність, ефективність і умови навколишнього середовища. [20]

Хмарне сховище. Багато організацій зберігають великі дані в хмарі за допомогою таких платформ, як Amazon Web Services (AWS), Google Cloud Platform (GCP) або Microsoft Azure. Ці платформи пропонують масштабовані рішення для зберігання даних, такі як Amazon S3, Google Cloud Storage і Azure Blob Storage.

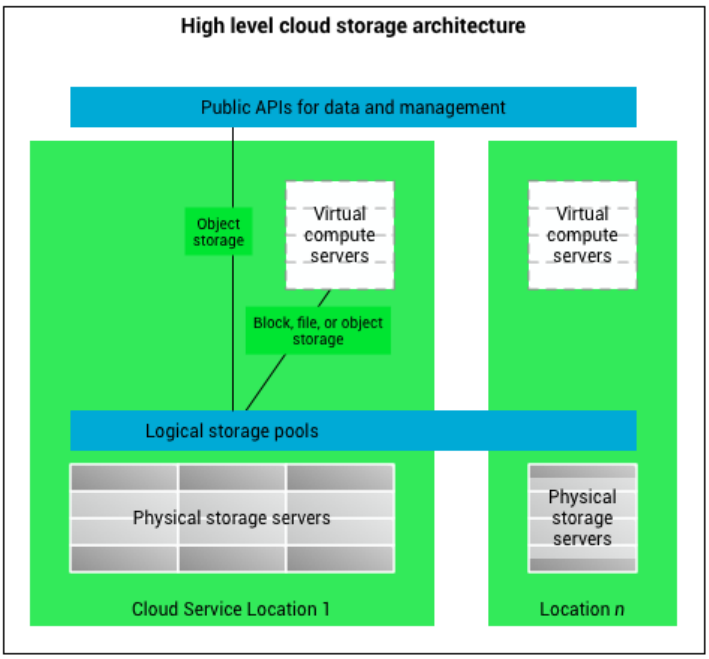


Рисунок 2.2 – Високорівнева архітектура хмарного сховища.

Такі платформи зазвичай мають великі обсяги сховищ, які можуть масштабуватися згідно з потребами користувача. Такі послуги можуть бути використані для зберігання різних типів даних, від текстових документів до мультимедійного вмісту.

Хмарні сховища можуть використовуватися для збору даних наступним чином:

– Хмарні сховища надають можливість зберігати великі обсяги даних без необхідності інвестування в власне обладнання. Ви можете завантажити дані у хмарне сховище та мати доступ до них з будь-якого пристрою з підключенням до Інтернету.
– Хмарні сховища забезпечують автоматичне резервне копіювання даних, що дозволяє захистити ваші дані від втрати. Це особливо важливо для великих обсягів даних, які можуть бути складно зберігати локально.
– Деякі хмарні платформи надають інструменти для аналізу та обробки великих даних безпосередньо в хмарі. Це дозволяє вам виконувати складні операції обробки даних без необхідності завантажувати їх на локальний комп'ютер.

- Хмарні сховища дозволяють кільком користувачам одночасно працювати з даними, спільно редагувати документи та обмінюватися інформацією.
- Деякі хмарні платформи мають API, які дозволяють інтегрувати їх з іншими сервісами та програмним забезпеченням для автоматизації процесів збору та обробки даних.

Озера даних. Озера даних (data lakes) — це сховища, які зберігають необроблені дані у рідному форматі, доки вони не знадобляться. Це форма архітектури зберігання даних, яка дозволяє збирати, зберігати, обробляти і аналізувати великі обсяги даних з різних джерел. Озера даних зазвичай використовуються для роботи з великими обсягами структурованих, напівструктурованих і неструктурованих даних.

Основна ідея озера даних полягає в тому, щоб зберігати всі дані без обробки або моделювання їх заздалегідь. Це відрізняється від традиційних методів зберігання даних, де дані зазвичай структуруються перед зберіганням.

У результаті озеро даних забезпечує гнучкість для великого різноманіття використання даних, включаючи аналітику, машинне навчання, штучний інтелект тощо.

Однак, без належної уваги до управління якістю даних, озера даних можуть стати місцем, де важко знайти потрібні дані, а також можуть виникнути проблеми з безпекою та конфіденційністю. Тому важливо мати ефективні стратегії інтеграції, управління та захисту даних в озерах даних.

У порівнянні з традиційними базами даних, де структура даних фіксована, озеро даних дозволяє зберігати дані у будь-якому форматі: текстовому, графічному, аудіо, відео тощо.

Озера даних зазвичай використовуються для аналізу великих обсягів даних та для використання їх у машинному навчанні, штучному інтелекті та інших аналітичних задачах. Вони дозволяють зберігати дані у їх природній формі та обробляти їх під час потреби, що дозволяє здійснювати більш гнучкий та ефективний аналіз і використання інформації. [20

Розподілені файлові системи - це системи зберігання даних, які розподілені по різних вузлах мережі. Вони дозволяють працювати з великими обсягами даних, розподіляючи їх зберігання та обробку між багатьма серверами. Це полегшує масштабування систем та забезпечує високу доступність даних.

Для використання розподілених файлових систем для збору великих даних, потрібно налаштувати мережеві вузли таким чином, щоб вони могли обмінюватися даними та координувати свою роботу. Це може включати налаштування механізмів реплікації даних для забезпечення надійності, а також використання алгоритмів розподіленого зберігання для ефективного розподілу даних між вузлами.

Після налаштування розподіленої файлової системи можна розпочати збір великих даних, завантажуючи їх на доступні вузли мережі та оброблюючи їх паралельно. Важливо розробити ефективні алгоритми обробки даних, які можуть працювати з розподіленими ресурсами та забезпечити високу продуктивність при роботі з великими обсягами даних.

Деякі з технологій розподілених файлових систем включають:

- Hadoop Distributed File System (HDFS): Розподілена файлова система, яка використовується в екосистемі Apache Hadoop для зберігання великих обсягів даних на кластерах серверів.
- Google File System (GFS): Розподілена файлова система, розроблена Google для зберігання та обробки великих обсягів даних на їхній інфраструктурі.
- Apache Cassandra: Розподілена база даних, яка також може діяти як файлова система, призначена для зберігання великих обсягів даних з високою доступністю та масштабованістю.
- GlusterFS: Розподілена файлова система, яка дозволяє об'єднувати різні сховища даних в одну єдину файлову систему, що дозволяє масштабувати обсяги даних.
- Lustre: Розподілена файлова система, спеціально розроблена для високопродуктивних обчислювальних кластерів, таких як суперкомп'ютери. [21]

Бази даних. База даних - це організована колекція даних, яка зберігається та може бути доступна для обробки та аналізу. Вона складається з таблиць, які містять записи (рядки) з конкретною інформацією, розділеною на різні стовпці (поля). [22]

Використання баз даних для збору та аналізу інформації може бути дуже корисним з кількох причин:

- Централізоване зберігання даних: База даних дозволяє зберігати всю інформацію в одному місці, що робить доступ до неї легшим і зручнішим.
- Швидкий доступ до інформації: Завдяки оптимізованій структурі та запитам до бази даних, можна швидко знаходити та отримувати необхідну інформацію.
- Автоматизація процесів: За допомогою баз даних можна автоматизувати багато рутинних операцій, що дозволяє зменшити кількість помилок та оптимізувати час.
- Аналіз даних: Бази даних надають зручну можливість аналізу великих обсягів інформації, що дозволяє отримувати цінні висновки та прогнози, корисні для прийняття рішень.
- Забезпечення безпеки даних: Бази даних можуть мати вбудовані механізми захисту, які дозволяють контролювати доступ до інформації та забезпечити конфіденційність даних.

2.2 Очистка даних

Очистка даних - це важливий етап обробки інформації, на якому проводиться видалення аномалій, виявлення та коригування помилок, а також конвертація даних у відповідний формат. [23]

Цей процес має на меті підготувати дані для подальшого аналізу та використання, забезпечити їхню точність та консистентність. Очищення даних зазвичай ділиться на такі етапи:

Виявлення аномалій. Перший крок у процесі очищення даних - це виявлення аномалій та некоректних даних. Це можуть бути помилки введення, відсутність даних (пропуски), некоректні значення або викиди (outliers). Для цього можна використовувати статистичні методи аналізу даних або ручну перевірку.

Одним із способів виявлення аномалій є *візуальний аналіз.* Початковий візуальний огляд даних може допомогти виявити очевидні помилки, такі як відсутність даних (пропуски), некоректні значення або викиди (outliers). Графіки, діаграми та інші візуальні засоби можуть допомогти ідентифікувати аномалії та виправити їх.

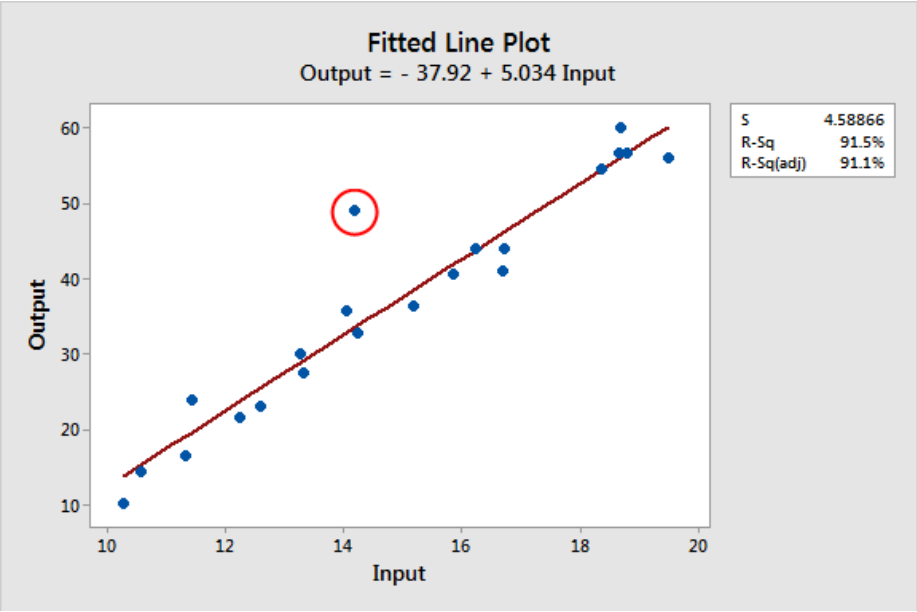


Рисунок 2.3 – Викид (outlier) в графіку даних

Додатково, використання статистичних методів може допомогти виявити відхилення від типового розподілу даних, аномалії та інші некоректності. До цього входять аналіз змінних, виявлення викидів за допомогою стандартних відхилень, квартилей та інших метрик. Для числових даних можна перевірити їх на відповідність певному діапазону значень. Це допоможе виявити аномальні або некоректні значення, які виходять за межі прийнятних значень.

Одним із ефективних способів виявлення помилок під час аналізу великих даних є порівняння даних з іншими джерелами, це може допомогти виявити невідповідності та помилки.

Якщо дані з одного джерела суттєво відрізняються від даних з іншого джерела, це може бути ознакою проблеми або помилки в одному з джерел. Також, використання правил та обмежень для даних може допомогти виявити аномалії. Наприклад, якщо ви очікуєте, що певна змінна має бути цілим числом або діапазон її значень обмежений, будь-яке відхилення від цих правил може бути ознакою помилки. Алгоритми машинного навчання також можуть бути використані для автоматичного виявлення аномалій та викидів в даних. Ці алгоритми можуть виявити складніші аномалії, які можуть бути важко виявити за допомогою традиційних методів.

Після виявлення аномалій необхідно здійснити їхнє *коригування*.

Коригування помилок - це процес виправлення некоректних, неправильних або недостовірних значень або структури даних для забезпечення їхньої точності та надійності. Цей етап є важливою частиною обробки даних і передбачає вжиття відповідних заходів для виправлення виявлених помилок. Процес коригування помилок може бути реалізованим наступними методами:

а) *видалення аномальних значень*. Першим кроком у коригуванні помилок може бути видалення аномальних значень, які виходять за межі прийнятного діапазону або не відповідають очікуваному формату даних. Це може бути зроблено шляхом простого видалення записів або заміною аномальних значень на значення за замовчуванням.

б) *виправлення помилок введення*. Якщо помилки в даних виникли через некоректне введення, наприклад, помилки при наборі або неправильність формату, то їх можна виправити вручну або за допомогою автоматизованих методів. Наприклад, можна використовувати правила перевірки даних або алгоритми автоматичної обробки тексту для виправлення помилок.

в) *заповнення пропусків*. Якщо в даних є пропуски (порожні значення), їх можна заповнити шляхом заміни їх середніми значеннями, медіанами або іншими показниками цього рядка даних. Це допомагає зберегти структуру даних та уникнути втрати інформації.

г) *використання контекстуальних правил*. Для виправлення деяких помилок можна використовувати контекстуальні правила або залежності між даними. Наприклад, якщо ви знаєте, що значення одного параметра повинно бути відповідно до іншого параметра, то можна використати це правило для виправлення некоректних даних.

д) *використання алгоритмів машинного навчання*. У деяких випадках для виявлення та коригування помилок можуть бути використані алгоритми машинного навчання. Наприклад, можна навчити модель на основі існуючих даних та використовувати її для прогнозування або виправлення невірних значень.

е) *відстеження та аудит змін*. Після внесення змін до даних важливо вести відстеження та аудит змін, щоб мати можливість повернутися до попередніх версій даних у разі необхідності. [24]

Після виправлення помилок може знадобитися *конвертація даних* у відповідний формат. Це може включати перетворення типів даних (наприклад, з текстового у числовий формат), перекодування даних (наприклад, зміна кодування символів) або інші маніпуляції для підготовки даних для подальшого аналізу.

Інструменти для очищення даних. Інструменти для очищення даних грають важливу роль в процесі підготовки даних для подальшого аналізу та використання. Їхнє використання допомагає забезпечити точність, консистентність та повноту даних, а також зменшити час, необхідний для очищення та підготовки даних. [25]

Таблиця 2.1 – Порівняння інструментів для очищення даних

Інструмент	Особливості	Переваги	Недоліки
Excel	Широкі можливості фільтрації та сортування даних	Легкий у використанні, загальна доступність	Обмежена масштабованість, обробка великих обсягів даних
OpenRefine	Відкритий вихідний код, багато функцій очищення	Можливість автоматизації процесу	Вимагає додаткового часу для вивчення
Python (Pandas)	Можливості програмування, багато функцій	Потужність та гнучкість	Вимагає вміння програмувати
SQL	Ефективна робота з базами даних	Добре підходить для великих обсягів даних	Вимагає знання мови SQL
Trifacta	Інтуїтивний інтерфейс, автоматизована обробка	Швидкість роботи, підтримка великих обсягів	Комерційний продукт, обмежена доступність

Microsoft Excel є одним з найпоширеніших інструментів для очищення даних. Він надає ряд функцій та інструментів для фільтрації, сортування, видалення дублікатів, розбиття та об'єднання даних, а також для застосування різних правил форматування та визначення правил перевірки даних. OpenRefine (раніше відомий як Google Refine) - це безкоштовний інструмент з відкритим вихідним кодом, спеціально призначений для очищення та перетворення даних. Він надає широкий спектр функцій для редагування та очищення даних, включаючи фасетування, кластеризацію, розподіл значень, автоматичне виправлення та багато іншого.

Python (бібліотека Pandas). Мова програмування Python в поєднанні з бібліотекою Pandas є потужним інструментом для очищення та аналізу даних. Pandas надає широкий набір функцій для завантаження, обробки, очищення та аналізу даних у табличному форматі, а також для видалення дублікатів, обробки пропусків, зміни типів даних та іншого.

SQL (Structured Query Language) - це мова програмування для роботи з базами даних. Вона може бути використана для очищення даних шляхом виконання різних запитів, фільтрації, сортування, групування, об'єднання таблиць та інших операцій.

Trifacta - це комерційний інструмент для візуального очищення та підготовки даних. Він надає інтуїтивний інтерфейс та ряд функцій для автоматичного виявлення, виправлення та перетворення даних, що допомагає спростити процес очищення даних для користувачів з різним рівнем навичок.

Ці інструменти, разом з іншими доступними ресурсами, допомагають аналітикам та дослідникам ефективно очищувати та підготовлювати дані для подальшого використання у аналізі, моделюванні та прийнятті рішень.

2.3 Інтеграція даних: процес і способи реалізації

Інтеграція даних - це процес об'єднання даних з різних джерел в один цілісний набір даних, що дозволяє забезпечити їхню консистентність та однорідність. [26] Інтеграція даних - це ключовий процес в сучасному управлінні даними, що стає особливо важливим в умовах росту обсягів і різноманітності даних, що накопичуються в компаніях та організаціях. Основна мета інтеграції даних полягає в тому, щоб об'єднати різні джерела даних у єдиний, цілісний набір даних, який буде зручною основою для аналізу, звітності та прийняття рішень. Цей процес включає вирівнювання схем даних, вирішення конфліктів та забезпечення цілісності та якості даних. Для успішної інтеграції даних необхідно використовувати різні методи та підходи.

Інтеграція даних - це ключовий процес у сфері управління даними, який полягає в об'єднанні інформації з різних джерел для створення єдиного, цілісного та згодом корисного набору даних. Основна мета інтеграції даних - забезпечити консистентність, однорідність та доступність даних для подальшого використання в аналізі, звітності, прийнятті рішень та інших процесах бізнесу.

Інформація може знаходитися в різних джерелах, таких як бази даних, текстові файли, спеціалізовані системи, документи тощо. Інтеграція даних передбачає збір і об'єднання цих даних в один централізований набір. Часто дані можуть бути неоднорідними, містити помилки, дублікати або бути представленими у різних форматах. Під час інтеграції важливо провести процес очищення, стандартизації та вирівнювання даних для забезпечення їхньої якості та зрозумілості. Інтегрований набір даних повинен бути консистентним та цілісним. Це означає, що дані повинні бути узгоджені між собою та відображати одну версію правди. При інтеграції даних важливо забезпечити, щоб інформація завжди була актуальною та оновлювалася вчасно.

3 РОЗРОБКА СЕРВІСУ ЗБОРУ І АНАЛІЗУ ДАНИХ

3.1 **Цілі та очікувані функції сервісу збору даних**

к

Розробка сервісу збору, аналізу і візуалізації даних - це складний та багатоетапний процес, який вимагає як технічних знань, так і розуміння потреб користувачів і особливостей даних, з якими сервіс буде взаємодіяти.

Як розробник такого сервісу, я розглядаю такі вимоги та функції до даного проекту:

– *Ефективний збір даних*. Система повинна бути здатна ефективно збирати дані з різних джерел, а саме: бази даних, файлові системи, хмарні сховища та інші. Вона повинна бути здатна обробляти великі обсяги даних в реальному часі.

– *Масштабованість*. Система повинна бути гнучкою та масштабованою, щоб працювати з великими обсягами даних і забезпечувати продуктивність навіть при збільшенні обсягів даних.

– *Аналіз даних*. Система повинна мати можливості для виконання різноманітних аналітичних завдань, включаючи очищення даних, виявлення патернів, побудову моделей тощо.

– *Візуалізація даних*. Система повинна надавати можливості візуалізації даних в зручний спосіб для користувачів. Це може бути створення графіків, діаграм, карт тощо.

– *Безпека даних*. Забезпечення безпеки даних є важливою вимогою. Система повинна мати вбудовані механізми захисту даних, шифрування та контролю доступу.

– *Інтеграція з іншими системами*. Система повинна бути здатна інтегруватися з іншими існуючими системами, щоб забезпечити обмін даними та спільну роботу з ними.

– *Легкість використання та інтуїтивний інтерфейс*. Інтерфейс користувача повинен бути легким у використанні та інтуїтивно зрозумілим для користувачів будь-якого рівня технічної підготовки.

– *Підтримка та поновлення*. Система повинна мати механізми для підтримки користувачів та регулярного оновлення функціоналу на основі зворотнього зв'язку та нових потреб користувачів.

Це загальний список очікуваних функції даної програми. Проте, це не являється узагальненим списком вимог до кожного подібного сервісу. Конкретні вимоги можуть варіюватися залежно від контексту та потреб користувачів.

3.2 Вибір інструментів

3.2.1 Вибір фреймворку

Фреймворк — це програмне забезпечення, яке надає структуру та набір інструментів для розробки додатків. Фреймворки допомагають розробникам швидше створювати програми, забезпечуючи базову структуру, загальні функції та засоби, що полегшують розробку, тестування та підтримку коду.

Фреймворк відіграє важливу роль у розробці веб-сервісу для збору і аналізу даних. Він допомагає вирішити такі проблеми:

– *Структуризація проєкту*. Фреймворк забезпечує стандартну структуру проєкту, що дозволяє зберігати код організовано та зрозуміло. Це особливо важливо для великих проєктів, які потребують злагодженої роботи декількох розробників.

– *Реалізація CRUD-операцій*. CRUD-операції (Create, Read, Update, Delete) є основними для взаємодії з базами даних. Фреймворки часто надають готові інструменти для роботи з базами даних, що значно спрощує створення, зчитування, оновлення та видалення даних.

– *Підтримка масштабування*. Веб-сервіси для збору та аналізу великих даних потребують ефективного масштабування для обробки великої кількості запитів та даних. Фреймворки забезпечують можливість горизонтального масштабування та підтримку розподілених обчислень.

– *Інтеграція з інструментами для роботи з великими даними*. Багато фреймворків мають інтеграції з інструментами та платформами для роботи з великими даними, такими як Hadoop, Apache Spark, та інші. Це дозволяє легко обробляти великі обсяги даних і виконувати складні аналітичні задачі.

– *Безпека*. Фреймворки надають механізми для забезпечення безпеки, включаючи аутентифікацію та авторизацію користувачів, захист від загроз типу SQL-ін'єкцій, міжсайтового скриптингу (XSS) та інших типів атак.

– *Модульність та повторне використання коду*. Фреймворки підтримують модульність, що дозволяє створювати компоненти, які можна використовувати повторно в різних частинах проєкту або навіть в інших проєктах.

– *Засоби тестування*. Веб-фреймворки часто включають вбудовані засоби для тестування коду, що полегшує написання тестів і забезпечує високу якість програмного забезпечення.

Під час створення веб-сервісу для збору та аналізу великих даних з різних джере, фреймворк допомогає наступним чином:

– *Збір даних*. Ви можете використати веб-фреймворк (наприклад, Django або Flask для Python) для створення API, яке отримуватиме дані від зовнішніх джерел. Фреймворк допоможе налаштувати маршрутизацію та обробку запитів.

– *Збереження даних*. Використовуючи ORM (Object-Relational Mapping), який надається фреймворком, ви зможете зручно працювати з базою даних. Наприклад, у Django є вбудований ORM, що спрощує взаємодію з базами даних.

– *Аналіз даних*. Для аналізу великих даних можна використовувати інтеграції фреймворку з аналітичними платформами (наприклад, Apache Spark). Фреймворк забезпечить зручну взаємодію з цими інструментами через API.

– *Представлення результатів*. Фреймворк дозволить створити інтерфейс для відображення результатів аналізу даних, використовуючи шаблонні системи для генерації HTML або засоби для створення RESTful API для клієнтських додатків.

Таким чином, фреймворк є важливим інструментом для розробки веб-сервісів, забезпечуючи ефективність, безпеку та підтримку великих обсягів даних.

В даній роботі я обрав фреймворк Python Flask, оскільки він володіє такими перевагами, як гнучкість і модульність, що дозволяє дозволяє додавати тільки ті компоненти, які дійсно будуть потрібні. Гнучка архітектура Flask не накладає

жорстких обмежень на структуру проєкту, що дозволяє адаптувати його під конкретні потреби програми. Також, Flask розроблений для того, щоб бути простим і легким у використанні. Він дозволить швидко створити додаток, не перевантажуючи його непотрібними компонентами.

Flask додатки легко розгортаються на різних платформах, включаючи Heroku, AWS, Google Cloud та інші. Це надає можливість швидко налаштувати середовище розгортання та забезпечити масштабованість веб-ресурсу.

Підтримка віртуальних середовищ. Використання віртуальних середовищ дозволяє ізолювати залежності додатка і зберігати чистоту робочого середовища.

Flask може працювати в поєднанні з Celery для виконання асинхронних задач, що важливо для обробки великих обсягів даних і забезпечення високої продуктивності.

Flask є ідеальним вибором для створення малого веб-ресурсу для збору і аналізу великих даних завдяки своїй легкості, гнучкості, розширюваності та відмінній підтримці спільноти. Використання Flask дозволяє швидко створити ефективний і масштабований додаток, що задовольняє конкретні потреби у зборі та аналізі даних.

3.2.2 Вибір серверу

Для запуску веб-додатку, буде використовуватися срвер Heroku щоб забезпечити його доступність в мережі Інтернет і обробку запитів користувачів. Ця платформа забезпечить безперервну роботу додатків, а також керування даними, безпеку та масштабованість.

Heroku - це хмарна платформа для розгортання, керування і масштабування веб-додатків. Вона надає зручні інструменти для створення, тестування і запуску програмного забезпечення без необхідності управління інфраструктурою. Heroku

підтримує багато мов програмування, включаючи Python, Ruby, Node.js, Java, PHP, Go і багато інших. Heroku автоматично масштабує додаток відповідно до навантаження, додаючи ресурси для забезпечення стабільної роботи. Heroku інтегрується з Git, дозволяючи зручно розгортати код на платформі. Також за допомгою Heroku можна легко інтегрувати різні додаткові сервіси, такі як бази даних, кешування, моніторинг і логування.

Розгортання ресурсу на сервері Heroku відбувається наступним чином:

Перш за все, необхідно створити обліковий запис на Heroku та встановити Heroku Command Line Interface (CLI), що дозволить керувати додатком Heroku з терміналу. Далі, необхідно загрузити код на Heroku за допомогою git, використовуючи команду git push heroku master для цього. Надалі, веб-додаток стане доступним у веб-браузері за допомогою команди heroku open. Це відкриє URL додатку у веб-браузері за замовчуванням.

3.2.3 Вибір апаратного забезпечення

Оскільки Heroku пропонує хмарне середовище, де ресурси можуть масштабуватися відповідно до потреб, основним фактором під час вибору програмного забезпечення буде саме типи сервісу Heroku. Це дозволяє підбирати план під потреби сервісу в реальному часі та переходити на більш просунуті рішення при зростанні навантаження на веб-ресурс. Проте, веб-сервіс також можна розгорнути на власному сервері що має свої переваги та недоліки.

Перш за все. використання власного сервісу надає повний контроль над апаратним забезпеченням, оперативною системою, конфігураціями та іншими аспектами інфраструктури. Це дозволить налаштовувати сервери так, як нам потрібно, і впроваджувати будь-які специфічні вимоги проекту.

Спеціалізовані вимоги. Якщо у веб-ресурсу з'являться особливі вимоги щодо безпеки, регуляторних вимог або високої доступності, розгортання на власному сервері може дати нам більше гнучкості в їх виконанні.

Економічна ефективність. У деяких випадках, коли веб-ресурс великий і потребує значних ресурсів, власні сервери можуть бути економічно вигіднішим варіантом порівняно з хмарними послугами на основі оплати за використання.

Скасування обмежень. На власному сервері немає обмежень політикою хмарного постачальника. Це дозволяє налаштувати будь-яку функціональність, яку потрібно, і використовувати різноманітні інструменти і технології.

Однак розгортання власного сервера також має свої недоліки.

Вартість та складність управління. Розгортання та управління власними серверами може бути витратними як по часу, так і по грошах. Вартість обладнання може досягти великої суми, а також його обслуговування та електроживлення.

Необхідність експертизи. Необхідно володіти належними знаннями або наймати кваліфікованих фахівців для управління та підтримки власних серверів. Це може включати знання мережевої безпеки, адміністрування серверів, резервного копіювання та відновлення даних.

Для самостійного збирання серверів для розгортання веб-ресурсу зі збору та аналізу великих даних, необхідне конкретне апаратне забезпечення. Я хочу запропонувати такі апаратні рішення:

– Процесор (CPU). Я обрав би Intel Xeon Gold 6254 або аналогічний процесор від AMD з високою кількістю ядер (наприклад, 18-24 ядра) та потоків (36-48 потоків). Ці процесори мають вражаючу обробну потужність, що дозволить ефективно обробляти великі обсяги даних та виконувати складні аналітичні операції.

– *Оперативна пам'ять (RAM)*. Я обрав би 64GB або 128GB DDR4 ECC RAM. ECC (Error-Correcting Code) RAM дозволяє виявляти та виправляти помилки в пам'яті, що дуже важливо для надійності при роботі з великими обсягами даних.

– *Сховище даних (Storage)*. Використав би NVMe SSD накопичувачі з ємністю 1TB або більше для швидкого доступу до даних. Додатково, можна використовувати SATA SSD або HDD з великою ємністю (наприклад, 4TB або 8TB) для довгострокового зберігання даних.

– *Материнська плата (Motherboard)*. Обрав би серверну материнську плату з підтримкою двох або більше процесорів та достатньою кількістю слотів для RAM та PCIe розширень. Наприклад, Supermicro X11DPH-T або аналогічну модель від іншого виробника.

– *Охолодження (Cooling)*. Використав би потужну систему охолодження з вентиляторами високої потужності та можливістю регулювання швидкості обертання. Ефективне охолодження забезпечить стабільну роботу сервера в умовах високого навантаження.

3.3 **З Розробка збору даних**

3.3.1 Збір даних із SQL

Перед початком розробки, слід визначити інструменти та сервіси, що будуть використані програмою та для її розробки. Програма буде розроблятися програмною мовою **Python** [27] та за допомогою бібліотеки **psycopg2**, яка необхідна для зв'язку програми із базою даних.

Для збору даних буде використовуватися метод із використанням бази даних. Обрана система керування базами даних – **PostgreSQL**. PostgreSQL - це потужна об'єктно-реляційна система керування базами даних (СКБД), яка зазвичай використовується для зберігання та управління великими обсягами даних.

Використання PostgreSQL має кілька переваг, включаючи високу надійність, підтримку стандартів SQL, розширюваність, велику кількість розширень та підтримку JSON-типу даних. Ця система також відкрита для спільноти, що означає активний розвиток і підтримку з боку спільноти користувачів та розробників.

Перш за все, я розробив просту програму яка підтримує використання користувацького інтерфейсу (UI), на основі якої буде відбуватися подальша розробка.

```
import tkinter as tk
from sqlalchemy import create_engine, Column, Integer, String
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker


def display_message():
    label.config(text="Привіт, світ!")

window = tk.Tk()
window.title("Сервіс збору даних")

label = tk.Label(window, text="Натисніть кнопку, щоб побачити повідомлення")
label.pack(padx=20, pady=20)

button = tk.Button(window, text="Натисни мене", command=display_message)
button.pack(padx=10, pady=10)

window.mainloop()
```

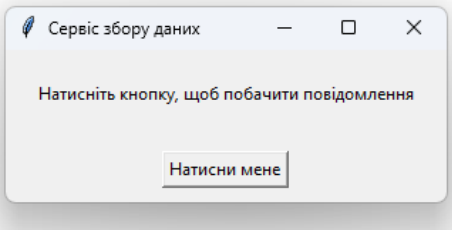


Рисунок 3.1 – Приклад виконання простої програми за допомогою мови Python

Далі, для прикладу застосування програми я створив базу даних за допомогою **psql** (програмна оболонка PostgreSQL) та загрузив випадкові дані із Інтернету. Назви бази даних – "dvdrental".

Далі, програма підключається до бази даних PostgreSQL за допомогою бібліотеки **sqlalchemy** та **psycopg2** [28] для перегляду даних. Програма може обрати таблицю та відобразити її стовпці та дані з них. Перед запуском, програма запитує пароль та логін, а також назву бази даних для підключення.

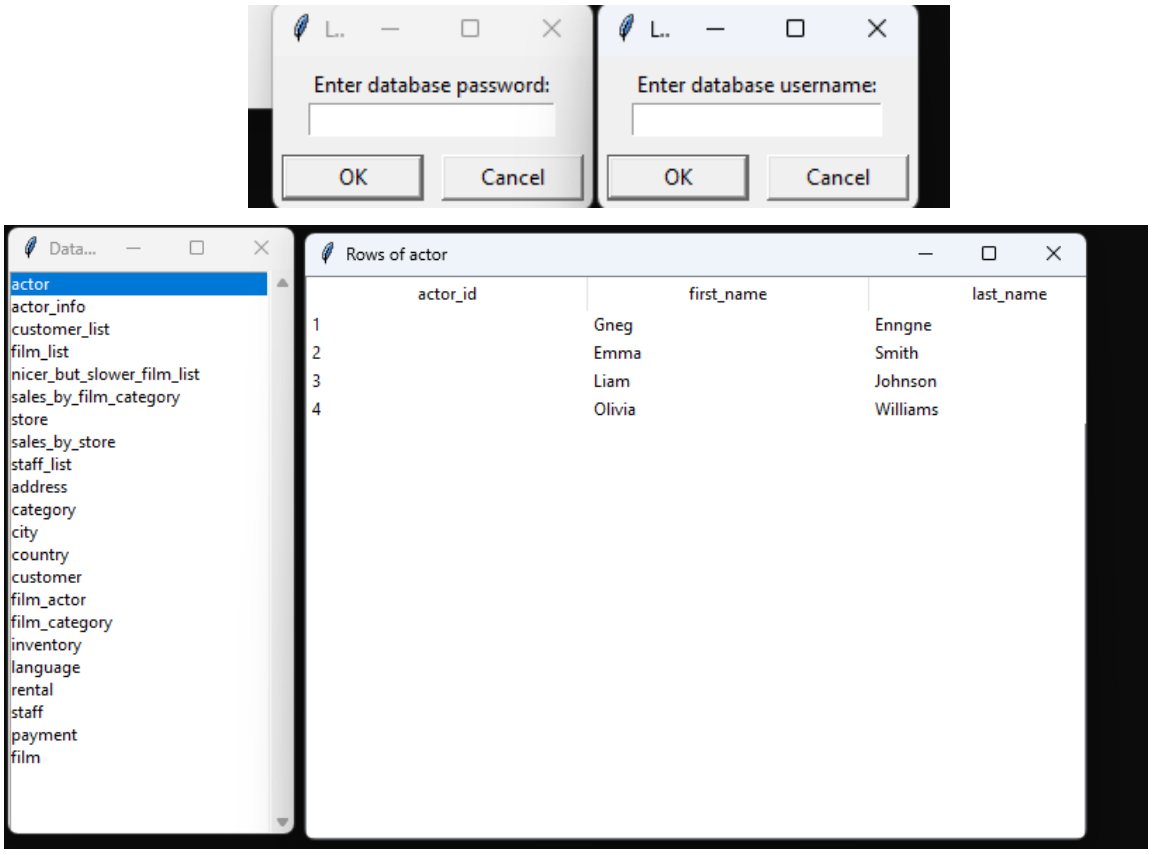


Рисунок 3.2 – Приклад виконання програми для відображення бази даних

3.3.2 Збір даних із файлової системи

Окрім збору даних з SQL, я також реалізую перегляд даних із файлових систем, також за допомогою Python. Існує декілька методів реалізації збору даних:

Бібліотека os: Бібліотека os в Python надає функції для взаємодії з операційною системою, що дозволяє зчитувати інформацію про файли та каталоги, переглядати їх зміст тощо. Наприклад:

```
import os
```

```
# Перегляд вмісту каталогу

files = os.listdir("/шлях до каталогу")

# Зчитування файлу

with open/("/шлях до файлу", "r") as file:

data = file.read()
```

Бібліотека **glob**: Ця бібліотека дозволяє здійснювати пошук файлів за шаблонами імені. Наприклад, ви можете отримати список усіх файлів у певному каталозі з певним розширенням:

```
import glob

files = glob.glob("/шлях_до_каталогу/*.розширення_файлу")
```

Бібліотека **pathlib**: З модулем pathlib можна зручно взаємодіяти з шляхами до файлів та каталогів. Це більш сучасний підхід порівняно з використанням рядків шляху. Приклад використання:

```
from pathlib import Path

# Отримання шляху до файла

file_path = Path("/шлях_до_файлу")

# Перевірка існування файлу

if file_path.exsts():

        with open(file_path, "r") as file:

                data = file.read()

        else

        print("Файл не існує")
```

Бібліотека **pandas**: Бібліотека pandas дозволяє легко зчитувати дані у форматах, таких як CSV, Excel у фрейми даних Python. Наприклад, для зчитування даних з CSV файлу:

```
import pandas as pd

df = pd.read_csv("/шлях_до_файлу.csv")
```

У своїй програмі я реалізую збір даних за допомогою бібліотеки os.

```
import os

import time

directory = input("Вееділ лях додиректорil: ")

if os.path.exists(directory):

	files = os.listdir (directory)

	for file in files:

		file_path = os.path.join (directory, file)

		if os.path.isfile (file_path):

		size = os.path.getsize (file_path)

		last modified = os.path.getmtime (file_path)

		print("Файл: (file), Розмір: (size) bytes, Остання зміна: (last_modified)")

time.sleep(5)
```

Програма запитуватиме необхідну директорію для збору даних, після чого відповість інформацією про цю директорію.

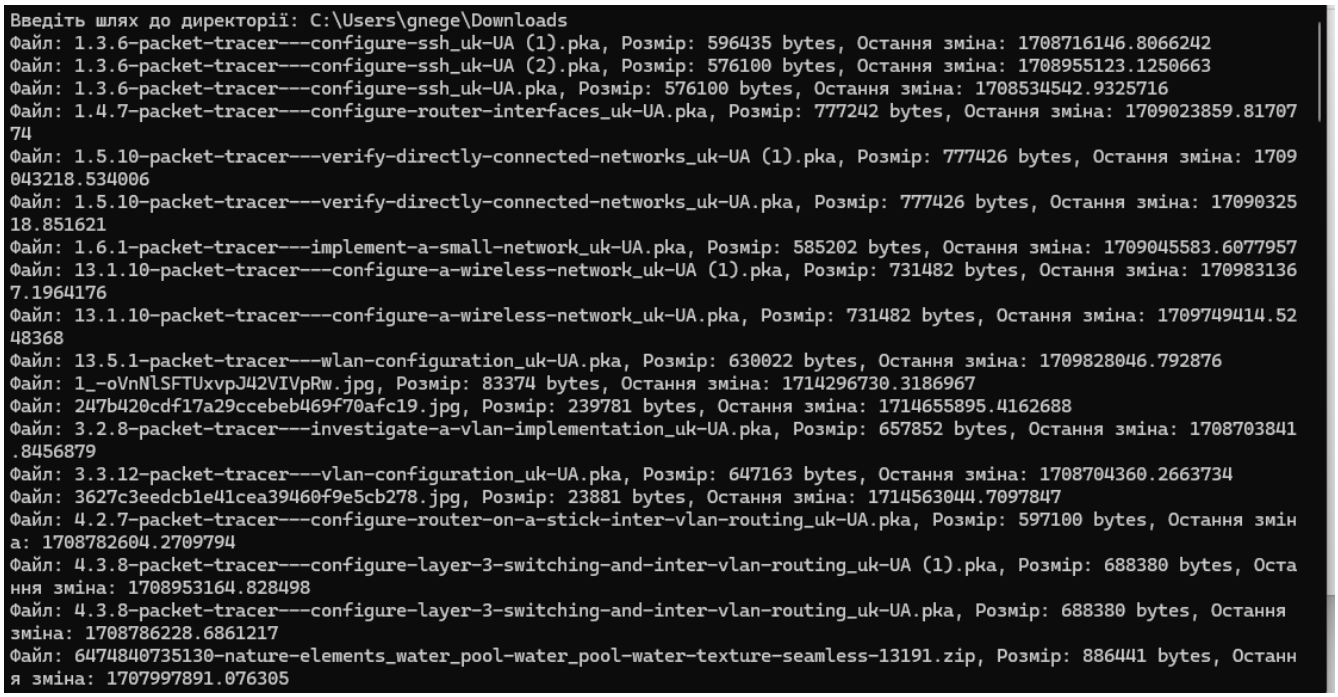


Рисунок 3.3 – Приклад виконання програми для відображення файлів із директорії "Загрузки" на комп'ютері користувача

3.3.3 Збір даних із хмарного сховища

Програма також буде володіти функціоналом збору даних із хмарного сховища. Для прикладу, я скористаюся хмарним сховищем Google Drive. Використання Python дає можливість автоматизувати процеси завантаження, видалення, копіювання та інших операцій із файлами у хмарному сховищі.

Забезпечення програми методом збору даних із Google Drive, або інших хмарних сховищ має багато переваг, одні з таких:

–*Доступність*. Дані можна звертатися з будь-якого місця, де є Інтернет, що робить їх легко доступними для спільного використання з колегами або збереженням з різних пристроїв.

– *Резервне копіювання*. Хмарні сховища можуть служити як засіб резервного копіювання важливих даних. Вони захищені від втрати через випадкове видалення або пошкодження пристрою.

– *Спільне використання*. Хмарні сховища дозволяють легко спільно користуватися даними з іншими користувачами, роблячи співпрацю більш ефективною.

– *Зручність*. Вони забезпечують зручний спосіб організації даних у папки та підпапки, а також швидкий доступ до них через інтерфейс хмарного сховища.

– *Заощадження часу*. Використання хмарного сховища може заощадити час, оскільки не потрібно копіювати файли на зовнішні пристрої або надсилати їх електронною

Для взаємодії з Google Drive за допомогою Python, необхідно виконати наступні кроки:

1. Створити проект Google та увімкнути API Google Drive: Необхіднр перейти на сторінку консолі розробника Google (https://console.developers.google.com/), створити новий проект або обрати вже існуючий. Потім увімкнути API Google Drive в налаштуваннях проекту.
2. Створити авторизаційні дані для проєкту, вибравши тип облікового запису (наприклад, обліковий запис служби) і завантажити JSON-файл, який містить ці дані.
3. Встановити бібліотеку PyDrive за допомогою pip.

3.4 **Розробка процесу аналізу та візуалізації даних**

Аналіз та візуалізація зібраних даних допомагають виявляти зв'язки, тенденції та патерни, які можуть залишитися непоміченими при поверхневому огляді. Вони дозволяють робити обґрунтовані висновки, приймати стратегічні рішення та прогнозувати майбутні події на основі фактичних даних. Це надає можливість покращити бізнес-процеси, розробляти нові продукти та послуги, а також оптимізувати виробництво та ресурсоємні процеси.

Я реалізую візуалізацію і аналіз даних шляхом програмування мовою Python і доповню логіку уже існуючої програми. Існує кілька бібліотек для візуалізації даних з бази даних PostgreSQL в Python, найпоширеніші із них:

– *Matplotlib*. Це основна бібліотека для створення графіків у Python. Вона може працювати з будь-якими даними, включаючи дані з PostgreSQL.

– *Seaborn*. Це високорівнева бібліотека для візуалізації даних на основі Matplotlib. Вона зручна для створення стильних графіків та діаграм.

– *Plotly*. Ця бібліотека дозволяє створювати інтерактивні графіки, які можна вбудувати в веб-сторінки або навіть створювати візуалізації з аналізом даних прямо у Jupyter Notebook.

– *Pandas Visualization*. Це вбудована візуалізація, яка базується на Matplotlib і розроблена для простоти використання з DataFrame з Pandas, що дозволяє легко створювати графіки безпосередньо з даних з PostgreSQL.

– *SQLAlchemy*. Це не саме бібліотека візуалізації, але ORM (Object-Relational Mapping) для роботи з базами даних у Python. Вона може взаємодіяти з PostgreSQL і забезпечити доступ до даних для будь-якої іншої бібліотеки візуалізації, яку ви виберете.

Я напишу програму за допомогою бібліотеки Matplotlib:

Для початку, слід реазіувати підключення до бази даних, це робиться наступним шляхом:

```
def connect_to_database():
try:
connection = psycopg2.connect (
user = "your_username",
password = "your_password",
host="your_host",
port="your_port"
database="dvdrental"
            )
            return connection
```

У цій ділянці коду, визначена функція connect_to_database, яка відповідає за підключення до бази даних PostgreSQL. Функція використовує psycopg2.connect() для створення з'єднання з базою даних, використовуючи вказані параметри. Якщо підключення успішне, повертається об'єкт з'єднання, інакше виводиться повідомлення про помилку.

Далі, програма визначає функцію fetch_data, яка виконує запит до бази даних і отримує дані з таблиці "actor". Вона використовує connection.cursor() для створення курсора, потім виконує запит SELECT * FROM actor, отримує всі дані та повертає їх. Якщо виникає помилка, виводиться повідомлення про помилку.

```
import os
def fetch_data(connection):
cursor = connection.cursor()
try:
       cursor.execute("SELECT * FROM actor")
actors = cursor.fetchall ()
```

```
return actors

except (Exception, psycopg2.Error) as error:

print("Помилка отримання даних з PostgreSQL", error
```

У цій частині програми реалізується візуалізація даних шляхом визначення функції visualize_data, яка використовує бібліотеку matplotlib для створення стовпчастої діаграми. Дані про імена акторів та їх ідентифікатори використовуються для створення діаграми. На виході ми бачимо стовпчасту діаграму з ім'ями акторів на осі Y та їх ідентифікаторами на осі X.

```
      def visualize_data (actors):
            actor_names = [actor [1] for actor in actors]
            actor_ids
            actor ids - [actor [0] for actor in actors]
            plt.bar (actor_ids, actor_names, color='skyblue')
            plt.xlabel('Actor ID')
            plt.ylabel('Actor Name')
            plt.title('Actors
            in DVD Rental Database')

def main():
      connection connect_to_database()
      if connection:
            actors fetch_data (connection)
            if actors:
                  visualize data (actors)
                  connection.close()
      else:
            print("No data retrieved from database.")
else:
```

```
print ("Connection to database failed.")
```

Нарешті, визначається головна функція main(), яка викликає функції підключення до бази даних, отримання даних та візуалізації даних. Якщо всі кроки успішні, з'єднання з базою даних закривається. Якщо виникає помилка під час будь-якого з кроків, виводиться повідомлення про помилку.

Таким чином, реалізується програма візуалізації даних у графіку:

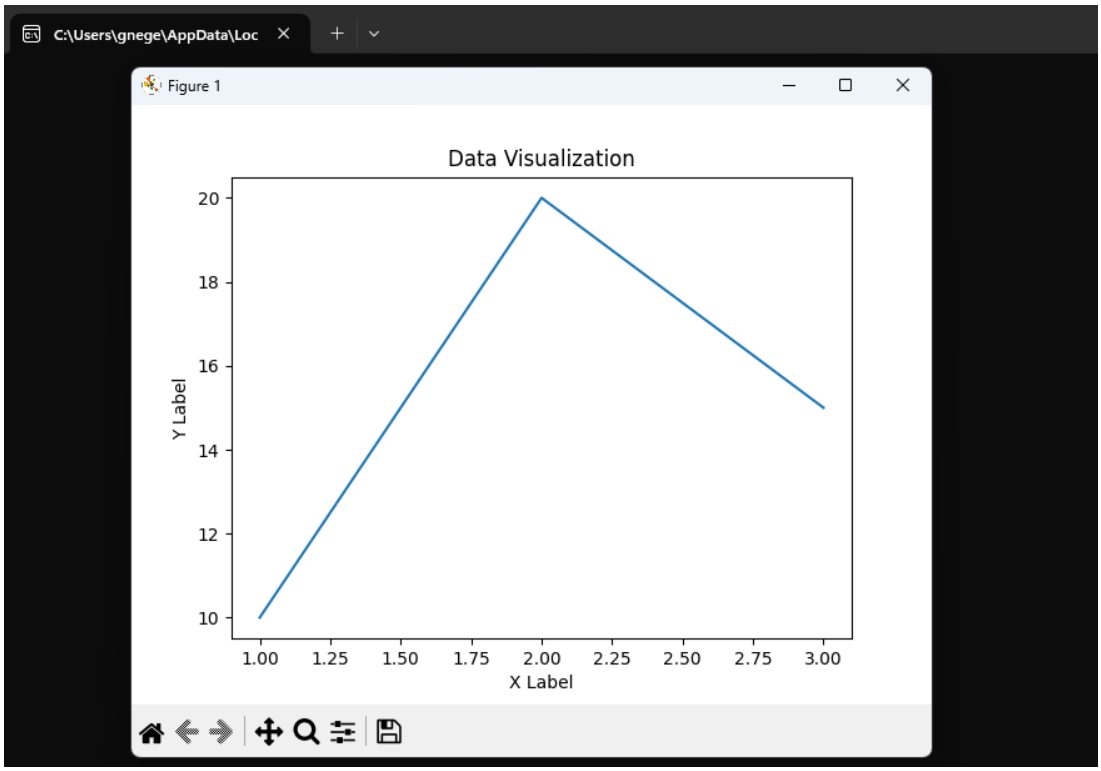


Рисунок 3.4 – Приклад виконання програми для візуалізації графіків даних

Цю програму можна використати як фунцію сервісу збору даних, ось результат її виконання для стовпця "actors" у базі даних "dvdrental":

Рисунок 3.5 – Приклад виконання програми для візуалізації графіків даних

Окрім візуалізації, можна застосовувати інші види аналізу даних, серед яких можуть бути:

Описовий аналіз. Вивчення основних статистичних характеристик даних, таких як середнє значення, медіана, мода, дисперсія, квантилі і т.д. Це допомагає зрозуміти загальну структуру даних.

```
quary = "SELECT * FROM ваша_таблиця"

data = pd.read_sql_query(query, conn)
```

```
# Описовий аналіз

descripton = data.describe()

print(description)
```

У цьому фрагменті коді дані зчитуються у DataFrame з допомогою бібліотеки **pandas**, і потім використовується метод **describe()** для отримання описової статистики про числові стовпці даних.

– *Агрегація та групування даних*. Групування даних за певними критеріями і обчислення агрегованих значень, таких як сума, середнє значення, медіана тощо.

```
cur = conn.cursor()

      # Виконання SQL-запиту в агрегацією та групуванням даних

            cur.execute("SELECT region, SUM(sales) FROM ваша таблиця GROUP BY region")

      # Отримання результатів

            rows - cur.fetchall ()

      # Виведення результатів

            for row in rows:

            print(row)
```

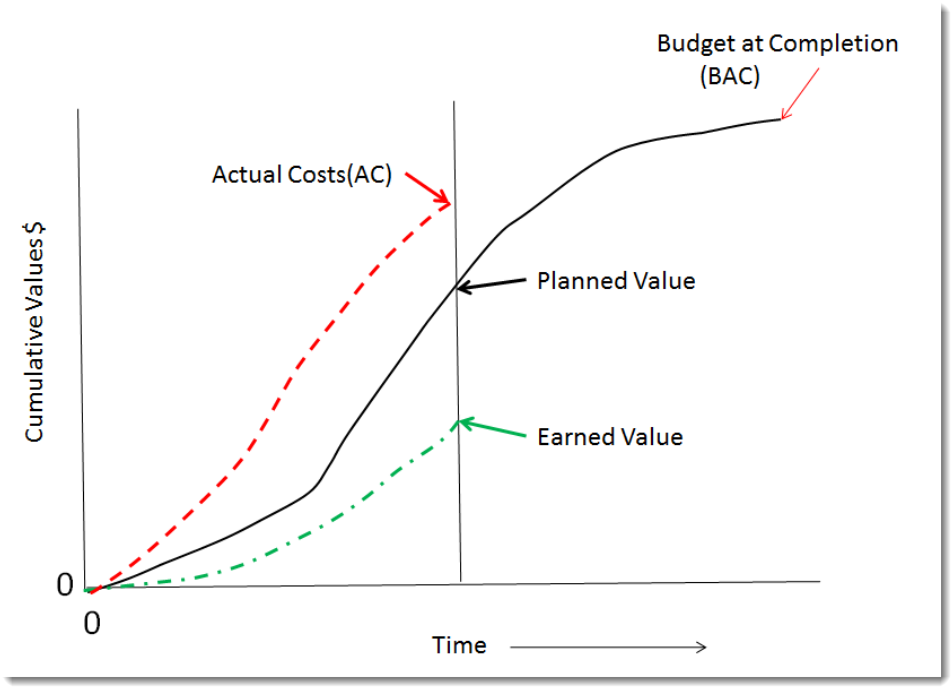
У цьому прикладі SQL-запит використовує функцію SUM() для агрегації значень стовпця sales для кожної групи, яку визначається в стовпці region, за допомогою оператора GROUP BY. Таким чином, можна отримати загальну суму продажів для кожного регіону.

– *Трендовий аналіз*. Вивчення змін у часі для виявлення трендів, сезонних варіацій та інших регулярних патернів у даних.

```
window_size = 7 # розмір вікна для скользячої середньої

      df['moving_avg'] = df['value_column'].rolling (window=window_size).mean()

      # Побудова графіка

      plt.figure(figsize=(10, 6))
```

```
plt.plot(df['date_column'], df['value_column'], label='Оригінальні дані')

plt.plot(df['date_column'], df['moving_avg'], label='Скользяча середня',
color='red')

plt.xlabel('Дата')

plt.ylabel('Значення')

plt.title('Трендовий аналіз')

plt.legend()

plt.show()
```

Цей код використовує бібліотеку Pandas для обробки даних та бібліотеку Matplotlib для побудови графіків. Ми використовуємо скользячу середню для згладжування даних і показуємо оригінальні дані та згладжені дані на одному графіку для порівняння.



Створення веб-ресурсу

Було створено невелику веб-сторінку, яка виглядає наступним чином:

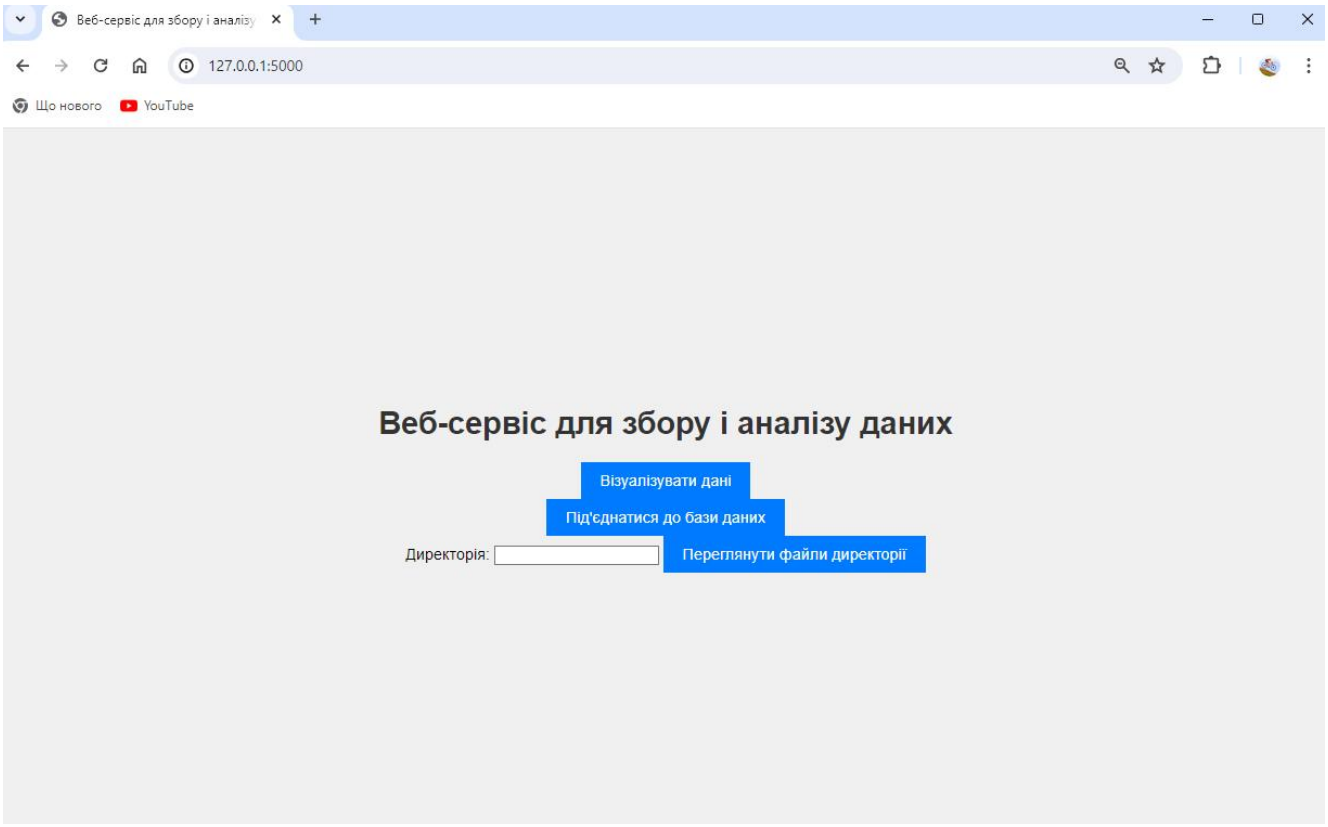


Рисунок 3.7 – Вигляд веб-сторінки сервісу збору даних у браузері Google Chrome

На даному етапі, сервіс володіє наступими функціямим: підключення та перегляд бази даних, візуалізації даних, перегляд директорії фалової системи.

Сторінка реалізована наступним чином: у директорії веб-сервісу створені такі файли: index.html, app.py, style.css та Python скрипти, написані у попередніх розділах.

Повний код кожного файлу можна побачити у додатках, проте я поверхнево опишу що робить кожний файл:

Файл index.html є основним файлом веб-сайту, який вказує браузеру, який контент відображати користувачеві при переході на домашню сторінку (головну сторінку) сайту. Основна його функція - це визначення структури та вмісту головної сторінки веб-сайту.

Цей файл містить HTML-код, який визначає структуру сторінки, такі як заголовок, текст, зображення, посилання і так далі. Крім того, він може містити посилання на зовнішні ресурси, такі як CSS-стилі та JavaScript-скрипти, які впливають на вигляд та поведінку сторінки.

Браузери автоматично шукають і завантажують файл index.html з кореневого каталогу веб-сайту, коли користувач відвідує домашню сторінку. Таким чином, цей файл є важливою частиною будь-якого веб-сайту, оскільки він визначає перший враження користувача від нього.

Файл app.py - це файл, який містить серверний код для веб-сайту, написаний на Python, з використанням фреймворка веб-додатків, такого як Flask або Django.

Основна його функція - це обробка запитів, які надходять на сервер від клієнтів (браузерів) і відправка відповідей назад до них. Файл app.py містить обробники маршрутів, які визначають, які дії виконувати при різних типах запитів (GET, POST, і так далі) до різних URL-адрес. У даній роботі, файл app.py використовується для взаємодії з базою даних та файловою системою користувацького комп'ютера.

Файл **style.css** використовується для стилізації веб-сторінок з використанням каскадних таблиць стилів (CSS). Основна його функція - це визначення зовнішнього вигляду (стилю) елементів HTML на сторінці.

При натисканні клавіші "Візуалізація даних", відкривається нова сторінка яка виконує Python скрипт візуалізації даних.

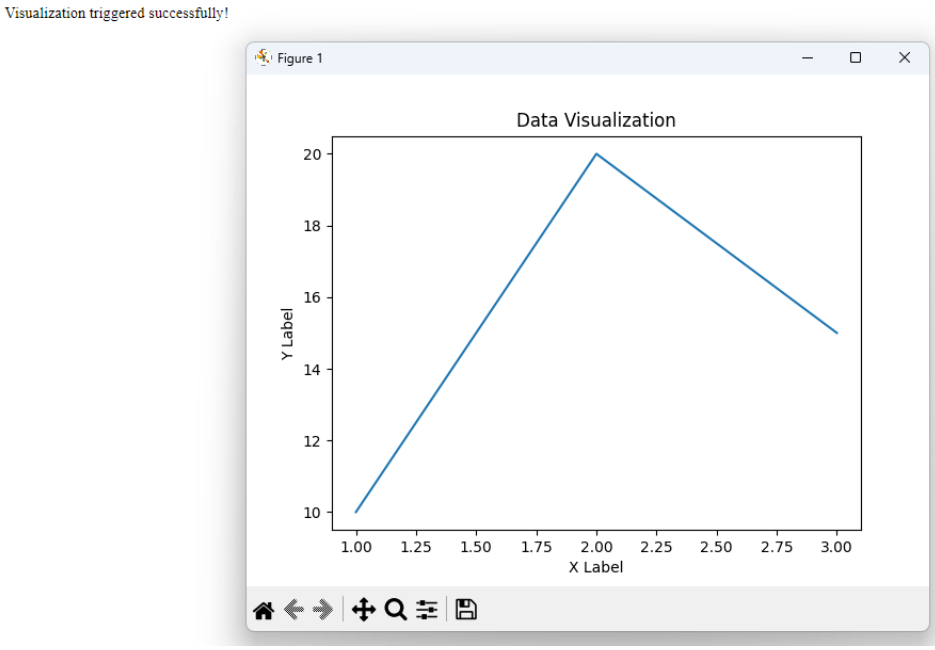


Рисунок 3.7 – Функція візуалізації на веб-сайті сервісу

Аналогічно, інші кнопки виконують відповідні функції за допомогою скриптів, написаних в попередніх розділах.

У подальшому, сервіс можна доповнювати додатковими функціями, такими як розширені методи аналізу даних, а також збір даних із інших джерел, таких як веб-сторінки чи API.

ВИСНОВКИ

Розробка сервісу для збору і аналізу великих даних з різних джерел є надзвичайно актуальною та перспективною задачею в сучасному світі. Цей процес стає все більш важливим для бізнесу, науки, медицини та інших галузей, оскільки дозволяє отримати цінні дані та приймати обґрунтовані рішення. Розроблений сервіс дозволяє автоматизувати процес збору, обробки та аналізу даних, що дозволяє зробити прийняття рішень більш обгрунтованим і швидким.

Впровадження такого сервісу може значно підвищити конкурентоспроможність компанії та допомогти виявити нові можливості для розвитку бізнесу.

У цій роботі було проаналізовано процес розробки та впровадження сервісу для збору і аналізу великих даних з різних джерел. Результати показали, що ефективне використання великих обсягів даних може стати ключовим фактором у успішному управлінні та розвитку бізнесу. Також, ця робота показала, що розробка такого сервісу потребує інтеграції різноманітних технологій, таких як програмування, обробка великих даних, хмарні технології та інші. Важливим етапом є розробка ефективних алгоритмів для аналізу великих обсягів даних.

Однак, успішна реалізація такого сервісу можлива лише за умови налагодження системи збору та збереження даних, а також врахування вимог до їх безпеки та конфіденційності. Важливо також розвивати інструменти візуалізації даних, щоб користувачі могли легко зрозуміти складні зв'язки та тренди у великих масивах інформації.

У підсумку, розробка сервісу для збору і аналізу великих даних відкриває нові можливості для управління даними та прийняття стратегічних рішень, а також має великий потенціал у різних галузях

.

ПЕРЕЛІК ПОСИЛАНЬ

1. Data growth and its impact on the SCOP database: new developments / A. Andreeva та ін. Nucleic acids research. 2007. Т. 36, Database. С. D419–D425. URL: https://doi.org/10.1093/nar/gkm993 (дата звернення: 07.04.2024).
2. Kushnir O. K., Chaplinsky V. R. Statistical methods for big data analysis. Modern economics. 2023. Т. 39, № 1. С. 75–81. URL: https://doi.org/10.31521/modecon.v39(2023)-11 (дата звернення: 04.04.2024).
3. Tukey J. W. The future of data analysis. The annals of mathematical statistics. 1962. Т. 33, № 1. С. 1–67. URL: https://doi.org/10.1214/aoms/1177704711 (дата звернення: 07.04.2024).
4. CIAO: chandra's data analysis system / A. Fruscione та ін. SPIE astronomical telescopes + instrumentation, м. Orlando, Florida , USA / ред.: D. R. Silva, R. E. Doxsey. 2006. URL: https://doi.org/10.1117/12.671760 (дата звернення: 07.04.2024).
5. Trends in big data analytics / K. Kambatla та ін. Journal of parallel and distributed computing. 2014. Т. 74, № 7. С. 2561–2573. URL: https://doi.org/10.1016/j.jpdc.2014.01.003 (дата звернення: 12.04.2024).
6. Fan J., Han F., Liu H. Challenges of Big Data analysis. National science review. 2014. Т. 1, no. 2. С. 293–314. URL:: https://academic.oup.com/nsr/article/1/2/293/1397586?login=false (дата звернення: 01.04.2024)
7. Microsoft Azure – Вікіпедія. *Вікіпедія*. URL: https://uk.wikipedia.org/wiki/Microsoft_Azure (дата звернення: 06.04.2024).
8. Офіційна сторінка Apache Hadoop URL: https://hadoop.apache.org/ (дата звернення: 06.06.2024).

9. Ramse P., Patil D. Data science with python. SSRN electronic journal. 2023. URL: https://doi.org/10.2139/ssrn.4666906 (дата звернення: 07.04.2024).
10. Willenborg L., de Waal T. Data analytic impact of SDC techniques on microdata. Elements of statistical disclosure control. New York, NY, 2001. С. 71–91. URL: https://doi.org/10.1007/978-1-4613-0121-9_3 (дата звернення: 21.03.2024).
11. Chow S.-C. Analysis of incomplete data. Chapman & hall/crc biostatistics series. 2002. URL: https://doi.org/10.1201/9780203910146.ch9 (дата звернення: 17.04.2024).
12. Hand D. J. Principles of data mining. 2007. Т. 30, № 7. С. 621–622. URL: https://doi.org/10.2165/00002018-200730070-00010 (дата звернення: 01.05.2024).
13. David Miller. Machine Learning for Data Analysis: A Practical Approach URL: https://www.sciencedirect.com/book/9780128213797/practical-machine-learning-for-data-analysis-using-python (дата звернення: 21.04.2024)
14. Sarah Garcia. "Understanding Statistical Methods for Data Analysis. URL: https://www.researchgate.net/publication/281280152_Understanding_the_process_of_statistical_methods_for_effective_data_analysis (дата зверненя: 24.04.2024).
15. Fisher C., Sanderson P. Exploratory sequential data analysis. ACM SIGCHI Bulletin. 1993. Т. 25, № 1. С. 34–40. URL: https://doi.org/10.1145/157203.157208 (дата звернення: 07.05.2024).
16. Data analysis methods for prospectivity modelling as applied to mineral exploration targeting: state-of-the-art and outlook / M. Yousefi та ін. Journal of geochemical exploration. 2021. Т. 229. С. 106839. URL: https://doi.org/10.1016/j.gexplo.2021.106839 (дата звернення: 07.04.2024).
17. Fischer M. M., Wang J. Introduction. Spatial data analysis. Berlin, Heidelberg, 2011. С. 1–11. URL: https://doi.org/10.1007/978-3-642-21720-3_1 (дата звернення: 21.04.2024).

18. Coding. Qualitative data analysis: an introduction. 1 Oliver's Yard, 55 City Road London EC1Y 1SP, 2013. С. 258–267. URL: https://doi.org/10.4135/9781529799606.n21 (дата звернення: 02.05.2024).
19. Gazis A. What is iot? The internet of things explained. Academia letters. 2021. URL: https://doi.org/10.20935/al1003 (дата звернення: 13.05.2024).
20. Data lakes / ред.: A. Laurent, D. Laurent, C. Madera. Wiley, 2020. URL: https://doi.org/10.1002/9781119720430 (дата звернення: 10.05.2024).
21. Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google File System. URL: https://static.googleusercontent.com/media/research.google.com/ru//archive/gfs-sosp2003.pdf
22. E. F. Codd. A Relational Model of Data for Large Shared Data Banks
23. Dorian Pyle. Data Preparation for Data Mining.
24. First steps in qualitative data analysis: transcribing. URL: First steps in qualitative data analysis: transcribing (дата звернення: 25.03.2024)
25. Rahm E., Do H. H. Data cleaning: problems and current approaches. 2000. URL: https://ul.qucosa.de/id/qucosa:32968 (дата звернення: 15.04.2024).
26. Python Tutorials. URL: https://www.w3schools.com/python/default.asp (дата зверненя: 21.04.2024).
27. Офіційна сторінка Python URL: https://www.python.org/ (дата зверненя: 15.04.2024).

Додаток А

```
from flask import Flask, render_template, request
import subprocess
app = Flask(__name__)
app.config['TEMPLATES_AUTO_RELOAD'] = True
@app.route('/')
def index():
    return render_template('index.html')
@app.route('/trigger-visualization', methods=['POST'])
def trigger_visualization():
    # Скрипт для візуалізації
    subprocess.Popen(["python", "visualization_script.py"])
    return "Visualization triggered successfully!"
@app.route('/trigger-database-viewer', methods=['POST'])
def trigger_database_viewer():
    # Скрипт для баз даних
    subprocess.Popen(["python", "database_viewer_script.py"])
    return "Database viewer triggered successfully!"
@app.route('/list-files', methods=['POST'])
def list_files():
    # Скрипт для файлової системи
    directory_path = request.form['directory_path']
    subprocess.run(["python", "list_files.py", directory_path], check=True)
    with open('list_files_output.txt', 'r') as f:
        output = f.read()
    return output
if __name__ == '__main__':
```

```
    app.run(debug=True)
```

Додаток 1.1 Скрипт основної програми мовою Python (файл app.py)

```
<!DOCTYPE html>

<html lang="uk">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Веб-додаток для збору і аналізу даних</title>

<link rel="stylesheet" href="https://pyscript.net/releases/2024.1.1/core.css" />

<script type="module" src="https://pyscript.net/releases/2024.1.1/core.js"></script>


</head>

<body>

    <h1>Веб-додаток для збору і аналізу даних</h1>

</body>

</html>
```

Додаток 1.2 – Скрипт веб-файлу HTML (файл index.html)

```
import tkinter as tk
import tkinter.simpledialog as sd
import psycopg2
from tkinter import ttk

class DatabaseViewer(tk.Tk):
    def __init__(self):
        super().__init__()

        self.title("Database Viewer")
        self.geometry("600x400")
```

```
        self.db_username = sd.askstring("Login", "Enter database username:")
        self.db_password = sd.askstring("Login", "Enter database password:", show="*")
        self.db_name = sd.askstring("Login", "Enter database name:")

        try:
            self.connection = psycopg2.connect(
                dbname=self.db_name, user=self.db_username, password=self.db_password,
host="localhost", port="5432"
            )
            self.cursor = self.connection.cursor()
        except psycopg2.Error as e:
            tk.messagebox.showerror("Error", f"Error connecting to database: {e}")
            self.destroy()
            return

        self.table_listbox = tk.Listbox(self)
        self.table_listbox.pack(side="left", fill="both", expand=True)
        self.table_listbox.bind("<<ListboxSelect>>", self.show_rows)

        self.scrollbar = tk.Scrollbar(self, orient="vertical")
        self.scrollbar.pack(side="right", fill="y")
        self.table_listbox.config(yscrollcommand=self.scrollbar.set)
        self.scrollbar.config(command=self.table_listbox.yview)

        self.show_tables()

    def show_tables(self):
        self.cursor.execute("SELECT table_name FROM information_schema.tables WHERE
table_schema='public'")
        tables = self.cursor.fetchall()
        for table in tables:
            self.table_listbox.insert("end", table[0])

    def show_rows(self, event):
        selected_table = self.table_listbox.get(self.table_listbox.curselection())
        if not selected_table:
```

```
            return

        self.cursor.execute(f"SELECT * FROM {selected_table}")
        rows = self.cursor.fetchall()
        columns = [desc[0] for desc in self.cursor.description]

        row_window = tk.Toplevel(self)
        row_window.title(f"Rows of {selected_table}")
        row_window.geometry("600x400")

        tree = ttk.Treeview(row_window)

        tree["columns"] = tuple(range(len(columns))) if rows else (0,)
        tree["show"] = "headings"

        for i, column_name in enumerate(columns):
            tree.heading(i, text=column_name)

        for row in rows:
            tree.insert("", "end", values=row)

        tree.pack(fill="both", expand=True)


if __name__ == "__main__":
    app = DatabaseViewer()
    app.mainloop()
```

Додаток 1.3 – Скрипт перегляду бази даних мовою Python

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Навчально-науковий інститут Інформаційних технологій
Кафедра Комп'ютерної інженерії

КВАЛІФІКАЦІЙНА РОБОТА НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

на тему: « Створення онлайн-сервісу для аналізу і візуалізації даних з різних джерел »

Виконав студент групи КІД-41
Марусяк Владислав
Васильович

Керівник кваліфікаційної роботи:
Розмаїтий Дмитро
Олегович викладач кафедри КІ

Київ – 2024

Мета роботи – розробка та впровадження сервісу для ефективного збору та аналізу великих обсягів даних з різних джерел з метою отримання цінної інформації для прийняття управлінських рішень та виявлення нових можливостей.

Об'єкт дослідження – процес збору та аналізу великих обсягів даних з різних джерел, таких як інтернет, датчики, соціальні мережі та інші джерела.

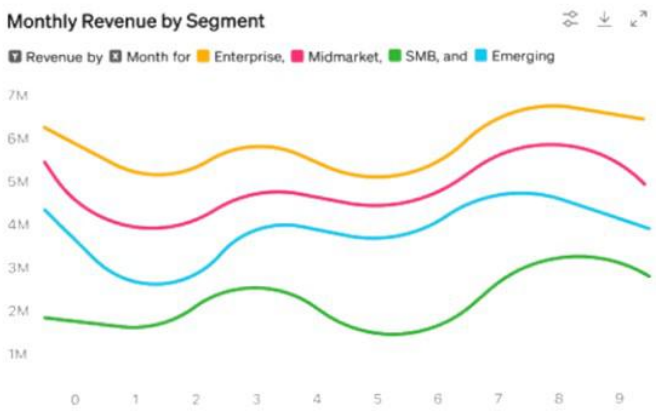
Предмет дослідження – розробка та впровадження програмного забезпечення, обробки та аналізу великих обсягів даних з різних джерел з метою отримання актуальної та корисної інформації.

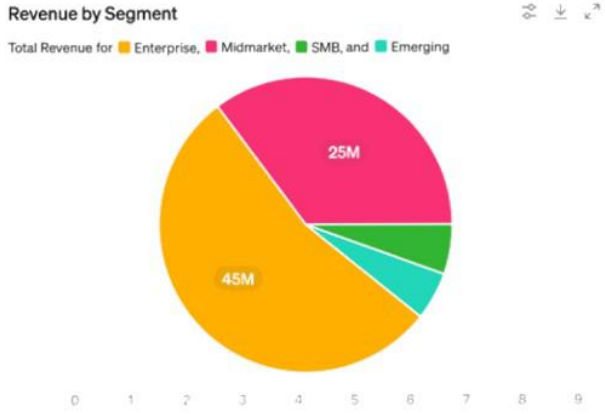
Актуальність обраної теми

Робота спрямована на розробку сервісу, який забезпечить можливість автоматизованого збору та аналізу великих обсягів даних з різних джерел, включаючи структуровані та неструктуровані дані. Досліджуються методи обробки, збереження та аналізу даних, а також розробляються алгоритми для виявлення закономірностей та трендів у великих масивах інформації. Результатом роботи є створення потужного інструменту для прийняття управлінських рішень та виявлення нових можливостей для розвитку бізнесу на основі аналізу великих даних.

Використання сервісу для аналізу та візуалізації великих даних має наступні переваги:

•**Обробка великих обсягів даних**: Ці сервіси здатні ефективно обробляти та аналізувати величезні обсяги даних, що дозволяє отримувати цінну інформацію з великих наборів даних, які традиційні методи обробки не можуть опрацювати.

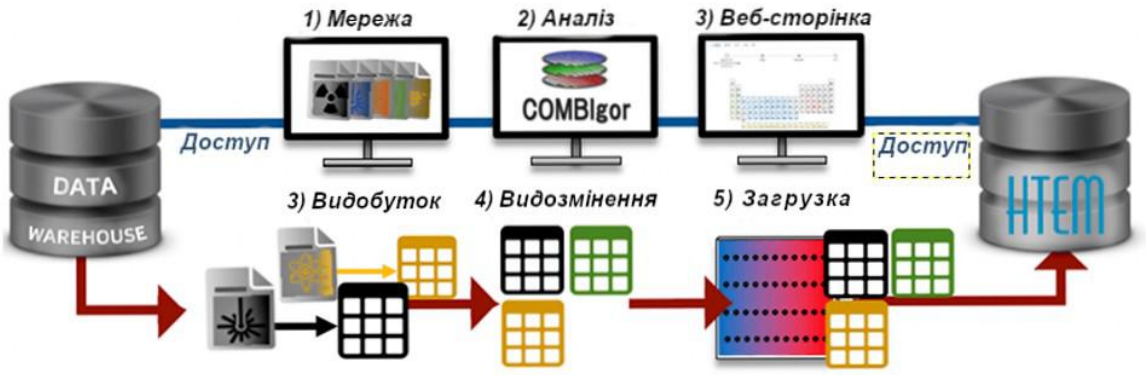




- **Кібербезпека**: Такі сервіси також включають функції кібербезпеки, які забезпечують захист даних від несанкціонованого доступу та кібератак. Це включає шифрування даних, автентифікацію користувачів, контроль доступу та моніторинг безпеки.

5

- **Автоматизація процесів**: Використання такого сервісу включає в себе інструменти для автоматизації складних процесів збору, очищення, обробки та аналізу даних, що економить час і зусилля аналітиків, а також дозволяє отримувати інформацію у реальному часі.



6

Інструменти для реалізації сервісу збору даних

Python - це потужна та динамічна мова програмування, яка має безліч бібліотек та фреймворків для різних задач. Чому Python підходить для такого проєкту:

- **Універсальність**: Python підходить для різних видів проєктів, від веб-розробки до наукових обчислень. Це дозволить нам використовувати одну мову для різних завдань, що зручно та ефективно.
- **Багата екосистема**: Python має велику кількість бібліотек та фреймворків для різних потреб, включаючи веб-розробку, обробку даних, машинне навчання та інше. Це робить його універсальним і потужним інструментом для розробки будь-якого типу веб-додатків.

Flask - це легкий і простий у використанні веб-фреймворк для Python. Я обрав саме цей інструмент через такі переваги:

- **Простота**: Flask має просту структуру та не накладає обмежень щодо архітектури додатку, що робить його ідеальним для маленьких та середніх проєктів.
- **Гнучкість**: Flask дає можливість використовувати різні бібліотеки та інструменти, що робить його дуже гнучким

Вибір серверу

У своїй роботі, заради демонстрації виконання програми, я запустив її на локальному сервері комп'ютера. Проте, для коректного розгортання такої програми у масштабах підприємства, вигідніше використовувати більш просунуті апаратні рішення. Серед яких можуть бути:

- Віртуальні приватні сервери VPS: простота налаштування, керованість, гнучкість, масштабованість. Проте ресурси обмежені провайдером.
- Сервери з хмарними платформами: Доступність, послуги та інструменти, підтримка. Може бути дорого при довгостроковому використанні послуг провайдера.
- Самостійно зібраний сервер: Повний контроль, безпека, незалежність.

Проте, збирання, налаштування, та підтримка сервера повністю залежить від вас та потребує значного часу та зусиль.

Цілі та вимоги даного сервісу

- Ефективний збір даних: Система повинна бути здатна ефективно збирати дані з різних джерел, а саме: бази даних, файлові системи, хмарні сховища та інші.
- Масштабованість: Система повинна бути гнучкою та масштабованою, щоб працювати з великими обсягами даних і забезпечувати продуктивність навіть при збільшенні обсягів даних.
- Аналіз даних: Система повинна мати можливості для виконання різноманітних аналітичних завдань, включаючи очищення даних, виявлення патернів, побудову моделей тощо.
- Візуалізація даних: Система повинна надавати можливості візуалізації даних в зручний спосіб для користувачів. Це може бути створення графіків, діаграм, карт тощо.

Структура проєкту

- Даний сервіс буде складатися із таких елементів: Веб-сторінка, яка надаватиме доступ користувачеві до сервісу, веб-фраємворк, за допомогою якого веб-сторінка буде читати python-скрипти, візуальний інтерфейс, створений за допомогою бібліотек Python та CSS та сервер, на якому ресурс буде розгортатися.

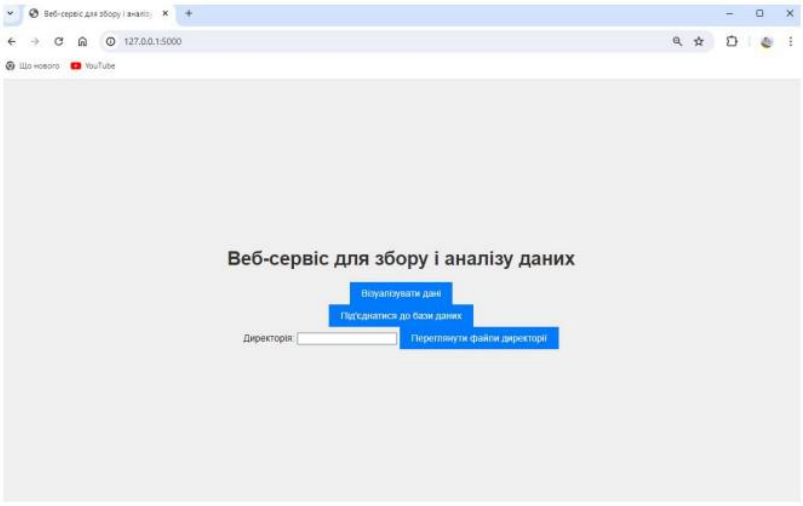
Опис попкрової роботи застосунку

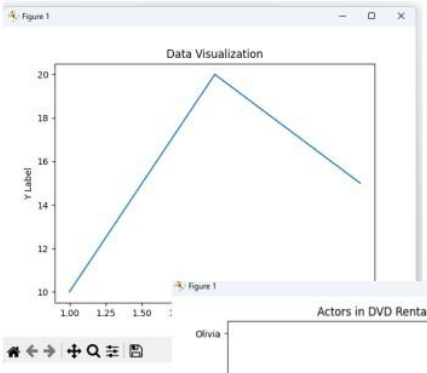
- **Запуск Flask-додатка. Під час запуску програми,** створюється екземпляр класу `Flask`.
  Визначаються маршрути (URL), які оброблятимуть запити.
- **Визначення маршрутів та обробка запитів**:
  Використовуються декоратори (`@app.route`) для визначення маршрутів.
  Виконуються функції, які обробляють ці маршрути.
- **Зв'язок з HTML**:

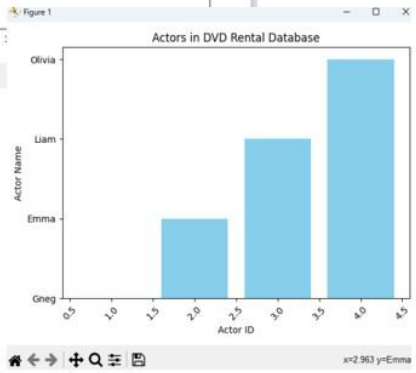
Для рендерингу HTML-сторінок використовується функцію `render_template`, яка завантажує HTML-файли з директорії програми.

- **Запуск сервера**:
  Запускається сервер Flask, використовуючи метод `app.run()`.
  Сервер слухає вхідні HTTP-запити.
- **Обробка користувацьких запитів**:
  Коли користувач надсилає запит (наприклад, відвідує сторінку чи заповнює форму), запит обробляється визначеним маршрутом.
  Маршрутна функція виконує відповідні дії (наприклад, обробка даних форми, взаємодія з базою даних).
  Результат (наприклад, HTML-сторінка) повертається користувачу у вигляді HTTP-відповіді.

Демонстрація використання програми







ВИСНОВКИ

У цій роботі було проаналізовано процес розробки та впровадження сервісу для збору і аналізу великих даних з різних джерел. Цей процес є важливим для бізнесу, науки, медицини та інших галузей, оскільки дозволяє отримати цінні дані та приймати обґрунтовані рішення. Розроблений сервіс дозволяє автоматизувати процес збору, обробки та аналізу даних, що дозволяє зробити прийняття рішень більш обгрунтованим і швидким.

У підсумку, розробка сервісу для збору і аналізу великих даних відкриває нові можливості для управління даними та прийняття стратегічних рішень, а також має великий потенціал у різних галузях, включаючи фінанси, маркетинг, медицину, науку та інші. Це важлива складова успішного функціонування сучасних організацій та допомагає відповідати викликам цифрової епохи.