НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

УДК 004.9:631.559:633

«ПОГОДЖЕНО»	**«ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ»**
Декан факультету інформаційних технологій	Завідувач кафедри комп'ютерних наук
Глазунова О.Г., д.п.н., професор	Голуб Б.Л., к.т.н., доцент
_____	_____
____ _____ 2023 р.	**__6__ ____листопада_____ 2023р.**

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему _____Інформаційно-аналітична система прогнозування врожайності сільськогосподарських культур_____

_____

Спеціальність ___122_____Комп'ютерні науки_____

_____
(код і назва)

Освітня програма __Комп'ютерний еколого-економічний моніторинг___

_____
(назва)

Орієнтація освітньої програми _освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми

_____д.т.н.,професор_____ _____ ____Семко В.В.___
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник магістерської кваліфікаційної роботи

_к.е.н., старший викладач ____ _____ _____Густера О.М._____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ **_____Мокан О.А._____**
(підпис) (ПІБ студента)

КИЇВ-2023

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) _____

ЗАТВЕРДЖУЮ
Завідувач кафедри_____
_____
_____ _____ _____
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"_29_" _____грудня_____2022 року

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ

_____
(прізвище, ім'я, по батькові)
Спеціальність _____122 Комп'ютерні науки _____
(код і назва)
Освітня програма ___Комп'ютерний еколого-економічний моніторинг _____
(назва)
Орієнтація освітньої програми_____освітньо-професійна _____
(освітньо-професійна або освітньо-наукова)
Тема магістерської кваліфікаційної роботи____Інформаційно-аналітична система прогнозування врожайності сільськогосподарських культур _____
_____
затверджена наказом ректора НУБіП України від "_29_"__грудня__2022р. №1917 –«С» ____

Термін подання завершеної роботи на кафедру _____2023 6 11_____
(рік, місяць, число)
Вихідні дані до магістерської кваліфікаційної роботи_Ukrstat_Data.gov.ua,__мережеві ресурси_____
_____
_____
_____
_____
Перелік питань, що підлягають дослідженню:
1. Системний аналіз предметної області _____
2.Моделювання системи _____
3.Розробка системи_____
4.Результати дослідження_____
Перелік графічного матеріалу (за потреби) _____
_____

Дата видачі завдання "__29__"_____грудня_____2022 р.

Керівник магістерської кваліфікаційної роботи _____Густера О.М._____
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ _____Мокан О.А._____
(підпис) (прізвище та ініціали студента)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

1. IAC- Інформаційно-аналітична система
2. Рис.- Рисунок
3. pH- Potential of hydrogen
4. HTML- HyperText Markup Language
5. CSS- Cascading Style Sheets
6. ТЗ- Технічне завдання
7. БД- База даних
8. UML- Unified Modeling Language
9. ОБД- Оперативна база даних
10. СД- Сховище даних
11. SSAS- SQL Server Analysis Services
12. SSIS- SQLServer Integration Services
13. OLAP- Online Analytical Processing
14. VS Code- Visual Studio Code
15. HTTP- Hypertext Transfer Protocol
16. SQL-Structured Query Language
17. СУБД- Система управління базами даних
18. ОС- Операційна система
19. REST- Representational State Transfer
20. MSE- Mean squared error
21. MAE- Mean absolute error
22. ПК- Персональний комп’ютер

ВСТУП

Сільське господарство – галузь господарства, завданням якої є забезпечення населення продовольством і отримання сировини для цілого ряду галузей промисловості. Україна має величезний потенціал для розвитку сільського господарства. Це відбувається, головним чином, через сприятливі природні умови для сільського господарства: родючий ґрунт і дуже сприятливий клімат на більшій частині території країни. Сільське господарство України є досить перспективною галуззю та одним із лідерів експорту продукції рослинництва та тваринництва на світових ринках. Окрім того, сільське господарство є основною рушійною силою для розвитку економіки країни та забезпечення добробуту населення. У сільському господарстві присутні певні досягнення за останні декілька років, але, незважаючи на це, в Україні ще багато питань є невирішеними.

Прогнозування врожайності є важливою задачею для аграрного сектора, оскільки воно дозволяє сільським господарствам та владі ефективно планувати виробництво, забезпечувати стабільність на ринку продукції та враховувати ризики, пов'язані з природними умовами та кліматичними змінами. Інформаційно-аналітичні системи (ІАС) стали потужним інструментом для збору, обробки та аналізу даних, які допомагають у точному прогнозуванні врожайності та визначенні оптимальних стратегій управління сільськогосподарським виробництвом.

У зв'язку з цим актуальність дослідження полягає в необхідності створення ІАС, яка дозволить прогнозувати врожайність сільськогосподарських культур з високою точністю та реагувати на можливі ризики та зміни у вирощуванні культур. Це особливо актуально в умовах сучасних глобальних викликів, таких як зміна клімату, зростання населення та зменшення доступної сільськогосподарської площі.

Предметом дослідження є розробка та впровадження інформаційно-аналітичної системи прогнозування врожайності сільськогосподарських культур.

Об'єктом дослідження є процес вирощування сільськогосподарських культур та фактори, які впливають на їх врожайність.

Метою дослідження є розробка, впровадження та оцінка інформаційно-аналітичної системи для прогнозування врожайності сільськогосподарських культур з використанням сучасних методів аналізу та обробки даних, з метою підвищення стабільності та ефективності аграрного виробництва.

Завданням для виконання роботи було поставлено провести аналіз наукової та технічної літератури з метою визначення сучасних методів та підходів до прогнозування врожайності, сформулювати технічні вимоги до розробки системи. Наступним кроком є збір необхідних даних для проведення прогнозу, таких як кліматичні умови, властивості ґрунту та врожайність культур за минулі роки. Для якісного аналізу необхідно обрати інструментарій за допомогою якого можна оброблювати дані. Після чого необхідно реалізувати алгоритми для обробки та аналізу даних. З отриманих результатів можна зробити висновки, і оцінити результати роботи системи за точністю прогнозу.

1. СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис основних аспектів предметної області

Сільське господарство є однією з найважливіших галузей для забезпечення харчової безпеки та сталого розвитку нашого суспільства. Сільськогосподарські культури є необхідними для вирощування їжі та сировини для промисловості, а також вони мають значний вплив на економіку та здоров'я населення. Однак в умовах зростаючого населення планети та зміни клімату, завдання сільськогосподарського виробництва стає все складнішим і вимагає більш точних та передбачуваних рішень[1]. Важливим аспектом сільського господарства є прогнозування врожайності сільськогосподарських культур. Ця задача має велике значення для ефективного управління господарством, виробництва продуктів харчування та запобігання голоду в світі. Недостатній прогноз може призвести до втрати врожаю, надмірної витрати ресурсів або навіть до продовольчої кризи.

Прогнозування врожайності сільськогосподарських культур є складним завданням через багатофакторність і незрозумілість природних та антропогенних процесів, які впливають на цей процес. Воно враховує в собі ряд важливих аспектів, таких як кліматичні умови, стан ґрунту, застосування добрив, технології обробки і поливу, а також даних по врожайності за минулі роки, які можуть впливати на урожайність[2].

Для досягнення більш точних та надійних прогнозів врожайності сільськогосподарських культур, використовуються інформаційно-аналітичні системи, які базуються на аналізі великої кількості даних з різних джерел, таких як дані з сільськогосподарських ділянок та інші. Ці системи допомагають покращити точність та довірчість прогнозів, що, в свою чергу, сприяє підвищенню продуктивності сільського господарства та забезпеченню стабільності у галузі харчування.

Здатність прогнозувати врожайність є важливою складовою сучасного сільського господарства та глобального харчового ланцюга. В сучасному світі,

точні, надійні та науково обґрунтовані прогнози стають критично важливими для забезпечення продовольчої безпеки та сталого розвитку. Отже, розглянувши основні аспекти предметної області, включаючи важливість сільського господарства, проблеми прогнозування врожайності, а також роль інформаційно-аналітичних систем у покращенні прогнозів. Подальші дослідження та розвиток таких систем можуть сприяти підвищенню продуктивності та стійкості сільського господарства, а також сприяти боротьбі з голодом та забезпеченню харчової безпеки на міжнародному рівні.

1.2 Визначення ключових факторів, що впливають на врожайність

Врожайність сільськогосподарських культур є однією з найважливіших характеристик у сільському господарстві, і вона безперечно впливає на якість та кількість продукції, що надходить на ринок. У світлі постійних змін у кліматі, зростаючого попиту на продукти харчування та необхідності забезпечення стабільного виробництва сільськогосподарських культур, вивчення факторів, що впливають на врожайність, є надзвичайно актуальним завданням. Врожайність є результатом впливу численних факторів, які взаємодіють між собою. Для забезпечення точних та надійних прогнозів врожайності необхідно ретельно досліджувати та аналізувати ці фактори[3]. Далі будуть розглянуті основні ключові фактори, які впливають на врожайність сільськогосподарських культур. Основні фактори, що впливають на врожайність, включають:

1. Кліматичні умови: Кліматичні умови є одним з найважливіших факторів, які впливають на врожайність сільськогосподарських культур. Клімат охоплює широкий спектр параметрів, таких як температура, опади, вологість, та інші метеорологічні та атмосферні умови, які можуть варіюватися в залежності від регіону та сезону[4]. Ось деякі конкретні аспекти впливу клімату на врожайність культур:

- Температура: Температурні умови грають важливу роль у рості і розвитку рослин. Різкі коливання температури, особливо під час цвітіння та плодоношення, можуть впливати на опилення та формування плодів. Екстремально високі або низькі температури можуть призвести до стресу рослин та зменшення врожайності.
- Опади: Кількість та розподіл опадів грають критичну роль у вирощуванні сільськогосподарських культур. Недостатність опадів або їхній надмір можуть спричинити посухи або затоплення, що впливає на ріст та розвиток рослин. Нерівномірний розподіл опадів також може впливати на нерівномірний розквіт та дозрівання плодів.

2. Ґрунт: Якість ґрунту та його фізичні та хімічні властивості мають вирішальне значення для здоров'я та розвитку рослин[5]. Ось деякі чинники які впливають на якість ґрунту:
    - Структура ґрунту- сприяє гарному проникненню кореневої системи рослин, доступу до води та повітря. Клейкі ґрунти можуть утворювати проблеми з дренажем, тоді як сипкі ґрунти можуть не утримувати вологу.
    - pH- ґрунту важливий, оскільки він впливає на доступність мінералів для рослин. Занадто кислий або лужний pH може обмежити засвоєння деяких поживних речовин.
    - Плодючість залежить від наявності поживних речовин, таких як азот, фосфор, калій та мікроелементи. Ґрунти з низьким рівнем плодючості можуть потребувати додаткового внесення добрив для підтримки росту рослин.
    - Гумус- це розкладені органічні матеріали, які зберігаються в ґрунті і служать джерелом поживних речовин та вологи для рослин.

- Вологість ґрунту- ґрунт здатний зберігати воду, яка доступна для кореневої системи рослин. Запаси води можуть визначити, наскільки довго культура може виживати без опадів або поливу.

3. Агрохімічні параметри: Використання добрив, пестицидів та інших агрохімікатів також впливає на якість та кількість врожаю. До агрохімічних параметрів відносяться:
    - Азот (N): є ключовим елементом для росту рослин, і він необхідний для синтезу білків, ДНК та інших біологічно важливих сполук. Брак азоту може призвести до відстаючого росту та низької врожайності.
    - Фосфор (P): необхідний для розвитку кореневої системи, квітіння та плодоношення рослин. Він впливає на передачу енергії в рослині та генеруванню її ДНК.
    - Калій (K): впливає на фотосинтез, регулює водний баланс та зміцнює стійкість рослин до стресових умов.
    - Мікроелементи, такі як залізо (Fe), марганець (Mn), мідь (Cu), цинк (Zn) і бор (B), необхідні рослинам в невеликих кількостях, але вони важливі для багатьох біохімічних процесів. Недостатність мікроелементів може спричинити хвороби та дегенерацію рослин.

    Виміряти ці агрохімічні параметри можна за допомогою лабораторних аналізів ґрунту. Результати аналізу дозволяють сільськогосподарським виробникам визначити, які добрива або агротехнічні заходи потрібні для оптимального забезпечення рослин поживними речовинами та іншими необхідними елементами. Досягнення балансу агрохімічних параметрів у ґрунті є важливою складовою успішного вирощування сільськогосподарських культур та досягнення високої врожайності.

Розуміння та керування цими параметрами є важливими завданнями для сільськогосподарських виробників, оскільки вони безпосередньо впливають на врожайність та якість продукції. Забезпечення оптимальних умов для росту та розвитку рослин, є важливою складовою успішного вирощування культур та досягнення високої врожайності. Аналіз та налагодження цих показників допомагають сільським господарям збалансувати виробництво та зробити його більш продуктивним та стійким до стресових умов.

1.3 Аналіз вимог до програмної та апаратної системи

У цьому розділі буде розглянути основні вимоги до програмної та апаратної системи. Буде розглянуто функціональні та технічні аспекти, які дозволять забезпечити ефективну та надійну роботу системи.

Вимоги до програмної системи включають в себе ряд інструментів для обох компонентів - серверної та клієнтської. Ця потреба обумовлена використанням клієнт-серверної архітектури, яка була використана під час розробки аналітичної системи.

Система повинна забезпечувати можливість обробки даних про кліматичні умови, ґрунт , врожайність за минулі роки, які безпосередньо впливають на кількість зібраного врожаю. Також у системі повинні бути реалізовані алгоритми, які дозволяють проводити прогнозування на основі зібраних даних. При побудові прогнозу необхідно використовувати методи статистичного аналізу, такі як часові ряди, та лінійна регресія. Результати отримані в ході роботи системи повинні бути доступні для користувача у зрозумілому для нього вигляді, включаючи графіки та інші способи візуалізації даних.

Розглядаючи технічні вимоги до серверної частини, стає зрозуміло що потрібен веб- сервер для розміщення сайту та обробки запитів користувача. Також для зберігання та обробки даних потрібна база даних, котра забезпечить ефективний доступ до інформації. Для побудови сайту необхідно буде використовувати спеціальні технології, такі як HTML, CSS, JavaScript, а також

мову програмування Python, а також спеціальні бібліотеки та інструменти, за допомогою яких будуть реалізовані аналітичні алгоритми.

Необхідно також зауважити, що веб-сайт повинен мати інтуїтивно зрозумілий інтерфейс, який буде зручним та зрозумілим для користувача, зручне розташування елементів. Не менш важливо важливим є те, що користувач повинен мати можливість отримати доступ до ресурсу на різних пристроях, та операційних системах, включаючи смартфони та планшети.

Користувачу для роботи з системою необхідно мати пристрій, на якому буде встановлено сучасний веб-браузер, стабільне інтернет-з'єднання з мінімальною швидкістю не менше 1 Мбіт/с для нормальної передачі даних.[6] Практично всі обчислення та обробка даних відбуваються на сервері, тому для користувача не обов'язково мати потужний процесор. Практично будь-який процесор, який підтримує сучасні операційні системи, буде достатнім. Проблеми, які можуть виникнути, скоріше пов'язані зі швидкістю встановлення та запуску веб-браузера, а не з відправленням запитів до сервера для отримання вмісту веб-сайту.

1.4 Аналіз наявних рішень

Сільське господарство має перед собою широкий спектр завдань, пов'язаних із передбаченням врожайності та керуванням сільськогосподарськими процесами. Існують різноманітні програмні рішення, спрямовані на сприяння фермерам та агрономам у вирішенні цих завдань.

Метою аналізу є з'ясування переваг і недоліків наявних рішень, виявлення можливостей для покращення та визначення ніш для нової інформаційно-аналітичної системи, яка розробляється у рамках цієї дипломної роботи.

Для аналізу наявних програмних рішень у сфері прогнозування врожайності сільськогосподарських культур розглянемо конкретні приклади:

Cropio - це інтернет-платформа для сільськогосподарського моніторингу та оцінки врожайності. Вона надає можливості відстежувати кліматичні умови,

вологість ґрунту, ріст рослин та інші фактори, що впливають на врожай. Cropio Програма використовує дані супутникового зондування та розумні алгоритми для створення прогнозів. Серед плюсів можна виділити, що програма має інтуїтивно зрозумілий та дружній інтерфейс, що спрощує роботу з нею. Користувачі можуть легко завантажувати дані та отримувати звіти без великих зусиль. Додаток аналізує історію врожайності, оцінює її та встановлює причину втрати врожаю. З недоліків можна виділити, що деякі фермери відзначають високу вартість, що робить застосунок менш доступним для деяких господарств.

Варто також зазначити про співпрацю з іншими компаніями, такими як Syngenta. Співпраця між Cropio та Syngenta створює комплексний підхід до сільськогосподарського управління, який поєднує аналітичні можливості Cropio з агрохімічними та науковими рішеннями Syngenta. Це дозволяє фермерам ефективно вирощувати рослини та мінімізувати ризики для їхнього врожаю, забезпечуючи стабільний та високий врожай, що відповідає потребам ринку та споживачів.

FarmLogs - це онлайн-платформа для моніторингу сільськогосподарських полів та прогнозування врожайності. Вона надає фермерам засоби для відстеження вологості ґрунту, росту рослин, кліматичних умов та інших параметрів. FarmLogs також надає аналітичні інструменти для покращення прийняття рішень. З переваг даного продукту можна відзначити те, що надається доволі детальна інформація про кліматичні умови, також користувачі можуть переглядати детальний аналіз вологості ґрунту , росту рослин та іншого. Також слід відзначити що у застосунку доступна пробна версія, що дає можливість спробувати його, навчитись користуватись, та зрозуміти, чи можна за допомогою функціоналу додатку отримати той результат який

Недоліками даного продукту є те, що він не завжди доступний в масштабі всієї країни, тобто застосунок має обмежену область покриття, та має високу вартість.

Climate FieldView - це програма, розроблена компанією The Climate Corporation, яка належить Monsanto (тепер Bayer). Вона допомагає фермерам

збирати та аналізувати дані про поля, включаючи інформацію про вологість ґрунту, глибину сіяння, розміщення рослин, індекси листя та інші параметри. Програма також надає можливості створення мап врожайності та прогнозів, також дозволяє швидко і легко без будь-яких перешкод збирати, зберігати і візуалізувати важливі дані про поля.

Climate FieldView активно співпрацює з іншими компаніями та партнерами, які надають продукти та послуги для сільського господарства. Ця співпраця дозволяє фермерам отримувати доступ до більш широкого спектру інструментів та рішень, може легко інтегруватися з іншими сільськогосподарськими системами та обладнанням, включаючи сільськогосподарські машини та агрохімічні засоби, що дає можливість вносити дані відразу до системи, та отримувати всю інформацію в застосунку. Ця співпраця дозволяє фермерам отримувати доступ до інтегрованих рішень та розширює можливості для покращення управління господарствами та збільшення врожайності.

Серед мінусів даного рішення можна відзначити те, що обладнання за допомогою якого проводяться виміри, може бути не сумісне з деякими видами техніки, також слід відзначити що застосунок є доволі складним для освоєння, та потребує чималих зусиль, що в купі з високою ціною робить його не досить привабливим для деяких фермерів.

Розглянувши вже створенні системи, можна зробити висновок що вони є досить потужними інструментами, проте слід зазначити, що можна використовувати для аналізу і прогнозування інструмент для аналітики від Microsoft PowerBi.

Power BI - це бізнес-аналітичний інструмент від Microsoft, який дозволяє візуалізувати та аналізувати дані з різних джерел для прийняття обґрунтованих бізнес-рішень. Цей інструмент може бути корисним і в сільському господарстві, зокрема для аналізу врожаю і прогнозування. Користувач може підключати різні джерела даних, такі як бази даних, таблиці Excel, сховища даних у хмарі, сенсори та інше. Фермери можуть імпортувати дані про врожайність, погоду, вологість

ґрунту та інші параметри для подальшого аналізу. Завантажені дані можна візуалізувати, будувати графіки, діаграми та багато іншого. Також можна побудувати прогноз врожаю беручи лише за основу історичні дані, про такий прогноз не буде враховувати багато інших факторів, тому не буде мати високу достовірність.

Серед мінусів можна відзначити високу трудомісткість та складність. Порівнюючи з вже існуючими ІАС така робота займе набагато більше часу, та не дасть такої точності в прогнозі.

На сьогоднішній день існує широкий спектр рішень для прогнозування врожайності сільськогосподарських культур. Вибір конкретного рішення залежить від таких факторів, як масштаб прогнозу, точність прогнозу, доступність даних, тощо.

1.5 Постановка завдання

Після ретельного огляду та докладного опису предметної області системи, вивчення програмних та технічних вимог і порівняння їх із наявними рішеннями настав час сформувати завдання для технічного фахівця, який візьме на себе розробку цієї системи. Для цієї мети використовується технічне завдання, яке визначається як інструмент для передачі програмістові конкретного уявлення про те, яким має бути вигляд системи, яку необхідно створити.

Технічне завдання (ТЗ) - це документ, який описує вимоги до системи або продукту з точки зору технічного аспекту. ТЗ є обов'язковим документом при розробці програмного забезпечення, оскільки воно забезпечує:

- Ясність і розуміння вимог до системи - дозволяє всім учасникам розробки, включаючи замовника, розробників, тестувальників і менеджерів, мати єдине розуміння того, що повинна робити система.
- Контроль якості - основою для проведення тестування системи. Тестування дозволяє перевірити, чи система відповідає вимогам, викладеним у ТЗ.

З’ясувавши для чого потрібно ТЗ, почнемо розгляд першого етапу розробки- технічні аспекти. На цьому етапі буде описано, як виглядатиме система, з яких модулів буде складатись, які категорії користувачів будуть задіяні у системі, та які функції будуть виконуватись у кожному модулі.

Система прогнозування складається з двох модулів:

Модуль прогнозування забезпечує прогнозування врожайності сільськогосподарських культур на основі статистичних методів навчання. Дозволяє користувачу вводити дані про культури, локацію поля, та інформацію про ґрунт, на основі яких буде будуватись прогноз. Модуль відображення графіків забезпечує візуалізацію прогнозів у вигляді графіків, дозволяє користувачу переглянути аналітичну інформацію по врожайності культур, кліматичних умовах у різних містах за певний проміжок часу.

Розглянувши модулі які повинні бути представлені у системі, можна перейти до огляду їх функціональності. У першому модулі відбувається процес прогнозування врожайності сільськогосподарської культури. Користувач обирає з випадаючого списку назву культури, регіон та місто, та вводить площу свого поля. Після обрання міста йому можна вивести показники якості ґрунту у обраному регіоні. Другий модуль призначений для відображення статистики та діаграм. Необхідно надати можливість користувачеві переглядати графік по врожайності культур за минулі роки, кількість опадів за регіонами, та графік по характеристикам ґрунту.

Зрозумівши функціональні вимоги , наступним кроком є розробка бази даних (БД), яка повинна буде зберігати дані необхідні для аналізу та прогнозування. Насамперед повинні зберігатись дані про історичні значення врожайності культур, значення кліматичних умов, вологості, середньої температури, кількості осадків, та характеристик ґрунтів. БД повинна бути добре організована, щоб її було легко використовувати. Таблиці повинні бути пов'язані між собою, щоб можна було легко отримувати доступ до потрібної інформації. Також вона повинна бути захищена від несанкціонованого доступу.

Дані, які зберігаються в БД, є цінними і можуть бути використані для незаконних цілей.

Наступним етапом у розробці системи є проведення прогнозування за допомогою статистичних методів. Після того як були знайдені дані та внесені у БД, необхідно реалізувати методи за допомогою програмування. Існує багато бібліотек та фреймворків які розробник може використовувати при реалізаціє даних методів. Обов’язковим пунктом є відображення точності прогнозу. Точність прогнозу - це міра того, наскільки добре прогноз відповідає фактичним значенням. Точність прогнозу можна оцінити за допомогою різних методів, які вимірюють відстань між прогнозованими значеннями і фактичними значеннями.

В результаті виконання проекту очікується отримати інформаційно-аналітичну систему, яка здатна надавати точні та надійні прогнози врожайності сільськогосподарських культур на основі зібраних даних та аналізу. Система має сприяти підвищенню ефективності сільського господарства та допомагати фермерам приймати обґрунтовані рішення щодо вирощування рослин.

2. МОДЕЛЮВАННЯ СИСТЕМИ

2.1. Діаграма прецедентів

Засоби UML (Unified Modeling Language) корисні через свою спроможність створювати графічні моделі системи, що робить процес розробки більш зрозумілим та ефективним. Основні переваги використання UML включають уніфікацію комунікації, можливість візуалізації складних систем, аналіз та проектування, можливість створення документації, можливість рефакторингу та оптимізації системи, управління проектом та спільне розуміння бізнес-процесів. Використання UML допомагає покращити якість та ефективність розробки програмних продуктів і полегшує спілкування між учасниками проекту.

Діаграма прецедентів в UML є інструментом, який використовується для відображення взаємозв'язків між акторами та прецедентами у системі. Іноді її називають також діаграмою варіантів використання.[7] Головною метою діаграми прецедентів є збір вимог до функціональності системи та визначення, як користувачі будуть взаємодіяти з нею.

Складові діаграми прецедентів:

- Актори- це зовнішні об'єкти, які взаємодіють з даною системою. Це може бути людина, інша система або організація.
- Прецеденти(випадок використання, дія)- це набори дій, які система виконує в відповідь на запит актора.
- Зв'язки(прямий, include – включення, extend - розширення)- це відношення між акторами та прецедентами. Існує три типи зв'язків: Прямий зв'язок - це основне відношення між актором та прецедентом. Включення - це відношення, при якому один прецедент включає в себе інший. Розширення - це відношення, при якому один прецедент розширює інший.

Для ІАС прогнозування сільськогосподарських культур була побудована діаграма прецедентів, яка представлена на рис.2.1, на ній зображено основні актори:

1. Оператор- вносить дані про якість ґрунту, кліматичні умови, та показники сільськогосподарських культур.
2. Аналітик- займається аналізом даних, на основі яких формує звіт.
3. Фермер- той, для кого і працює система, на основі створеного звіту, приймає рішення.

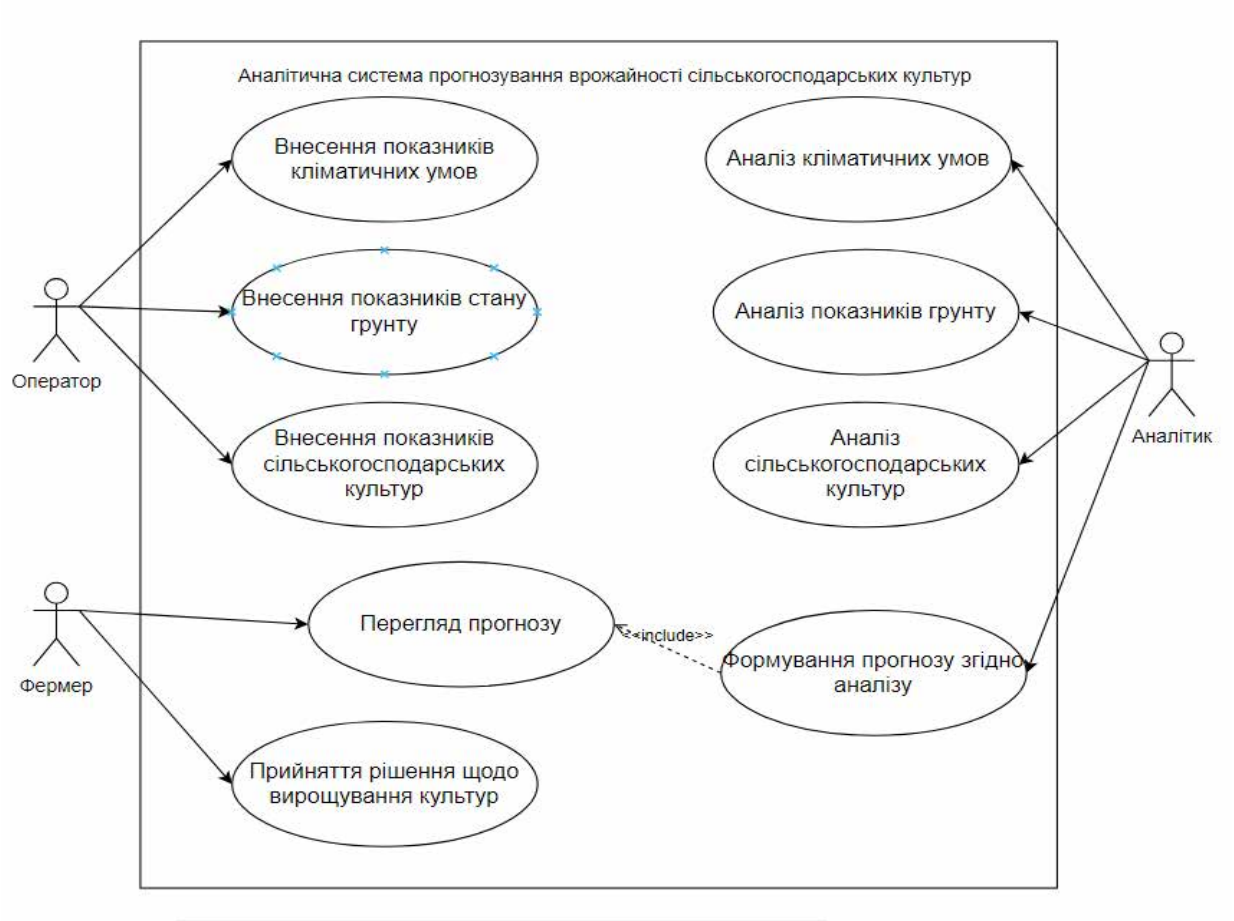

Рис. 2.1 Діаграма прецедентів

Розглянемо один із прецедентів ближче:

Назва прецеденту: «Формування прогнозу»

Мета прецеденту: Аналіз всіх факторів що впливають на врожайність.

Оптимістичний сценарій:

А. Оператор заніс всі необхідні дані в систему.

Б. Аналітик проводить аналіз цих даних.

В. На основі оброблених даних формує прогноз.

Г. Відправляє отриманий результат фермеру.

Прагматичний сценарій:

Умова 1. Для однієї з культур немає історичних даних.

Умова 2. Для одного з регіонів немає значень кліматичних умов.

Умова 3. Для одного з регіонів немає значень показників ґрунту.

2.2.Архітектура системи

Система обрана з клієнт-серверною архітектурою з міркувань ефективності та відповідності певним вимогам.

Клієнт-серверна архітектура є однією з найпоширеніших архітектурних парадигм у світі програмного забезпечення. Клієнти та сервери взаємодіють один з одним за допомогою мережі. Вони надсилають запити до серверів, а сервери відповідають на запити. Ця архітектурна модель розподілених систем полягає у розділенні програми на дві головні компоненти: клієнт і сервер[8].

Клієнт - це програма або пристрій, який надсилає запити до сервера для отримання певних послуг або даних. Він взаємодіє з користувачем і відповідає за відображення інформації та введення користувача. Клієнт може бути реалізований на різних платформах, таких як комп'ютери, смартфони, планшети тощо.

Сервер - це програма або обладнання, яке обробляє запити від клієнтів і надає їм необхідні послуги або дані. Сервер зазвичай працює у фоновому режимі, обробляючи багато запитів одночасно та забезпечуючи ефективний доступ до ресурсів.

Ця архітектура надає кілька важливих переваг:

1. Віддалений доступ: Завдяки цій архітектурі, користувачі можуть мати віддалений доступ до системи. Це особливо важливо для фермерів та аналітиків, які можуть працювати з системою з будь-якого місця, де є Інтернет.

2. Одночасна робота багатьох користувачів: Клієнт-серверна архітектура дозволяє обробляти запити великої кількості користувачів одночасно. Це важливо, оскільки система може бути використана багатьма фермерами та аналітиками одночасно.
3. Масштабованість: Система може бути легко масштабована за потреби. Додавання нових серверів або ресурсів може покращити продуктивність та відповідати зростаючим потребам користувачів.
4. Модульне оновлення програмного забезпечення: Клієнт-серверна архітектура дозволяє оновлювати окремі компоненти системи без перерви у роботі всієї системи. Це забезпечує зручну та безпечну процедуру поновлення.

Проте клієнт-серверна архітектура має і деякі недоліки, зокрема:

1. Вартість - клієнт-серверна архітектура може бути дорожчою, ніж інші архітектури, оскільки вимагає додаткових серверів та мережевого обладнання.
2. Складність - клієнт-серверна архітектура може бути складнішою для розробки та обслуговування, ніж інші архітектури.

   На рис.2.2 зображено архітектуру системи, вона має такі вузли:

   1. Робочі станції аналітика, оператора та фермера: Ці компоненти представляють робочі місця користувачів, які взаємодіють з системою.
   2. Два сервера з базою даних та сховищем даних: Ці сервери відповідають за зберігання та обробку інформації. База даних містить структуровану інформацію, а сховище даних може містити неструктуровану інформацію.

   Зв'язки між компонентами:

   - Робоча станція оператора підключена до сервера за допомогою мережі. Оператор використовує робочу станцію для внесення даних до сервера.

- Сервер підключений до робочої станції аналітика за допомогою мережі. Аналітик використовує робочу станцію для перегляду прогнозів врожайності та генерування звітів.

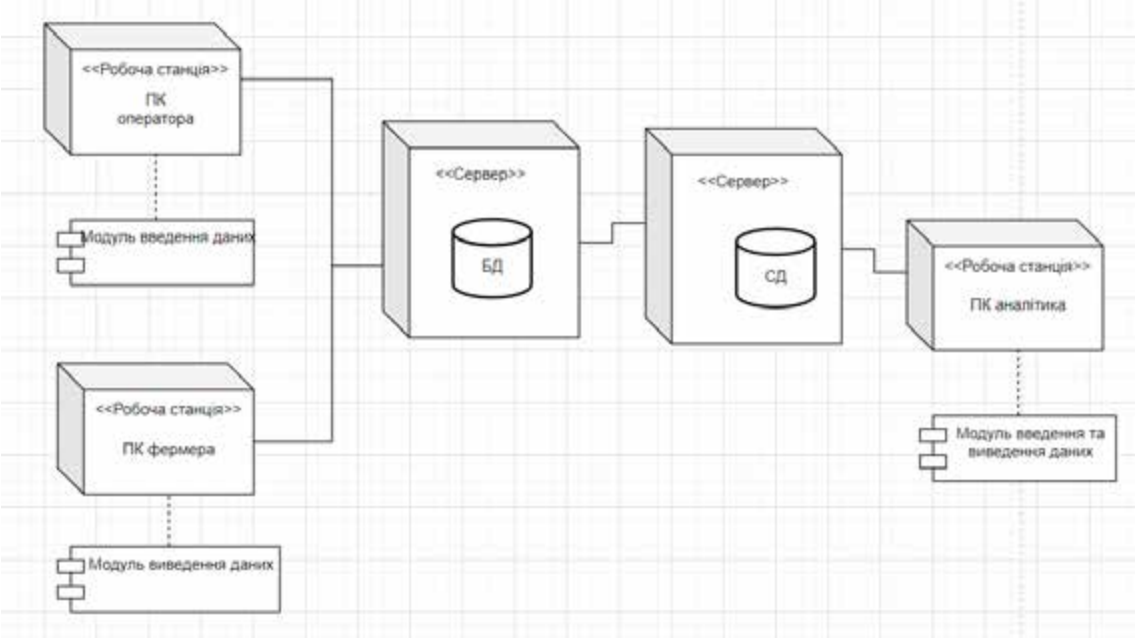

Рис. 2.2 Топологія системи

2.3 Опис джерела даних

Важливим кроком для реалізації системи є розробка структури оперативної бази даних, та наповнення її даними.

Оперативна база даних (ОБД) є важливою частиною інформаційної системи та має на меті зберігання, постійне оновлення та забезпечення доступу до даних, які є використовуваними в щоденній роботі організації. ОБД володіє низкою ключових характеристик та функцій. Вона функціонує для зберігання різноманітних видів інформації, таких як кліматичні дані, історична врожайність, агротехнічні параметри, характеристики ґрунтів та інші суттєві інформаційні ресурси. Оперативна база даних регулярно оновлюється за допомогою автоматизованих систем, датчиків та інших джерел, щоб завжди мати актуальну інформацію. В ній об'єднуються дані з різних джерел з метою

створення комплексних наборів даних, які можна використовувати для подальшого аналізу.[9]

Для розроблюваної системи джерелом даних є офіційні статистичні джерела України. Офіційні статистичні джерела є важливим джерелом даних для ОБД в системі прогнозування врожайності сільськогосподарських культур.

Офіційні статистичні джерела, такі як урядові агентства та статистичні офіси, забезпечують цінну інформацію про сільське господарство та врожайність. Їхні завдання включають:

- Збір та обробка даних про виробництво сільськогосподарських культур.
- Моніторинг площі під посівами та їхнього стану.
- Реєстрація внесення добрив та засобів захисту рослин.
- Формування статистики щодо врожайності та урожайності культур.

Офіційні статистичні джерела проводять систематичний збір даних від сільгоспвиробників, аграрних підприємств, сільськогосподарських кооперативів та інших джерел. Ці дані охоплюють різні аспекти сільського господарства, включаючи виробництво, зберігання та транспортування сільськогосподарської продукції. Після збору дані піддаються обробці та агрегації, щоб створити комплексні набори даних, які можна використовувати для аналізу та прогнозування врожайності.

Україна має свої офіційні статистичні джерела, такі як Державна служба статистики України[10]. Дані з цього джерела використовуються для аналізу та прогнозування сільськогосподарської діяльності в Україні, та зокрема для виконання прогнозування у ході цього проекту.

На рисунку 2.3 представлена структура оперативної бази даних для розроблюваної системи:

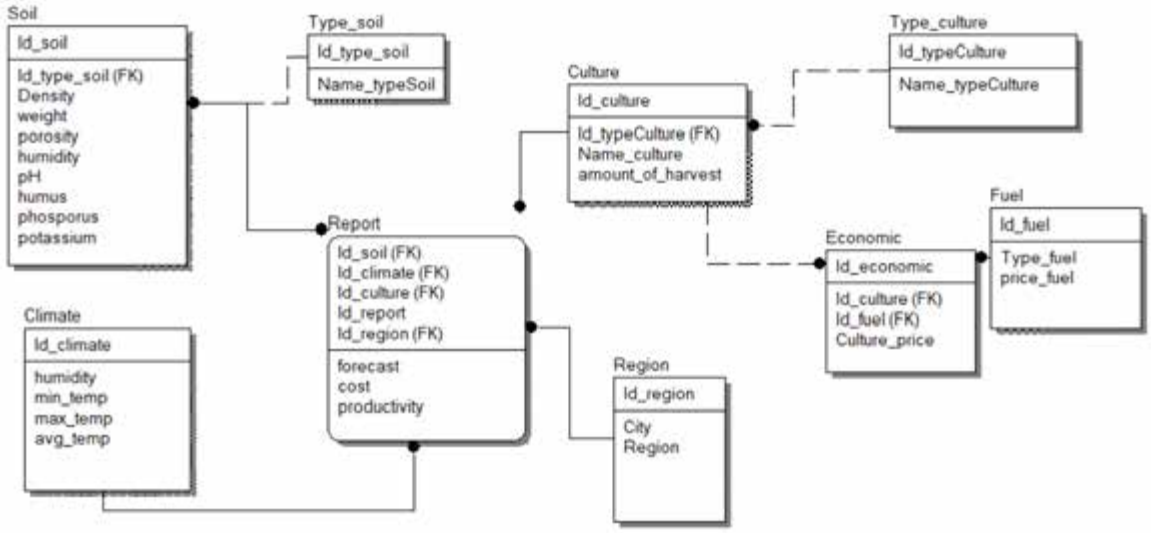

Рис. 2.3 Оперативна база даних

Оперативна база даних складається з таких сутностей:

Soil- містить дані про ґрунт:

- Щільність ґрунту
- Вагу
- Пористість
- Вологість
- Значення pH
- Гумус
- Фосфор
- Калій

TypeSoil-містить дані про тип ґрунту:

- Тип ґрунту

Climate-містить дані про кліматичні умови

- Вологість
- Мінімальна температура
- Максимальна температура
- Середнє значення температури

Report – сутність що містить в собі інформацію про всі інші сутності

- Прогноз
- Прибуток
- Урожайність

Culture-містить інформацію про культуру

- Назва культури
- Кількість зібраної культури

TypeCulture- містить інформацію про тип культури

- Тип культури

Economic-містить у собі дані про ціну культури

- Ціна культури

Fuel-містить дані про паливо

- Тип палива
- Вартість палива

Region- містить дані про області України

- Місто
- Область

2.4. Структура джерела інформації для проведення інтелектуального аналізу

Для проведення повноцінного інтелектуального аналізу даних, необхідно дослідити що таке сховище даних, та дати відповідь на ряд питань: для чого воно потрібне у системі? Чим воно відрізняється від ОБД? З яких джерел надходять дані до СД? Відповіді на ці питання будуть розглянуті у цьому розділі. Після чого буде спроектоване сховище даних до розроблюваної системи.

Сховище даних (СД) - це система, спрямована на довгострокове зберігання інтегрованої інформації, з орієнтацією на конкретну предметну область та відображення різних поглядів на неї. Ця система рідко піддається змінам і підтримує хронологічну структуру даних, де дані організовані відповідно до основних аспектів діяльності підприємства чи організації, таких як замовники, продажі, склад і т. д.

Важливою особливістю СД є інтеграція інформації в єдиний формат, що полегшує аналіз та використання даних з різних джерел. Підтримка хронології означає, що інформація в СД систематизована за послідовними періодами часу, що сприяє аналізу динаміки подій та трендів.

У відмінність від оперативної бази даних, де дані організовані відповідно до процесів (наприклад, відвантаження товару або виписка рахунків), предметна організація даних в СД спрямована на спрощення аналізу та прискорення аналітичних обчислень. Сховища даних орієнтовані на бізнес-поняття та структуровані навколо ключових аспектів предметної області, а не на послідовностях бізнес-процесів.[11]

Дані, які надходять до сховища, можуть мати різні характеристики, такі як назви, формати, одиниці вимірювання та методи кодування, оскільки вони отримуються з різних джерел. Перед тим як зберігати ці дані в сховищі, вони проходять процес перевірки, відбору та нормалізації. Це включає в себе перетворення даних в єдиний формат, вид і кодування, а також обчислення сумарних показників, якщо це необхідно. Після цього дані представляються користувачеві як єдиний інформаційний простір, що значно спрощує їх аналіз.

Наприклад, якщо у чотирьох різних джерелах дані про код товару були представлені чотирма різними способами кодування, то в сховищі даних вони будуть перетворені до єдиної системи кодування для забезпечення єдиної структури та формату.

Незважаючи на те, що інформація для сховища даних отримується з OLTP-систем баз даних, це не веде до зайвого накопичення даних. Мінімізація надлишковості даних забезпечується тим, що вони спочатку очищуються і

проходять фільтрацію даних до сховища, в процесі якої вони очищаються від непотрібної інформації, яка не може бути використана у бізнес-аналізі.

На рисунку 2.4 представлена схема сховища даних для розроблюваної системи:

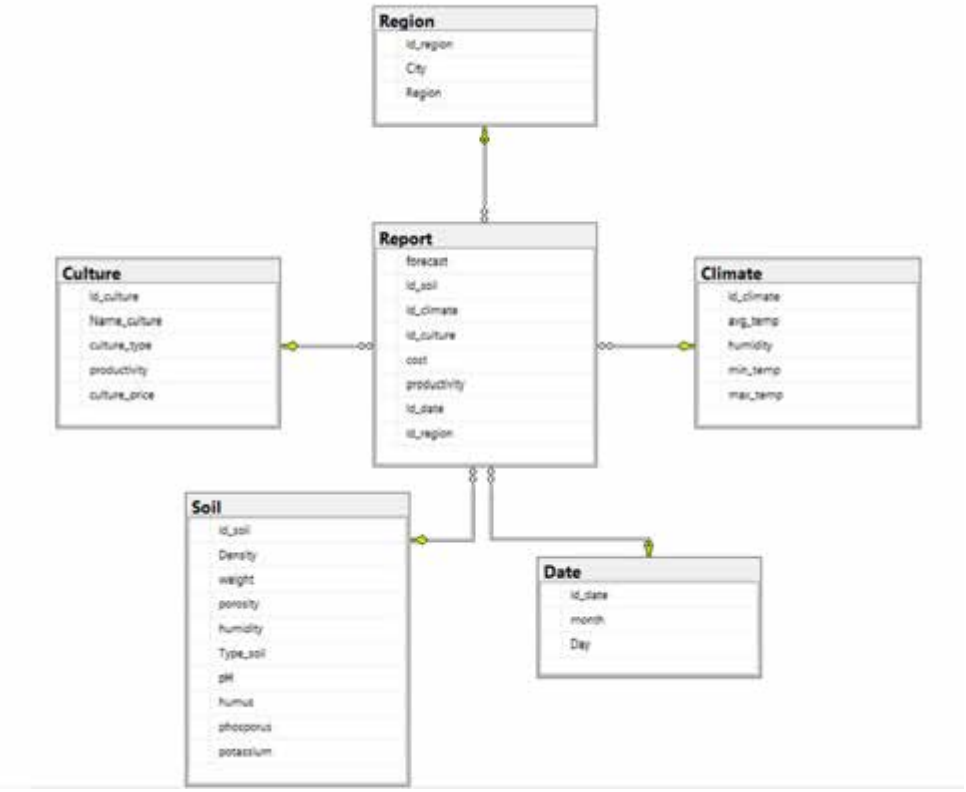

Рис. 2.4 Сховище даних

Опис структури:

Soil- вимір що зберігає дані про якість та характеристики ґрунту

- Id-soil- унікальне значення виміру
- Density- щільність ґрунту
- Weight- вага ґрунту
- Porosity-пористість ґрунту
- Humidity- вологість
- type soil- значення виміру типу ґрунту
- pH-значення вмісту pH
- Humus-значення вмісту гумусу

- Phosporus-значення вмісту фосфору
- Potassium-значення вмісту калію

Date- вимір що містить інформацію про дату

Culture- вимір що містить інформацію про культури

- Id culture- унікальне значення виміру
- Name culture – назва культури
- Culture_type-тип культури
- Productivity- значення врожайності за минулі роки
- Culture_price-ціна культури

Region- вимір що містить дані про області України

- Id_region- унікальне значення виміру
- City- назва міста
- Region- назва області

Report- таблиця фактів, містить інформацію з даними, для проведення аналізу та створення звіту

Climate-вимір що містить інформацію про кліматичні умови

- Id climate- унікальне значення виміру
- Avg_Temp- середнє значення температури повітря
- Humidity- вологість повітря
- Min_temp-мінімальна температура
- Max_temp-максимальна температура

Для того аби реалізувати процес наповнення СД даними, необхідно спочатку створити OLAP-куб, за допомогою служб сервісу SSAS.

OLAP-куб - це структура, що дозволяє швидкий та багатогранний аналіз даних. Також її можна описати як здатність до маніпулювання та дослідження

даних з різних точок зору. Використання кубів даних дозволяє подолати обмеження, які характерні для реляційних баз даних. Реляційні бази даних не завжди ефективні для швидкого аналізу та обробки великих обсягів інформації. Хоча існують інструменти для створення звітів на базі реляційних систем, проте вони можуть працювати досить повільно, коли потрібно опрацювати великі об'єми даних. За допомогою куба OLAP можна легко та швидко аналізувати дані.[12]

Щоб створити куб OLAP, використовується служба SQL Server Analysis Services (SSAS), яка надає зручні інструменти для роботи з OLAP-кубами та налаштування їх під інфраструктуру сховища даних.[13] OLAP-куб є частиною структури даних в службі SSAS і будується на основі бази даних OLAP, що дозволяє проводити практично миттєвий аналіз даних. Під час цієї роботи ми використовували середовище Visual Studio для взаємодії з SSAS. Перш за все, було потрібно визначити джерело даних, яким може бути або база даних OLAP, або сховище даних. OLAP-куб підключається до цього джерела даних для читання і обробки невикористаних даних шляхом виконання агрегування та опрацювання пов'язаних з ними вимірів.

Створений куб зображений на рисунку 2.5:

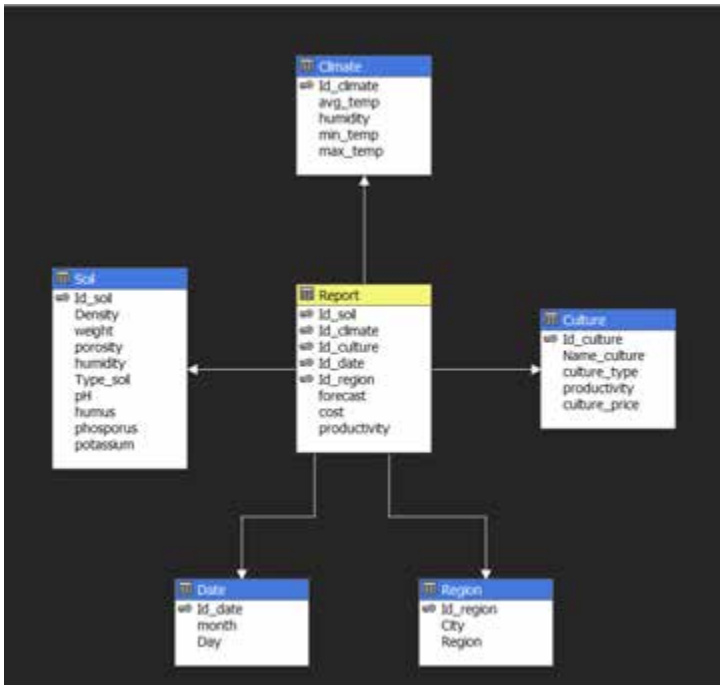

Рис. 2.5. OLAP-куб

Після створення кубу, необхідно буде наповнити його даними, використаємо для цього інструмент SSIS.

SQL Server Integration Services (SSIS) є ключовим компонентом у системі Microsoft SQL Server і призначений для реалізації процесу інтеграції даних із різних джерел. SSIS включає в себе можливості створення, налаштування і виконання пакетів для обробки цих даних.[14] Ці пакети можуть включати операції, такі як видобуток інформації із різних джерел, наприклад, баз даних, текстових файлів або веб-сервісів. Після видобутку даних можна виконувати їх перетворення та обробку (наприклад, фільтрацію, об'єднання та обчислення), а також завантажувати їх до цільових систем, таких як інші бази даних або сховища даних.

SSIS надає можливість створювати пакети за допомогою графічного інтерфейсу, що дозволяє визначати порядок виконання операцій обробки даних та налаштовувати параметри кожної операції. Це спрощує процес інтеграції даних та дозволяє розробникам створювати складні інтеграційні рішення без необхідності писати власний програмний код.

В інструменті SSIS є служба Data Flow, за допомогою якої було

проведено заповнення таблиць вимірів та фактів. Спочатку було створено потоки даних, по яким будуть передаватись дані.

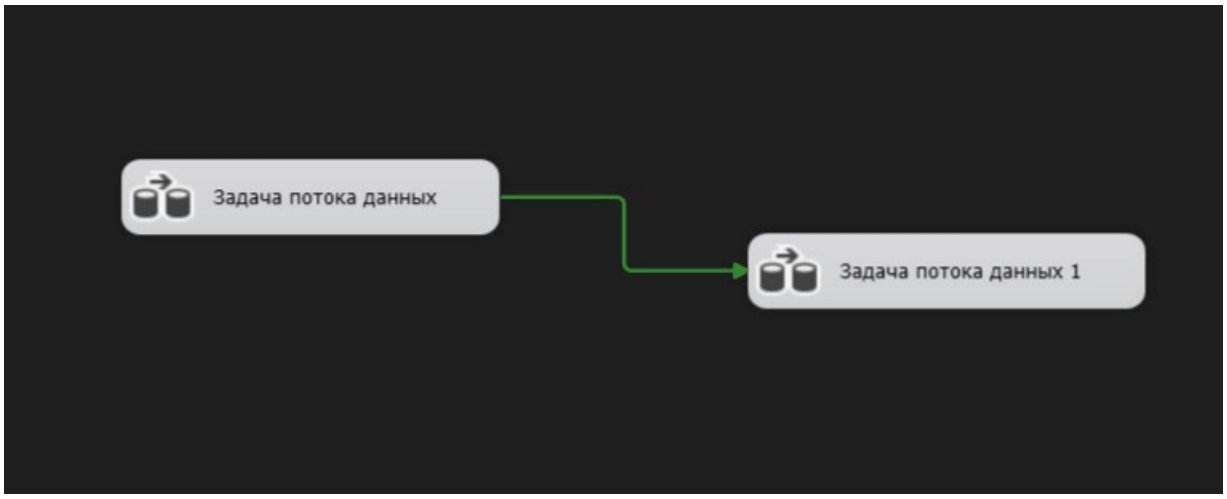

Рис. 2.6. Потоки даних

На першому етапі ми заповнюємо таблиці вимірів, для першого рівня:

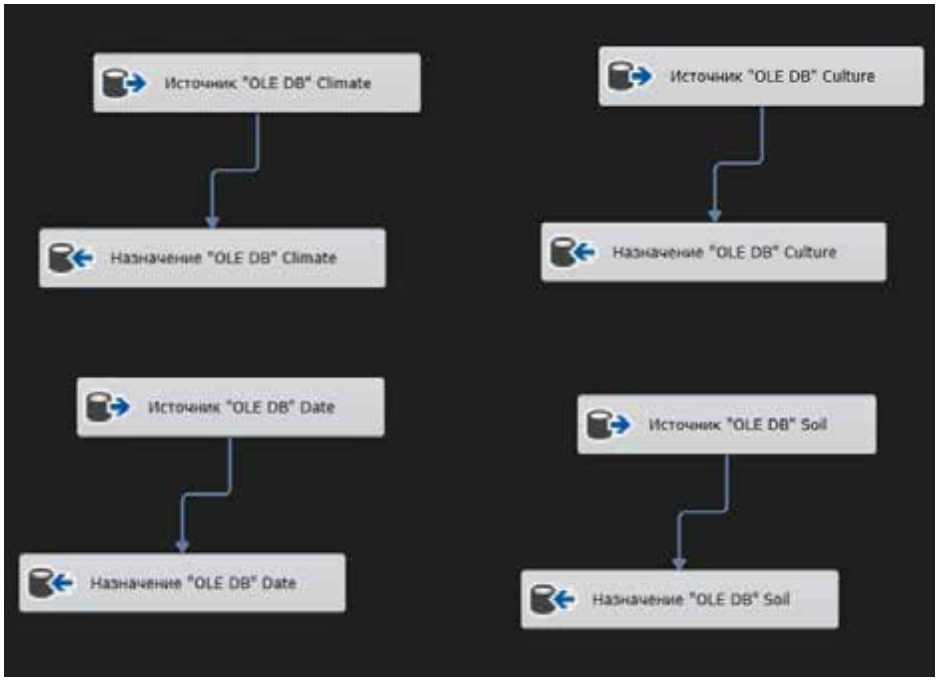

Рис. 2.7. Потоки даних 1-го рівня

Після чого налаштовуємо процес передачі даних, на прикладі для таблиці Culture:

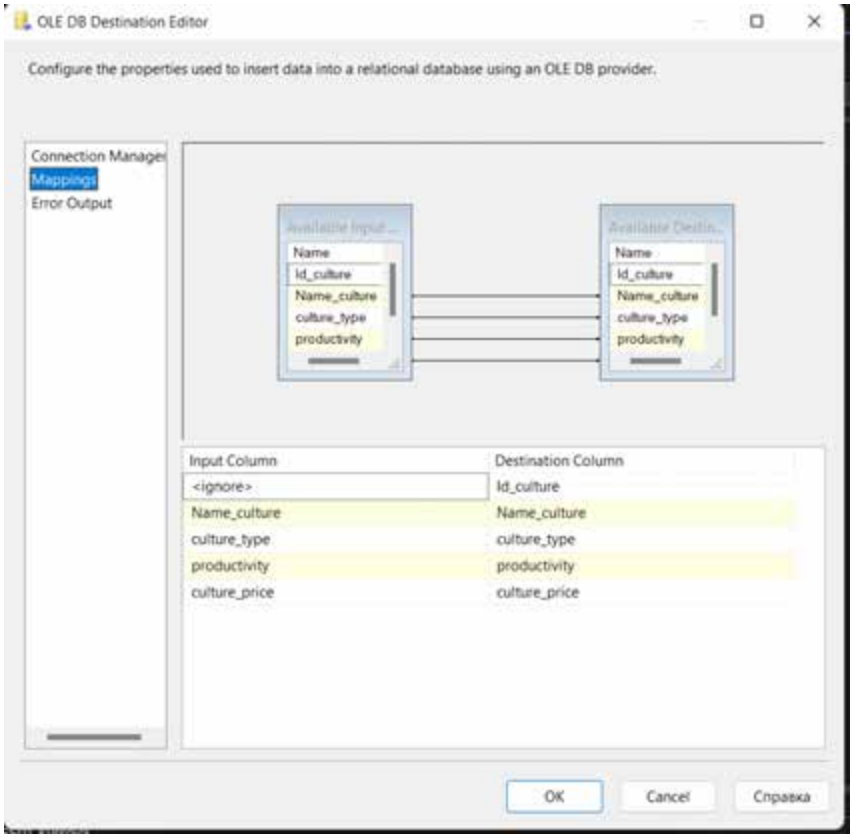

Рис. 2.8.Процес передачі даних

На другому етапі було створено потоки даних для другого рівня, а саме таблиці фактів:

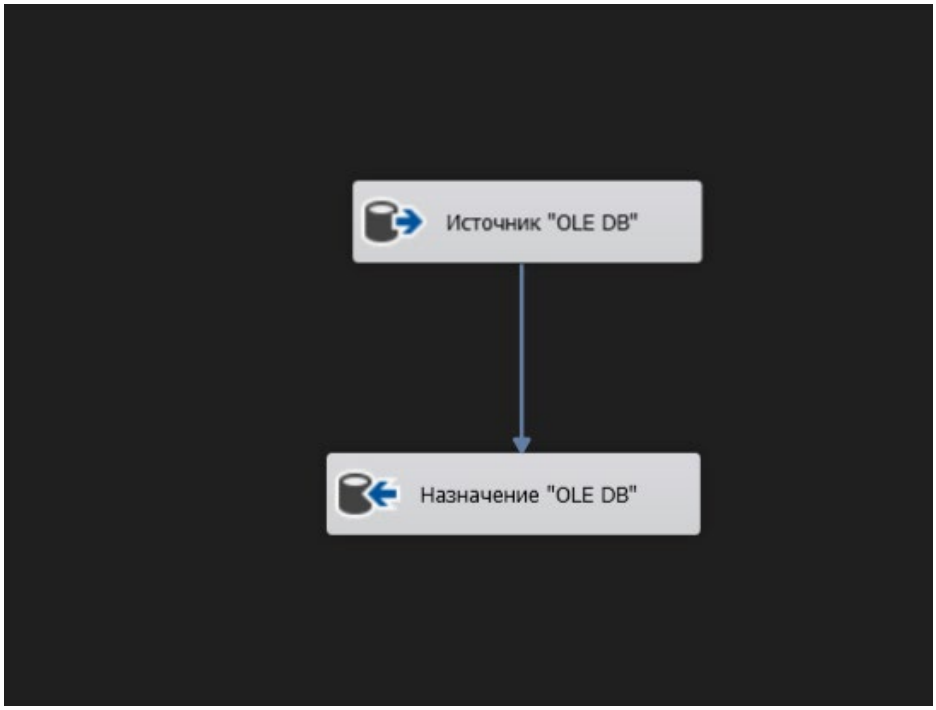

Рис. 2.9. Потоки даних 2-го рівня

3. РОЗРОБКА СИСТЕМИ

3.1 Опис архітектури системи

Комп’ютерна система прогнозування врожайності сільськогосподарських культур складається з забезпечення доступу до джерел інформації, розробки концепцій бізнес-планування, обробки запитів і відправлення відповідей, створення користувацького інтерфейсу та визначення способів представлення даних в цьому інтерфейсі.

Відповідно до вимог розробленої системи, проста архітектура клієнт-сервер може поділити систему на два модулі: програмний модуль, відповідальний за зберігання на сервері, та інтерфейс користувача, розташований на стороні клієнта. Модуль обробки даних обслуговує як сторону клієнта, так і сервера. Ця архітектура спільно використовується обома модулями, проте основна увага приділяється налаштуванню сервера.

Останнім часом архітектура REST (REpresentational State Transfer) стала стандартною при проектуванні веб-сервісів і веб-API. Архітектурний стиль REST, є одним з ключових підходів до створення веб-додатків і веб-сервісів. REST виникла з метою створення простого і ефективного механізму взаємодії між різними компонентами розподілених систем, таких як Інтернет. Цей підхід був розроблений і описаний Роєм Філдінгом у своїй дисертації 2000 року, і з того часу REST став широко використовуваним стандартом для реалізації веб-сервісів та API. Усі дані та функціональність виражаються у вигляді ресурсів, які мають унікальні ідентифікатори (URL). Ресурси можуть мати конкретну природу, наприклад, це може бути сторінка користувача на веб-сайті або запис в базі даних. Ці ресурси можуть бути представлені у різних форматах, таких як HTML, XML або JSON, і клієнт може вибрати той формат, який відповідає його потребам. Взаємодія між клієнтом і сервером відбувається за допомогою обміну повідомленнями про поточний стан ресурсу. Кожен запит від клієнта має містити всю необхідну інформацію для зрозуміння його мети, включаючи всі необхідні дані. Ця взаємодія між клієнтом і сервером має бути безстандартною, що означає,

що кожен запит від клієнта повинен містити всю необхідну інформацію для обробки запиту, і сервер не зберігає жодного контексту клієнта між запитами. REST використовує стандартні HTTP-методи, такі як GET, POST, PUT та DELETE, для виконання різних операцій над ресурсами. Наприклад, GET використовується для отримання даних, а POST - для створення нових ресурсів.

Розглянемо також RESTful, це архітектура яка використовується для опису API та веб серверів, вона дає можливість користувачам легко взаємодіяти з сервером , використовуючи стандартні HTTP запити. Це робить їх ідеальними для розробки розподілених систем, а також для створення масштабованих та надійних додатків.

Розширення Flask-RESTful дозволяє легко створювати RESTful веб-сервіси на основі фреймворку Flask для розробки веб-додатків на мові програмування Python. Взаємодія Flask із Flask-RESTful надає зручний та продуктивний спосіб створення веб-сервісів, які відповідають принципам REST. Основні переваги використання Flask-RESTful у спільній роботі з Flask включають: Легку налаштовуваність, Flask-RESTful надає інструменти для організації маршрутів і обробки запитів в стилі REST. Це дозволяє швидко налаштувати обробку різних HTTP-методів для ресурсів. Також є можливість автоматично перетворювати дані з Python-об'єктів у формати, які легко розуміють клієнти (наприклад, JSON або XML), і навпаки. Це спрощує обробку вхідних і вихідних даних. Він допомагає валідувати вхідні дані та обробляти помилки, що виникають під час обробки запитів. Це робить веб-сервіс більш надійним та стійким до помилок.

Усі ці можливості дозволяють створювати високоефективні та надійні веб-сервіси, які відповідають стандартам REST, використовуючи Flask та Flask-RESTful. Ця комбінація є популярним вибором серед розробників Python, які створюють веб-додатки і сервіси.

3.2 Вибір інструментарію для розробки системи

Під час проектування та розробки інформаційно-аналітичної системи прогнозування врожайності сільськогосподарських культур, вибір інструментарію є критичним етапом, який визначає успішність та ефективність всього проекту. У цьому розділі буде розглянуто основні аспекти вибору інструментів та технологій, які будуть використовуватися під час розробки системи.

В процесі виконання роботи , було розроблено систему за допомогою Visual Studio Code. Це безкоштовний текстовий редактор, що використовується для створення програмного забезпечення.[15] VS Code підтримує багато мов програмування, включаючи JavaScript, Python, Java, C++, і багато інших. Це дозволяє працювати із різними мовами, що дає можливість реалізовувати різні задачі, без необхідності переходити до інших інструментів. Великою плюсом є те, що під час кодування VS Code надає пропозиції щодо завершення рядків та виправлення поширених синтаксичних помилок, що дуже пришвидшує та полегшує процес написання коду. Також дуже привабливим є великий набір розширень, які можна встановити, щоб розширити його можливості. Це дає можливість налаштовувати редактор під конкретні вимоги. Не дивлячись на велику кількість функцій та інструментів, редактор має дуже зручний та легкий у користуванні інтерфейс, швидкий у роботі, та не вимагає великої кількості ресурсів комп’ютера. Проте справедливо треба зазначити, що VS Code не підходить для всіх видів робіт. Відсутність певних функцій, які притаманні спеціалізованим інструментам для певних областей розробки (наприклад, графічного дизайну чи наукових обчислень), може зробити VS Code менш підходящим для деяких видів робіт.

Для розробки бази даних було використано pgAdmin. Це безкоштовний інструмент для управління та розробки баз даних PostgreSQL, який пропонує зручний графічний інтерфейс для керування серверами, створення та редагування баз даних, таблиць, індексів і багатьох інших об'єктів. Він підтримує

різноманітні можливості, включаючи виконання SQL-запитів, редагування таблиць, налаштування системи безпеки на рівні ролей та інше.[16] pgAdmin доступний для використання на різних операційних системах, має активну спільноту користувачів і підтримує функцію імпорту та експорту даних. Що було дуже важливим для виконання цього проекту, оскільки надає можливість імпортувати дані у форматі .XLSX та .CSV, оскільки дані було отримані саме у цих форматах. Незважаючи на деякі особливості налаштування та потребу у високих обчислювальних ресурсах, це потужний інструмент, який спрощує роботу з базами даних PostgreSQL.

Для розробки серверу була використана мова Python, для веб частини HTML та CSS, а для розробки даних було використано SQL. Далі розглянемо кожну з цих мов детальніше.

Python - це інструмент з безмежними можливостями для розробки програм та вирішення різних завдань, і його популярність серед глобальної спільноти розробників обумовлена його простотою, зрозумілістю коду та потужністю. Python став справжньою зіркою в світі програмування завдяки своїм властивостям. Однією з ключових переваг Python є його легкість розуміння. Читаючи код на Python, можна легко зрозуміти його, навіть якщо ви новачок у програмуванні. Його синтаксис схожий на звичну англійську мову, що робить його особливо доступним для початківців. Вирази та команди виглядають природно і легко запам'ятовуються. Пайтон також славиться своєю активною спільнотою розробників. Це означає, що завжди є де шукати допомогу, якщо ви зіткнетесь з певною проблемою або потребуєте консультації. Відкритий доступ до численних ресурсів і бібліотек робить Python дуже привабливим вибором для фахівців у цій галузі.

Не менш важливою складовою при розробці на Python,є фреймворки. Фреймворки, представляють собою заздалегідь розроблені та організовані комплекти бібліотек, стандартів, зразків та інших корисних інструментів, які спрощують і прискорюють процес розробки програмного забезпечення. Вони надають загальну структуру і фундамент для створення програм, що допомагає

уникнути втрат часу на початкове занурення в деталі та рутини розробки.[17] Фреймворки включають в себе готові рішення для стандартних задач, що дозволяє розробникам зекономити час, який раніше витрачали на написання основного коду та архітектури програми. Вони використовуються для розробки різноманітних програм, таких як веб-додатки, мобільні застосунки, настільні програми, ігри, розподілені системи, розробка штучного інтелекту, обробка даних та інші завдання. Вони допомагають розробникам зосередитися на бізнес-логіці та функціональності, позбавляючи їх необхідності писати базовий код з нуля. Один з фреймворків який було використано при розробці системи є Flask.

Flask - це веб-фреймворк, за допомогою якого можна об'єднати бекенд (логіку серверної сторони) та фронтенд (користувацький інтерфейс, який бачить користувач) в одному веб-додатку. Він дозволяє розробникам створювати веб-сайти та веб-додатки, які можуть взаємодіяти з користувачем через браузер. Основна ідея полягає в тому, що Flask надає можливість визначати URL-адреси та обробляти HTTP-запити, такі як GET або POST. Коли користувач відкриває веб-сайт у браузері, він відправляє HTTP-запит на сервер, де Flask обробляє цей запит і повертає відповідь у вигляді HTML-сторінки або інших даних, які браузер може відобразити.[18]

Flask розширює свої можливості поза генерацією HTML. Він також забезпечує підтримку роботи з шаблонами, що дозволяє вставляти дані з сервера безпосередньо у сторінки HTML. Це дозволяє створювати динамічні сторінки, які можуть змінюватися відповідно до інформації на сервері або дій користувача. За допомогою Flask можна також передавати та отримувати дані у форматі JSON, що робить його ідеальним інструментом для розробки веб-служб та API. Це означає, що бекенд на Flask може ефективно обробляти запити, виконувати необхідну логіку та передавати дані фронтенду або іншим додаткам через мережу.

Отже, Flask допомагає єднати роботу серверної сторони (бекенду) та користувацького інтерфейсу (фронтенду) у єдиному веб-додатку, що дозволяє створювати повноцінні та інтерактивні веб-сайти та додатки.

Розглянемо детальніше протокол HTTP, метод GET та POST.

HTTP (Hypertext Transfer Protocol) - це стандартний протокол для передачі гіпертекстових даних через комп'ютерну мережу. Він використовується для обміну інформацією між веб-серверами і клієнтами (найчастіше веб-браузерами), що дозволяє користувачам отримувати доступ до веб-сторінок, зображень, відео, аудіо і інших ресурсів через Інтернет. HTTP визначає правила і формати для цієї взаємодії, забезпечуючи стандартизований спосіб обміну даними.[19]

GET - це один з методів HTTP-запитів, використовуваний для запиту ресурсів з веб-сервера. Коли клієнт виконує GET-запит, він надсилає запитання серверу, і сервер повертає запитані дані. GET-запити зазвичай використовуються для отримання вмісту веб-сторінок або інших ресурсів з сервера. GET-запити можуть містити параметри, які додаються до URL-адреси і передаються серверу для точного визначення запитаних даних.

POST - це інший метод HTTP-запитів, використовуваний для надсилання даних на веб-сервер для обробки. У відповідь на POST-запит, сервер може виконувати різні дії, такі як збереження даних, оновлення ресурсів, обробку форм, аутентифікацію тощо. POST-запити дозволяють передавати великі обсяги даних та зберігати їх у відповідних форматах на сервері.

Отже, HTTP - це протокол для передачі даних, GET - метод для отримання ресурсів, а POST - метод для надсилання даних на сервер для обробки.

Наступним важливим інструментом розробки є бібліотеки. Далі будуть розглянуті бібліотеки, необхідні для створення проекту.

Psycopg2 - це бібліотека для мови програмування Python, розроблена для взаємодії з базами даних PostgreSQL. Вона надає можливість використовувати всі функціональні можливості PostgreSQL та легко працювати з базою даних. Psycopg2 спрощує підключення до бази даних, виконання SQL-запитів, обробку результатів та керування транзакціями. Бібліотека надає зручний і простий спосіб підключитися до PostgreSQL-сервера, використовуючи різні параметри підключення.[20]

В цілому, Psycopg2 представляє собою потужний інструмент для взаємодії з PostgreSQL у Python-додатках та забезпечує зручну та надійну роботу з цією базою даних.

Matplotlib - це бібліотека для мови програмування Python, спрямована на візуалізацію даних. Вона є однією з найбільш популярних інструментів для створення графіків, діаграм і інших графічних представлень даних. Matplotlib надає широкий набір можливостей для створення якісних інтерактивних графіків, які полегшують аналіз і відображення оброблених даних.[21]

Matplotlib у системах прогнозування дозволяє створювати різноманітні типи графіків, такі як лінійні, стовпчасті, кругові діаграми, теплові карти, графіки розсіювання і інші. Відображення даних в графічному вигляді сприяє аналізу і зрозумінню взаємозв'язків в даних, виявленню закономірностей та патернів. У системах прогнозування часто потрібно представити результати прогнозування або аналізу даних. Matplotlib дозволяє створювати зрозумілі та інформативні графіки, які можна використовувати для звітності та комунікації результатів з іншими користувачами.

HTML та CSS - це дві основні технології, які використовуються для розробки веб-сайтів і веб-додатків. HTML використовується для створення структури веб-сторінок, визначення текстового та візуального контенту, розміщення зображень, посилань та форм для обміну даними з користувачами. Це визначає основну "кістку" сторінки, включаючи заголовок, текст, таблиці та інші елементи. З іншого боку, CSS відповідає за стиль та зовнішній вигляд веб-сторінок. Використовуючи CSS, розробники можуть задавати кольори, шрифти, розташування та дизайн елементів на сторінці. Це дозволяє зробити веб-сторінки більш привабливими та професійними. Крім того, CSS дозволяє створювати адаптивний дизайн, який працює на різних пристроях та розмірах екрану. Разом HTML і CSS визначають основу веб-розробки і дозволяють розробникам створювати красиві та функціональні веб-сторінки. HTML відповідає за структуру і контент сторінки, а CSS - за її вигляд та стиль. Ці дві технології

спільно доповнюють одна одну і дозволяють створювати візуально привабливі та ефективні веб-додатки для всесвітньої мережі[22].

HTML та CSS дозволяють створювати дизайни, які легко пристосовуються до різних розмірів екранів. Це важливо, оскільки веб-сайти повинні бути зручними як на комп'ютерах, так і на мобільних пристроях. Ці мови постійно вдосконалюються, останні версії - HTML5 та CSS3 - включають в себе широкий спектр нововведень, таких як можливості для мультимедіа, анімації, адаптація до мобільних пристроїв, розширені можливості управління шрифтами та інше.

SQL(Structured Query Language) - це мова, яка дозволяє працювати з базами даних. Вона допомагає зберігати, редагувати та вибирати інформацію з бази даних, як спосіб взаємодіяти зі списками даних. SQL - це інструмент для роботи з інформацією у великих сховищах даних. Запити SQL виконуються до бази даних і дозволяють нам взаємодіяти з даними у ній. SQL включає в себе команди, які можуть бути використані для створення таблиць, встановлення відносин між ними, отримання даних з таблиць за певними умовами (оператор SELECT), зміни інформації (оператор UPDATE), видалення даних (оператор DELETE) та багато інших операцій.[23]

Основна перевага SQL полягає в його універсальності та можливості використовувати його для різних типів баз даних. SQL є стандартом для реляційних баз даних і використовується в багатьох галузях, таких як аналіз даних, розробка програмного забезпечення та управління інформацією. Він дозволяє ефективно працювати з великими обсягами даних і виконувати складні операції над ними, що робить його важливим інструментом для обробки інформації в сучасному світі.

3.3 Вибір та розробка бази даних

Для виконання цього проєкту, як вже було описано раніше, буде використовуватись реляційна база даних, а саме PostgreSQL. ЇЇ завдання

очевидне, зберігати дані, проте нас цікавить саме структуроване зберігання даних, бо зберігати дані можна і в примітивному вигляді, наприклад у текстовому файлі, але це нас не влаштовує. Тож надалі в цьому розділі буде розглянуто всі ключові аспекти, надана відповідь на всі питання, порівняємо PostgreSQL з аналогами, та розглянемо цю СУБД детальніше.

PostgreSQL - це сучасна та потужна система управління базами даних (СУБД), яка славиться своєю надійністю. Важливою перевагою її використання є той факт, що вона є відкритою системою з вільним доступом до вихідного коду. Це означає, що її можна використовувати безкоштовно і адаптувати під свої потреби. PostgreSQL є системою управління реляційними базами даних ,що дозволяє зберігати дані у вигляді таблиць зі структурою, яка визначає відносини між ними.[24]

Реляційна база даних - це тип бази даних, яка базується на моделі реляційних таблиць. Основні характеристики реляційних баз даних та PostgreSQL включають такі аспекти:

- Інформацію зберігають у вигляді таблиць, де кожна таблиця має свою власну унікальну групу стовпців і рядків. Взаємозв'язки між даними встановлюються за допомогою таблиць.
- В реляційних базах даних вимагається, щоб дані були введені відповідно до попередньо визначеної структури, що сприяє більшій точності та цілісності інформації.
- В реляційних базах даних можна встановлювати правила цілісності, які служать для запобігання помилковим або недопустимим даним.
- Реляційна модель дозволяє встановлювати відносини між таблицями, що дозволяє зв'язувати дані між собою.
- Системи управління базами даних реляційного типу, на прикладі PostgreSQL, здатні виконувати складні операції та обробляти великі обсяги інформації, що робить їх популярними в корпоративних та наукових сферах.

Нереляційні бази даних (NoSQL бази даних) відрізняються від реляційних баз даних у своєму підході до організації та зберігання інформації. Реляційні бази даних використовують таблиці зі стовпцями для структурування та зберігання даних і мають фіксовану схему, що означає, що структура даних визначається перед введенням даних. Напроти, нереляційні бази даних зазвичай не мають фіксованої схеми і можуть зберігати дані у різних форматах, таких як документи, об'єкти чи колекції. Це дозволяє їм ефективно взаємодіяти з неструктурованими або змінюються даними. Нереляційні бази даних також володіють горизонтальним масштабуванням, що означає, що їх можна легко розподіляти на кілька серверів для обробки великих обсягів даних та високої навантаженості. Ці бази даних часто використовуються в проектах, де швидкість обробки даних вкрай важлива, і де схема даних може змінюватися динамічно, таких як соціальні мережі або системи аналізу даних. Вони також можуть містити різні моделі ,такі як бази даних документів, ключі, стовпчасті графи та інші, кожна з яких підходить для різного типу завдань. Отже, вибір між реляційними і нереляційними базами даних залежить від конкретних потреб проекту та характеристик даних, які обробляються.

Розглянувши типи баз даних та їх відмінності, повернемось до розгляду обраної СУБД. PostgreSQL володіє розширеними можливостями та додатковими функціями, включаючи обробку геоданих, текстовий та пошуковий аналіз, індексацію для швидкого пошуку і багато інших. Крім того, він є повністю сумісним з мовою SQL, що дозволяє використовувати стандартні SQL-запити. Ця СУБД забезпечує високу ефективність обробки даних, особливо при роботі з великими обсягами і відзначається підтримкою різних операційних систем. PostgreSQL також включає механізми для забезпечення надійності і можливості відновлення даних у випадку непередбачуваних збоїв. Важливо відзначити, що PostgreSQL має велику та активну спільноту розробників і користувачів, що гарантує наявність багатьох ресурсів і підтримки для користувачів.

PostgreSQL виділяється серед багатьох продуктів баз даних завдяки своєму розширеному набору типів даних, що може бути визначений користувачем.

Більшість інших СУБД обмежують користувачів попередньо встановленими типами даних, такими як цілі числа чи тексти. Однак PostgreSQL дозволяє створювати власні типи даних, що відповідають вашим потребам. Наприклад, ви можете створити складений тип для комплексних чисел, тип для координат та навіть свій власний чисельний тип. Ці інноваційні типи даних ефективні та корисні тільки тоді, коли ви визначите для них необхідні операції, такі як додавання чи віднімання. PostgreSQL також автоматично створює для вас типи масивів, коли ви створюєте власні типи, спрощуючи роботу з колекціями даних. Однією з інших корисних можливостей PostgreSQL є можливість автоматичного створення попередньо визначених таблиць. Наприклад, якщо ви створите таблицю для збереження інформації про собак, PostgreSQL автоматично опрацьовує цю інформацію як окремий тип даних, що спрощує взаємодію з нею та сприяє продуктивності. У вас є можливість створювати функції, які обробляють дані для одного об'єкта за один раз, а також функції, які можуть діяти з наборами об'єктів одночасно. Користуючись спеціальними типами даних, ви можете підвищити продуктивність і зручність роботи з вашими даними. Потім можна створювати функції, які стають все більш важливими для вашого проекту та скорочують кількість коду, який потрібно підтримувати[25].

Наступним кромок буде порівняння PostgreSQL, Oracle, Microsoft SQL Server і IBM DB2 - це всі системи управління базами даних (СУБД), і вони мають свої відмінності і сфери застосування. Розглянемо деякі ключові відмінності між ними:

1. Власність і ліцензування:
    - PostgreSQL: Відкритий вихідний код і безкоштовний для використання. Має активну спільноту розробників.
    - Oracle: Комерційна СУБД, яка потребує ліцензії і оплати. Володіє великою функціональністю і підтримкою.
    - Microsoft SQL Server: Комерційна СУБД, доступна на ліцензійній основі. Популярна серед підприємств.

- BM DB2: Комерційна СУБД, яка також потребує ліцензії. Зазвичай використовується в великих підприємствах.

2. Підтримувані операційні системи:
   - PostgreSQL: Підтримується на багатьох операційних системах, включаючи Windows, Linux, macOS і багато інших.
   - Oracle: Підтримується на численних операційних системах, але може бути складним у встановленні та налаштуванні.
   - Microsoft SQL Server: В основному підтримується на Windows, хоча є версії, які працюють на Linux.
   - IBM DB2: Доступний для різних операційних систем, включаючи Windows, Linux, AIX, IBM і багато інших.
3. Масштабованість та продуктивність:
   - PostgreSQL: Добре масштабується і підходить для великих обсягів даних. Продуктивність залежить від налаштувань та оптимізації.
   - Oracle: Має вражаючу продуктивність і масштабованість, але вимагає великих ресурсів і витрат.
   - Microsoft SQL Server: Ефективний для середніх і великих підприємств. Має гарну підтримку бізнес-логіки.
   - IBM DB2: Відомий своєю продуктивністю та масштабованістю, особливо для платформи IBM Power Systems.
4. Функціональність і особливості:
   - PostgreSQL: Має багатий набір функціональності, включаючи розширення і можливості для геопросторового аналізу та JSON-документів.
   - Oracle: Володіє величезними можливостями, включаючи підтримку гарячого резервного копіювання та високою доступності.
   - Microsoft SQL Server: Включає багато інструментів для аналітики і бізнес-інтелекту, також має підтримку інтеграції з продуктами Microsoft.
   - IBM DB2: Відомий своєю оптимізацією для аналітичних завдань і великих даних.

5. Останні досить важливим аспектом, порівняємо вартість:
    - PostgreSQL: Безкоштовний із відкритим вихідним кодом, але можуть виникати витрати на підтримку та налаштування.
    - Oracle, Microsoft SQL Server і IBM DB2: Комерційні продукти, які вимагають ліцензій і оплати, вартість може бути значною.

Підсумовуючи сказане, можна сказати, що при виборі між цими СУБД важливо враховувати потреби, фінансові можливості та доступні ресурси для кожного проекту, а також конкретні вимоги і особливості використання. Кожна із них має свої переваги і може бути вигідною в конкретних ситуаціях.

Розглянувши таку велику кількість сильних сторін PostgreSQL, для об'єктивності слід описати також її недоліки, котрі для певних систем можуть стати критичними при виборі СУБД. Тож розглянемо її недоліки.

Зазвичай, для установки PostgreSQL без додаткових розширень потрібно більше 100 МБ або більше місця на диску. Це правило не враховує сценарії, коли PostgreSQL використовується на дуже малих пристроях, наприклад, як просте кеш-сховище. В таких випадках існують легкі альтернативи у вигляді менших баз даних, які можуть задовольнити ваші потреби без зайвого обсягу. Важливо враховувати, що PostgreSQL відомий своєю високою надійністю і безпекою, і якщо ви розробляєте додаток, який акцентується на безпеці на рівні застосування, то ви можете не потребувати всіх функції безпеки на рівні бази даних PostgreSQL. Якщо ваша програма використовується невеликою кількістю користувачів або вам потрібна проста однокористувацька база даних, то ви можете розглядати легкі бази даних, такі як SQLite або Firebird, як альтернативу. Проте, зазвичай PostgreSQL використовується разом з іншими системами баз даних. Наприклад, його можна поєднувати з Redis або Memcache для кешування результатів запитів, що покращує продуктивність. Також PostgreSQL може використовуватися разом з SQLite для зберігання окремих наборів даних для офлайн-запитів, коли він виступає як основна база даних програми.

3.4 Реалізація алгоритмів та функціональності системи

У системі всі дані оброблюються на стороні сервера, у контексті фронтенд системи, її основна роль полягає в відображенні даних. Серверну логіку можна розділити на дві основні складові: логіку пошуку даних і логіку опрацювання даних.

Для початку розглянемо один з найважливіших аспектів, отримання даних у систему, для цього було розроблено підключення до бази даних за допомогою psycopg2, основні принципи роботи якого було розглянуто раніше, а зараз на практиці розглянемо його використання. На рисунку 3.1 представлено реалізацію підключення системи до БД.

```
import psycopg2
import psycopg2.extras


def ret_culture_yield_single(culture_name):
    conn = psycopg2.connect(dbname='DWH', user='postgres', password='1234', host='localhost')

    # init cursor
    cursor = conn.cursor(cursor_factory = psycopg2.extras.RealDictCursor)
```

Рис. 3.1. Використання psycopg2 для підключення до бази даних

Спочатку ми імпортуємо необхідні бібліотеки для роботи з базою даних PostgreSQL у Python. Розглянемо підмодуль psycopg2.extras:

import psycopg2.extras: Ця лінія імпортує підмодуль extras з бібліотеки psycopg2, який містить додаткові функції та класи для роботи з результатами запитів до бази даних.

Далі ми визначає функцію ret_culture_yield_single, яка буде використовуватися для отримання інформації про врожайність певної культури. Після чого відбувається безпосереднє підключення до бази даних PostgreSQL . Обов’язково необхідно вказувати параметри, такі як ім'я бази даних, ім'я користувача, пароль і хост (сервер бази даних). З'єднання - це спосіб отримати доступ до бази даних. Останнім етапом підключення є ініціалізація курсору, який є інструментом для виконання SQL-запитів до бази даних. Важливою частиною цього рядка є параметр cursor_factory, який встановлює тип курсора на

RealDictCursor. Цей тип курсора дозволяє отримувати результати запитів у вигляді словників, що полегшує роботу з даними. Отже, розглянувши цей код, було показано процес створення з'єднання з базою даних і підготовкою курсору, який можна використовувати для взаємодії з базою даних і отримання даних про врожайність певної культури.

Процес побудови прогнозу будується за допомогою лінійної регресії. Використання лінійної регресії в сільському господарстві та аграрному секторі стає дедалі актуальнішим у сучасних умовах. Цей статистичний метод став необхідним інструментом для аналізу та прогнозування врожайності сільськогосподарських культур. Завдяки своїй спроможності моделювати взаємозв'язки між різними факторами та врожайністю, лінійна регресія допомагає фермерам та агрономам приймати обґрунтовані рішення, що впливають на виробництво сільськогосподарських культур.

Лінійна регресія - це статистичний метод, який використовується для аналізу та моделювання залежності між залежною змінною (в моєму випадку, врожайністю сільськогосподарських культур) і однією або кількома незалежними змінними (факторами), які впливають на цю залежну змінну. Лінійна регресія передбачає, що залежність між змінними може бути виражена лінійною функцією. Незалежна змінна, це змінна, яка діє самостійно і не піддається впливу інших змінних. Змінюючи незалежну змінну, ми призводимо до зміни значень залежної змінної. Залежна змінна - це та змінна, яку ми вивчаємо, і яку регресійна модель намагається передбачити. У завданнях лінійної регресії кожне спостереження або випадок включає як значення залежної змінної, так і значення незалежної змінної. Переваги лінійної регресії включають:

- Простота розуміння та використання
- Широке застосування
- Ефективність в задачах прогнозування

Недоліки лінійної регресії включають:

- Неможливість моделювання нелінійних зв'язків
- Чутливість до мультиколінеарності

Лінійна модель - це математична модель, яка використовується для опису залежностей між змінними. У випадку лінійної моделі, припускається, що залежність між змінними є лінійною, тобто може бути виражена у вигляді лінійної функції. Модель лінійної регресії виглядає наступним чином:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p + e, \quad (3.1)$$

Де:

- Y - залежна змінна(Урожайність);
- $X_1, X_2, ... X_p$ - незалежні змінні(параметри ґрунту, кліматичні умови);
- $\beta_0, \beta_1, \beta_2, ... \beta_p$ - параметри моделі
- e- похибка.

Лінійні моделі можуть мати одну або більше незалежних змінних, і їх можна адаптувати до різних ситуацій. Параметри лінійної моделі (наприклад, коефіцієнти перед змінними) обчислюються за допомогою статистичних методів, які найкраще відповідають реальним даним.

Загалом лінійна регресія є важливим інструментом у багатьох галузях, включаючи сільське господарство, економіку, медицину та багато інших. Вона допомагає розуміти, які фактори впливають на залежну змінну та як їх взаємодія може впливати на результати.

Наступним важливим модулем є побудова графіків, за допомогою бібліотеки matplotlib, основні функції якої теж було розглянуто раніше. Основна мета цього модулю - створити різні типи графіків, такі як стовпчикова діаграма та лінійний графік, які візуалізують дані про врожайність певних культур та регіонів.

Підключаємо бібліотеку до проекту, за допомогою команди(1): import matplotlib.pyplot as plt (1). Наступним етапом розглянемо побудову графіку, який

буде відображати врожайність культури протягом років. Спочатку отримуємо значення врожайності культури по рокам(2) :

f_y = math.floor(float(psql.ret_culture_yield_single(culture_name)))(2).

Після чого значення років і врожайності ми перетворюємо у списки, за допомогою даного коду(3):

years = list(data.keys())

values = list(data.values()) (3).

Після всіх процедур, можна переходити безпосередньо до створення самого графіку, процес зображений на рисунку 3.2.

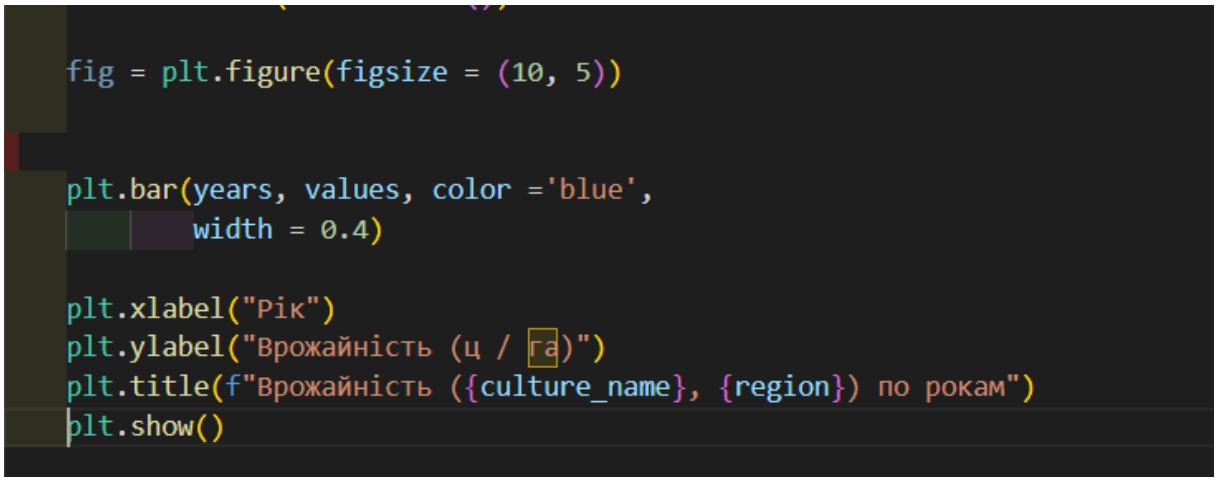

Рис. 3.2. Створення графіку

Ці рядки використовують бібліотеку matplotlib для створення стовпчикової діаграми та додають підписи до осей та заголовок. Відображення графіку відбувається виконання команди, plt.show().

3.5 Вибір інструментів для тестування системи

Тестування системи дуже важливий етап розробки, воно необхідне для проведення повноцінного дослідження та підтвердження правильності роботи системи. Для цього необхідно розглянути які існують для цього способи.

Розглянемо два типи тестування, ручне та автоматизоване. Якщо говорити про ручний спосіб тестування, то тут все інтуїтивно зрозуміло, це процес , коли

людина вручну проводить тести, для того аби виявити помилки або недоліки, та перевірити якість функціонування системи. В процесі тестувальник вручну проводить тестові сценарії, симулюючи дії користувача, включаючи перевірку вхідних та вихідних даних, правильність обробки даних та реакцію системи на вхідні дані. Перевіряє коректність відображення інтерфейсу, натискає на всі доступні кнопки та перевіряє реакцію системи. В разі виявлення якихось недоліків, це все має бути задокументовано. Важливим етапом перевірка роботи системи у різних браузерах, а також на різних пристроях. Особливо важливо буде перевірка коректності відображення контенту на мобільних пристроях, адже багато користувачів отримують доступ до сайту саме з мобільних телефонів.

Розглянувши основні принципи ручного тестування, можна сказати, що воно має свої переваги, такі як можливість виявлення складних проблем і внесення інтуїтивних оцінок щодо користувача. Проте слід також сказати про недоліки. Ручне тестування вимагає багато часу та може бути суб'єктивним. Тому здебільшого ручне тестування поєднується з автоматизованим тестуванням для більшої ефективності та швидкості виявлення помилок. Розглянемо детальніше інструменти для проведення автоматичного тестування.

Для проведення автоматичного тестування було обрано інструмент від Google – PageSpeed Insights.

PageSpeed Insights - це безкоштовний інструмент від Google, призначений для оцінки швидкості завантаження та продуктивності веб-сторінок. Цей інструмент допомагає веб-розробникам і власникам сайтів зрозуміти, наскільки ефективно їх веб-сайти завантажуються на різних пристроях та платформах. PageSpeed Insights аналізує сторінку та надає рекомендації щодо оптимізації, які допомагають покращити продуктивність та швидкість сторінки[26]. Основні можливості PageSpeed Insights включають:

1. Оцінка продуктивності, присвоює сторінці оцінку швидкості для мобільних та настільних пристроїв. Ця оцінка базується на різних факторах, таких як час завантаження, розмір сторінки, кількість запитів, оптимізація зображень і інші метрики продуктивності.

2. Надає рекомендації щодо можливих змін та оптимізацій, які можна внести на сторінку для покращення її продуктивності. Ці рекомендації включають у себе покращення завантаження зображень, кешування, зменшення кількості запитів до сервера та інші фактори.
3. Можна перевірити швидкість та продуктивність вашої сторінки на різних пристроях, включаючи мобільні телефони та планшети. Це дозволяє впевнитися, що ваш веб-сайт оптимізований для різних аудиторій.
4. Аналіз сторінки за HTTP і HTTPS для сторінок, що використовують звичайний протокол HTTP, так і для сторінок з шифруванням HTTPS.
5. Інструмент PageSpeed Insights надає розширений набір засобів для розробників, які дозволяють отримувати більше технічних деталей щодо швидкості сторінки.

Загалом можна підвести підсумок, що PageSpeed Insights може допомогти виявити області, які потребують оптимізації на веб-сайті, і покращити загальну продуктивність. Це особливо важливо для поліпшення користувацького досвіду та оптимізації для пошукових систем, оскільки швидкість завантаження сторінок впливає на позиції в пошукових результатах.

4. РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

4.1 Результати розробки системи прогнозування врожайності

Прогнозування врожайності є важливим завданням для сільського господарства. Це дозволяє фермерам планувати свої витрати та прибутки, а також приймати рішення про ведення господарства. У цьому розділі будуть представлені результати розробки системи прогнозування врожайності. Система була розроблена для прогнозування врожайності культур по регіонам України. Для навчання системи було використано дані про врожайність культур за останні 5 років, а також дані включали в себе інформацію про погодні умови, тип ґрунту, технічні показники ґрунту, які можуть впливати на врожайність. В результаті було розроблено веб-систему, розглянемо її детальніше, на Рис.4.1. зображено загальний вигляд сторінки, яка складається з двох розділів, розглянемо розділ «Головна сторінка».

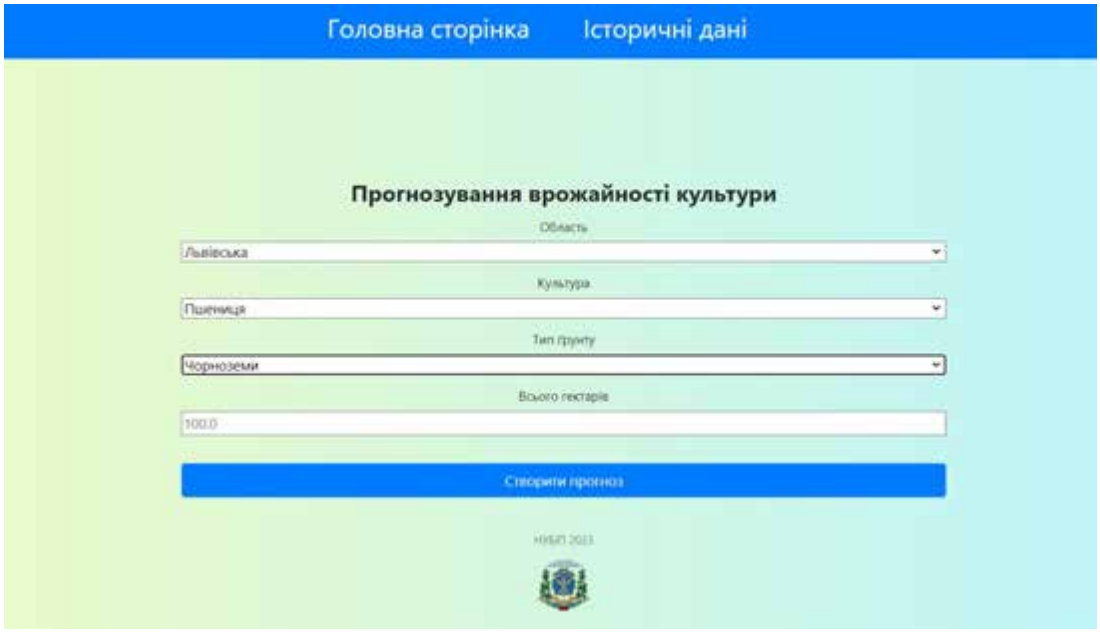

Рис. 4.1. Головна сторінка

На цій сторінці користувач може обрати одну з двадцяти чотирьох областей України, обрати культуру для якої йому було б цікаво побудувати прогноз, тип ґрунту на його полі, та ввести загальну кількість гектарів, на якій

він планує вирощувати культури. Розглянемо кожну з цих можливостей детальніше, на Рис.4.2. представлено вибір областей.

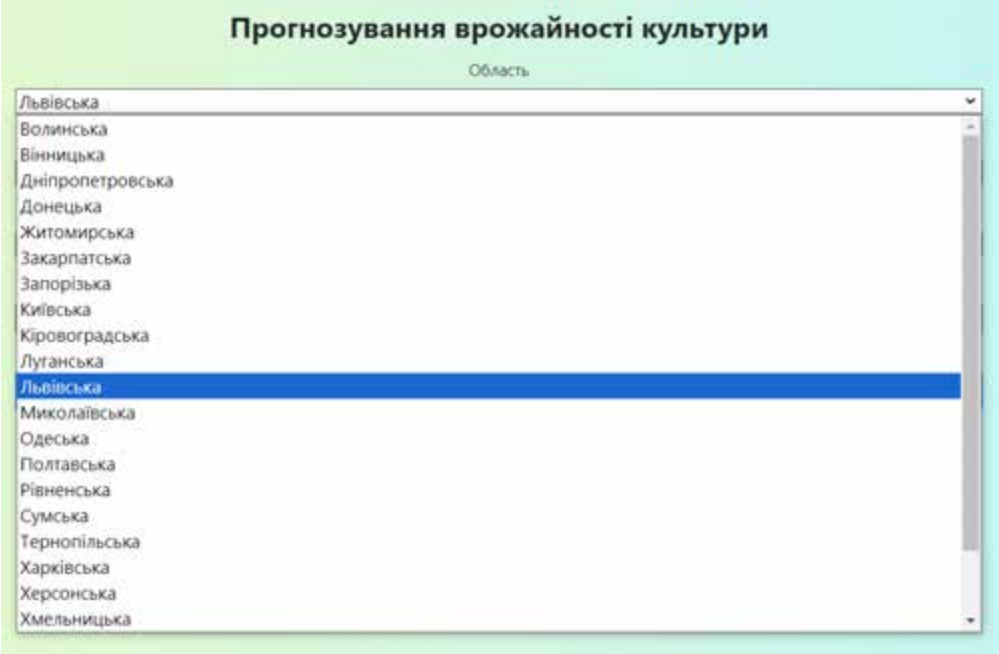

Рис. 4.2. Вибір області

Після вибору області, користувач може обрати культуру яка його цікавить, загалом для вибору доступно сто п’ятнадцять різних культур та їх типів, були додані ті культури, по яким є інформація щодо історичної врожайності, на Рис. 4.3. представлений процес вибору культури.

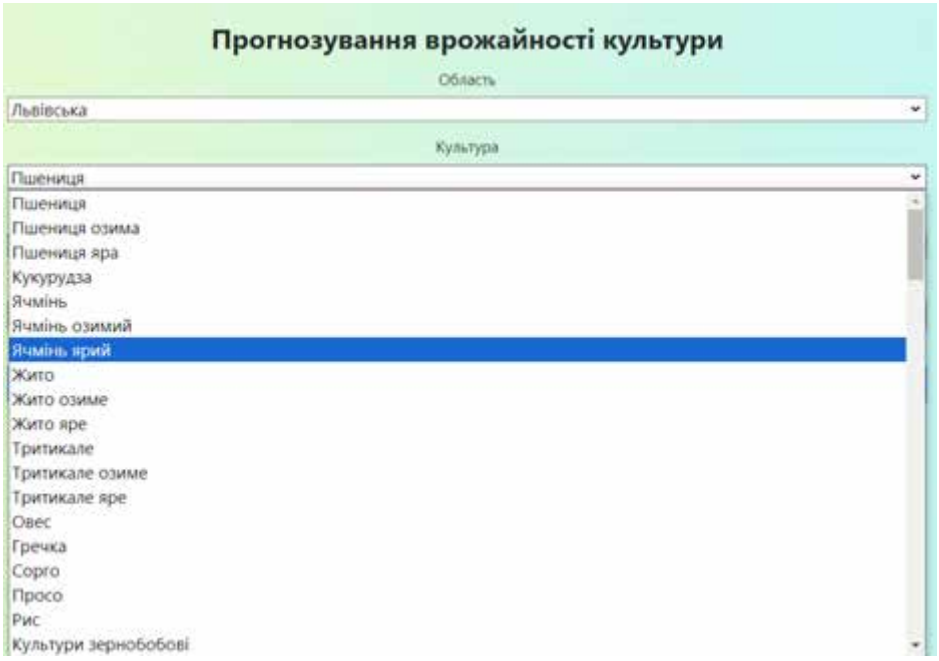

Рис. 4.3. Вибір культури

Після вибору культури користувачу доступний вибір типу ґрунту на його полі, побачимо це на Рис.4.4.

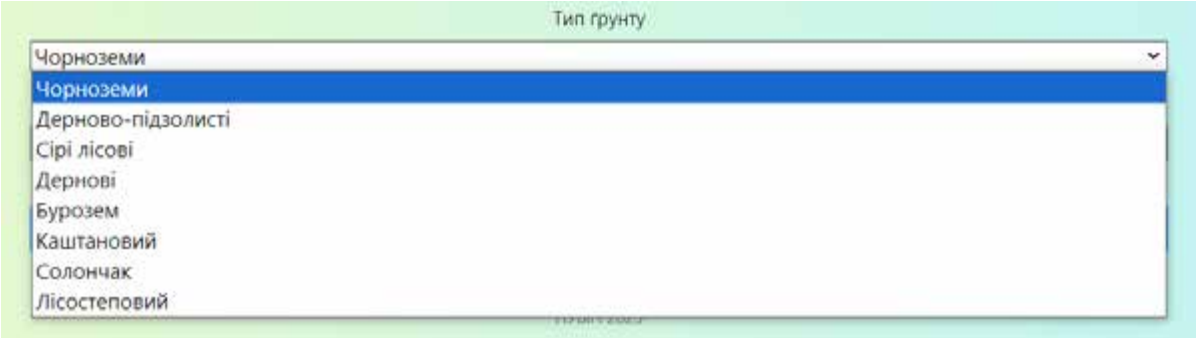

Рис. 4.4. Вибір типу ґрунту

Останнім полем для вибору є кількість гектарів, користувач ввівши значень, побудувавши прогноз, зможе отримати конкретне значення можливості врожаю певної культури, на Рис.4.5. зображено поле для вводу значень.

Рис. 4.5. Кількість гектарів

Вибравши всі параметри, користувач натиснувши кнопку «Створити прогноз», може побачити результат, а саме створену діаграму з прогнозом, показники ґрунту для вибраної області, точність моделі та прогнозу, а також пораховану суму прогнозованого врожаю для введеної кількості гектарів, на Рис.4.6. представлено побудований прогноз.

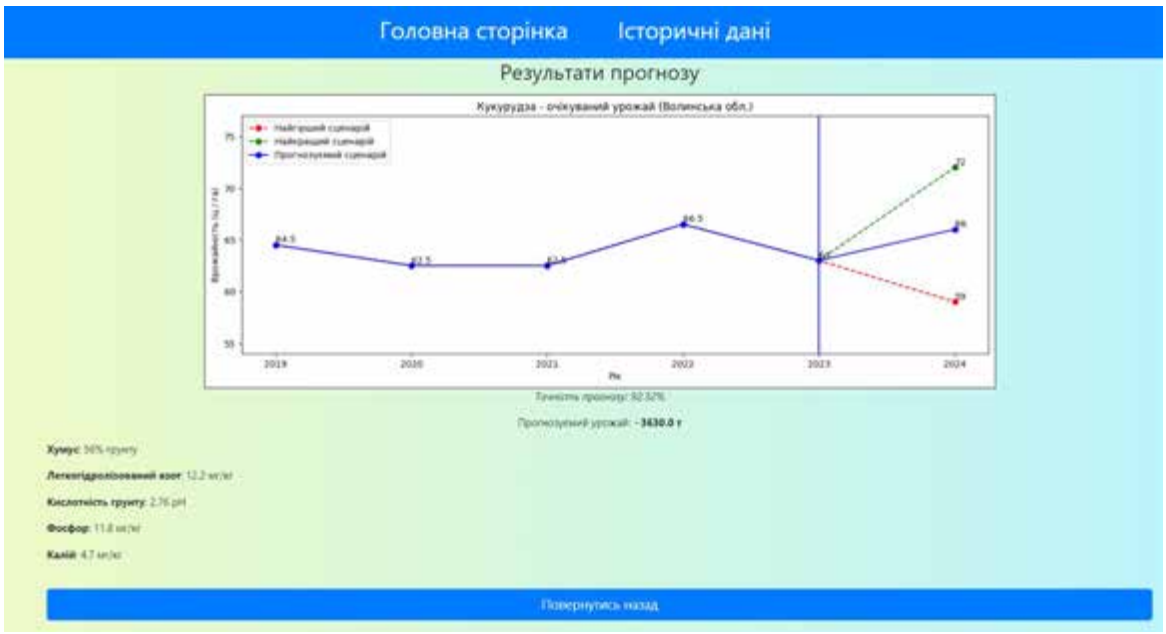

Рис. 4.6. Побудований прогноз

Також в системі реалізовано другий розділ, під назвою «Історичні дані», у якому можна переглянути гістограму по історичним даним врожайності культур по областях, на Рис.4.7. зображено меню вибору параметрів.

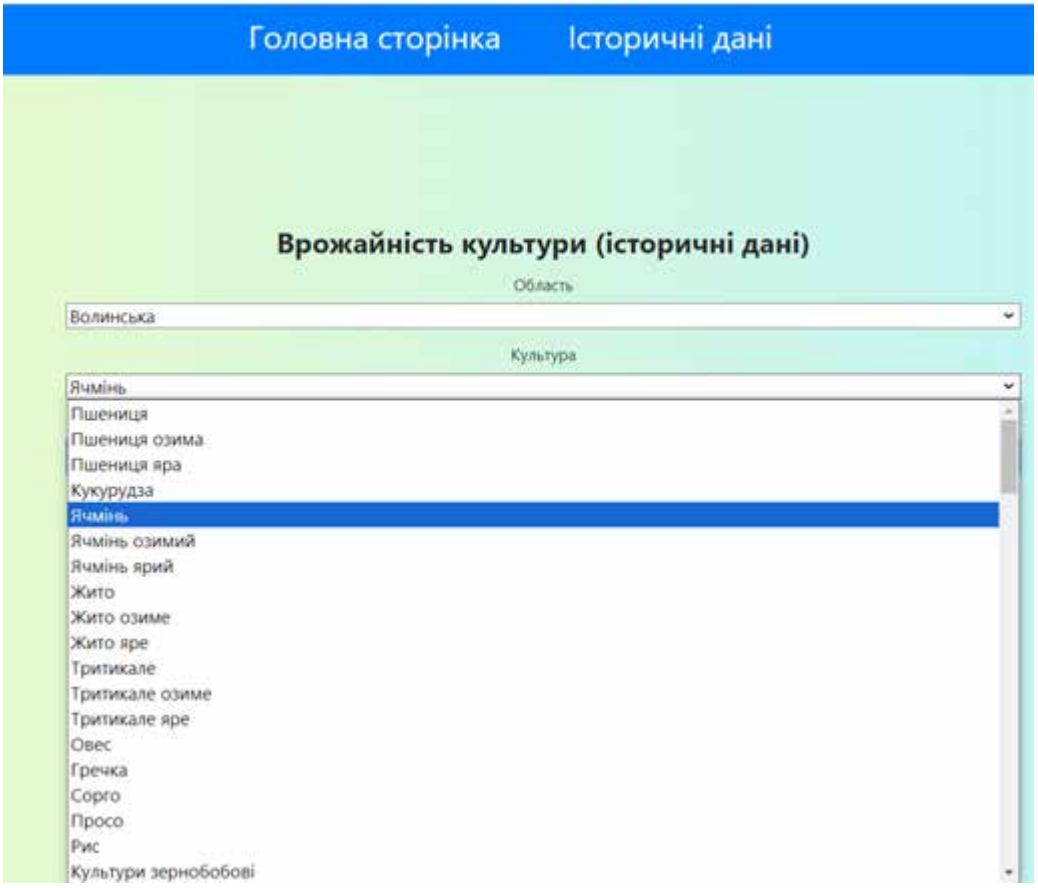

Рис. 4.7. Вибір параметрів для побудови гістограми

Обравши параметри, користувачу відображається створена гістограма по врожайності культури в проміжку 2017-2023 років, на Рис.4.8. зображено вигляд сторінки.

Рис. 4.8. Створена гістограма

4.2 Аналіз точності та ефективності інформаційно-аналітичної системи

Метою аналізу точності та ефективності інформаційно-аналітичної системи є отримання глибокого розуміння її функціонування та визначення наскільки задовольняються потреби користувачів. Зокрема, об'єктами аналізу є: Оцінка точності моделі, визначення наскільки прогнози є точними, за допомогою використання метрик точності, які ми надалі розглянемо.

Для оцінки точності моделі прогнозу врожайності на основі різних показників, включаючи кліматичні умови, історичні дані врожайності та якість ґрунту, можна використовувати різні метрики відповідності, такі як середньоквадратична помилка (MSE), середня абсолютна помилка (MAE), коефіцієнт детермінації (R-squared).Розглянемо детальніше кожен з цих методів.

Середньоквадратична помилка (MSE model) є ключовим показником, що застосовується для оцінки точності моделей в різних галузях, таких як статистика, машинне навчання, фізика, економіка та інші наукові області. Ця метрика обчислює середнє значення квадратів різниці між прогнозованими і фактичними значеннями. MSE вимірює, наскільки "далеко" прогнозовані значення відхиляються від фактичних даних. Помилки, які є великими за абсолютним значенням, виявляються ще більшими після піднесення до квадрата. Середньоквадратична помилка (MSE) може мати будь-які додатні значення, і чим вона менше, тим краще працює модель. Високе значення MSE свідчить про значну середню похибку моделі[27].

Формула для обчислення середньоквадратичної похибки має наступний вигляд:

$$MSE\ model = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, \quad (4.1)$$

Де:

- y_i- фактичне значення
- $\hat{y}_i$- прогнозоване значення
- N - кількість спостережень

MSE допомагає оцінити модель, порівнюючи середньоквадратичну різницю між її прогнозованими та фактичними значеннями. Проте, для більш об'єктивного аналізу моделі треба використовувати інші метрики разом із MSE, особливо коли великі викиди мають вагому роль у вашій конкретній задачі.

Існує також метод MSE (baseline), який відображає рівень помилки у прогнозі, який можна досягти, використовуючи простий, зазвичай статистичний метод прогнозування без залучення складних аналітичних прийомів чи складних моделей. Наприклад, цей метод може передбачати майбутнє значення, базуючись на середньому арифметичному чи іншому простому статистичному показнику, обчисленому на основі історичних даних.

Визначальною метою MSE baseline є визначення, чи є ваша більш складна модель прогнозу кращою в порівнянні з таким простим базовим методом. Якщо MSE model вашої моделі менший, ніж MSE baseline методу, це свідчить про те, що ваша модель працює краще і надає більш точні прогнози, ніж простий метод.

Формули MSE model та MSE baseline дуже схожі, проте мають одну дуже суттєву відмінність, представимо формулу MSE baseline:

$$MSE\ model = \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y}_i)^2, \quad (4.2)$$

Де:

- y_i- фактичне значення
- $\bar{y}_i$- середнє значення спостережень
- N - кількість спостережень

Різниця полягає лише у знаках $\bar{y}_i$ та $\hat{y}_i$, проте результат виходить дуже різний, і говорить багато про що.

Дослідивши дані методи, стає очевидним, що для обрахунку точності моделі у розроблюваній системі, його буде недостатньо, проте вони будуть складовою частиною, яка буде використовувати також інші моделі, які разом зможуть дати необхідний результат, тож розглянемо їх далі.

Середня абсолютна помилка (Mean Absolute Error, MAE) - це метрика, яка використовується для вимірювання точності моделі прогнозування. Вона обчислюється як середнє арифметичне абсолютних значень різниці між

фактичними та прогнозованими значеннями. Формула для обчислення MAE виглядає:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|, \quad (4.3)$$

Де:

- y_i- фактичне значення
- $\hat{y}_i$- прогнозоване значення
- N - кількість спостережень

MAE виражається в одиницях вимірювання аналізованої величини, і чим менше значення MAE, тим вища точність моделі. Однією з переваг MAE є її більша стійкість до викидів порівняно з MSE, оскільки вона не збільшує помилки через піднесення до квадрату різниць.

Розглянувши другий метод, який як стало зрозуміло трохи краще може відображати точність , можемо перейти то фінальної частини, а саме розгляду коефіцієнту детермінації.

Коефіцієнт детермінації, позначений як R^2, є статистичним показником, який використовується для оцінки того, наскільки добре прогнозована модель відповідає фактичним даним. Цей показник показує частку варіації в фактичних даних, яка може бути пояснена прогнозованими значеннями моделі. У більш загальному вигляді, R^2 визначає, наскільки добре модель підходить для ваших даних і як вона точно передбачає майбутні результати. Це і є те що потрібно для вирішення задач системи, тому детальніше розглянемо цей метод.

Коефіцієнт детермінації може приймати діапазон значень від 0 до 1. Що на відміну від попередніх методів , дає нам об’єктивно і точно оцінити модель. Якщо інтерпретувати ці значення, то можна сказати що коли R^2=0, то модель не пояснює жодної варіації, і прогноз є абсолютно некорисним. Якщо ж R^2=1, то модель ідеально пояснюю варіацію, і робить прогнози дуже точними. Але як показує практика в більшості випадків значення становлять в межах від

$0<R^2<1$, і це говорить що модель пояснює частку варіації в даних, і точність прогнозу корелюється з отриманим значенням.

Формула для обчислення R^2 виглядає наступним чином:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y}_i)^2}, \quad (4.4)$$

Де:

- y_i- фактичне значення
- $\hat{y}_i$- прогнозоване значення
- $\bar{y}_i$- середнє значення спостережень
- N - кількість спостережень

Проте розглянувши методи MAE та MSE, зрозуміти та реалізувати формулу R^2 можна простіше, представивши її у наступному вигляді, які включають у себе формули (4.1) та (4.2):

$$R^2 = 1 - \frac{MSE(model)}{MSE(baseline)} \quad (4.5)$$

Отже, розглянувши способи для оцінки точності моделі , можна сказати що, R^2 надає об'єктивну оцінку того, наскільки добре модель відповідає даним і як точно вона прогнозує результати, тому з впевненістю можна сказати, що він є найбільш оптимальним показником для оцінки якості моделі прогнозування, та не дарма був обраний для використання у створюваній системі.

4.3 Результати тестування системи

Після того як було розглянуто основні методи тестування системи, перейдемо до розгляду цього питання на практиці, та протестуємо розроблену систему. Для початку розглянемо ручне тестування, та самостійно перевіримо

роботу системи, функціональність кнопок та сумісність системи з мобільними пристроями. Для цього скористаємось мобільним телефоном, та відкриємо сайт у браузері Google Chrome, та перевіримо відображення всіх елементів.

На Рис. 4.9 зображено головне вікно, з можливістю вибору області, культури, та типу ґрунту. Як можна побачити всі елементи добре адаптовані під мобільний пристрій, та функціонують.

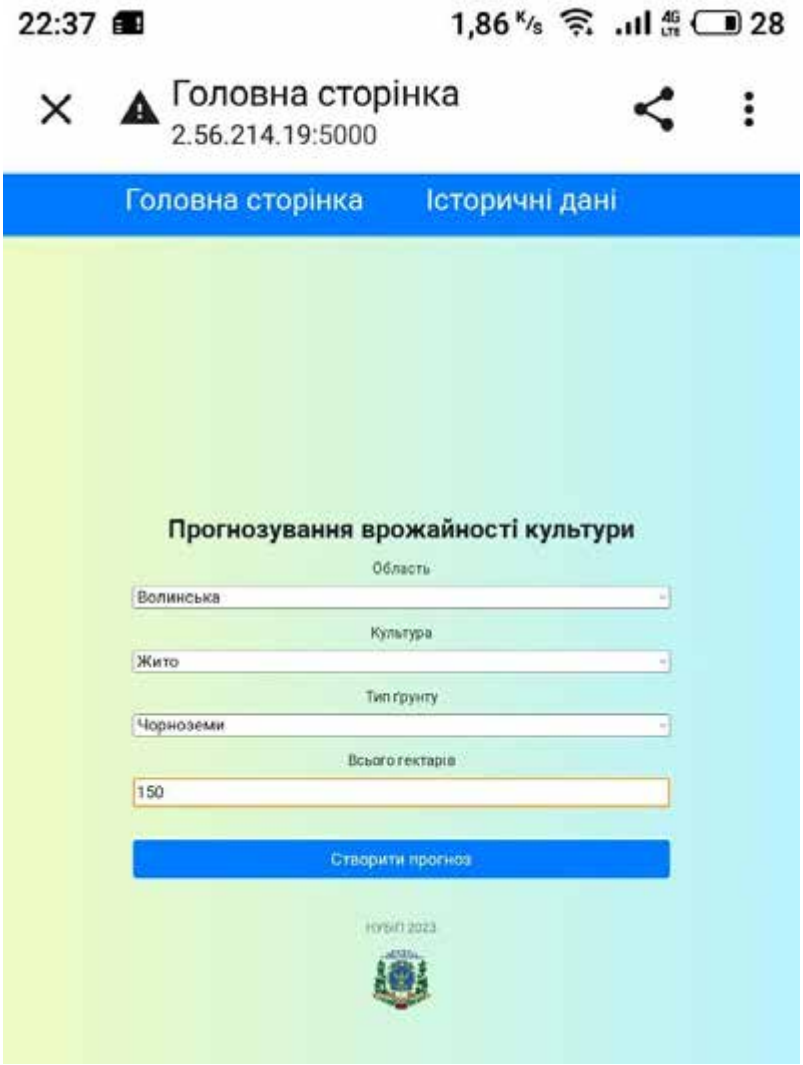

Рис. 4.9. Головне меню прогнозування

На Рис. 4.10 Зображено побудований графік прогнозу врожайності жита, для Волинської області. Всі елементи добре поміщаються на екрані, коректно відображаються, а всі кнопки працюють.

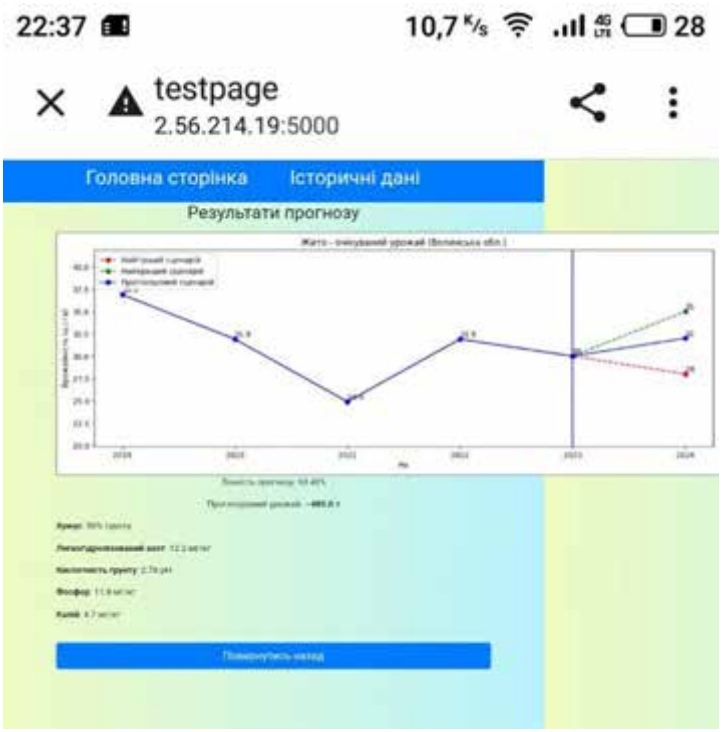

Рис. 4.10. Побудований прогноз

Упевнившись у коректному відображені головної сторінки, розглянемо пункт історичні дані, побудуємо гістограму, та протестуємо коректність відображення. На Рис.4.11 Зображено пункт меню «Історичні дані».

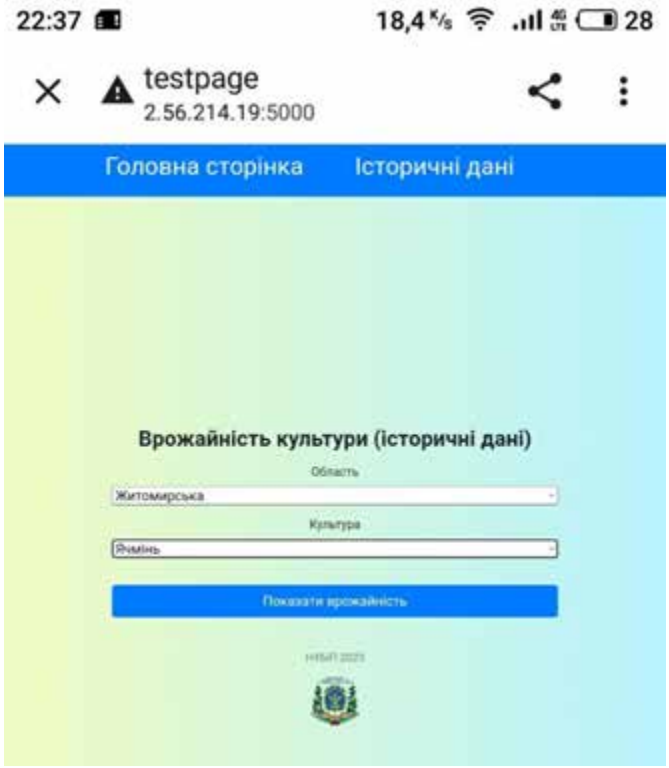

Рис. 4.11. Меню історичних даних

Бачимо що все відображається та працює вірно. Перейдемо до побудови гістограми врожайності , яка побудована на Рис.4.12

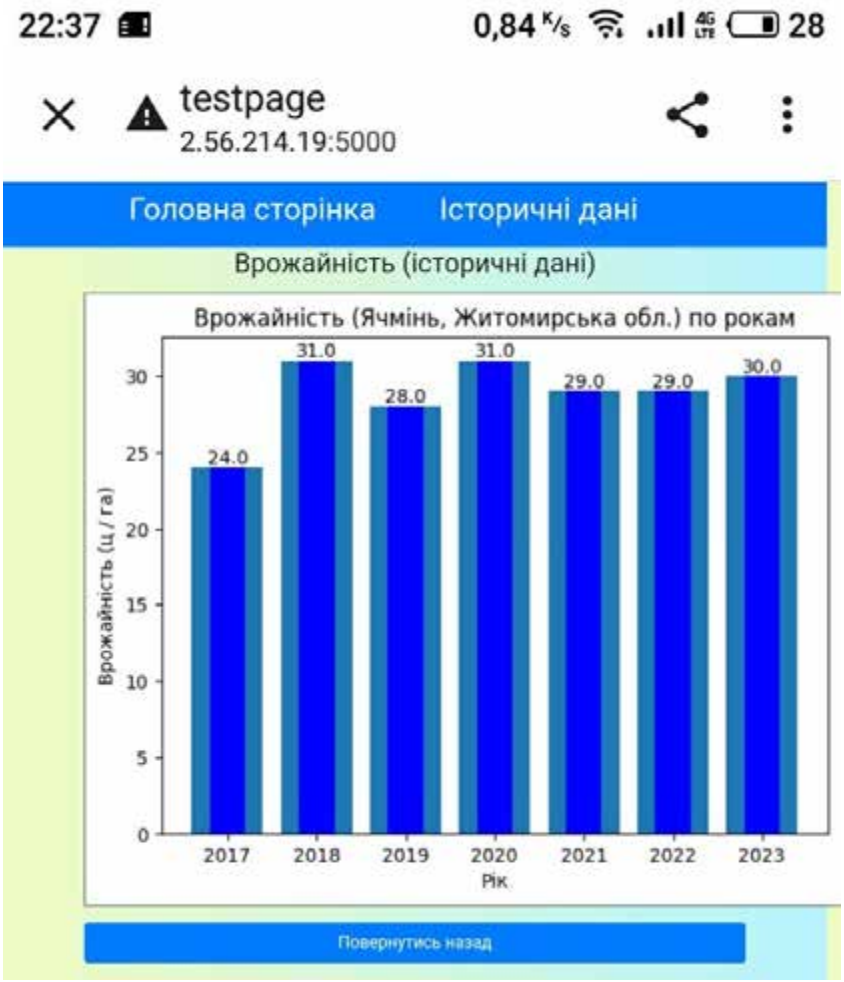

Рис. 4.12. Гістограма сторичних даних врожайності

Побудувавши гістограму по врожайності ячменю у житомирській області, бачимо, що все працює правильно, всі елементи на сторінці відображаються коректно.

Протестувавши систему вручну, перейдемо до автоматичного тестування. Для цього як зазначалося раніше було використано інструмент PageSpeed Insights. Одразу слід відзначити, що можна дослідити систему як для використання з мобільних пристроїв, так і з персональних комп’ютерів. Користуватись даним інструментом досить просто, слід лише надати посилання на веб сайт, і одразу отримати шкалу ефективності веб сайту. На рис.4.13 зображено ефективність роботи веб сайту з мобільного пристрою:

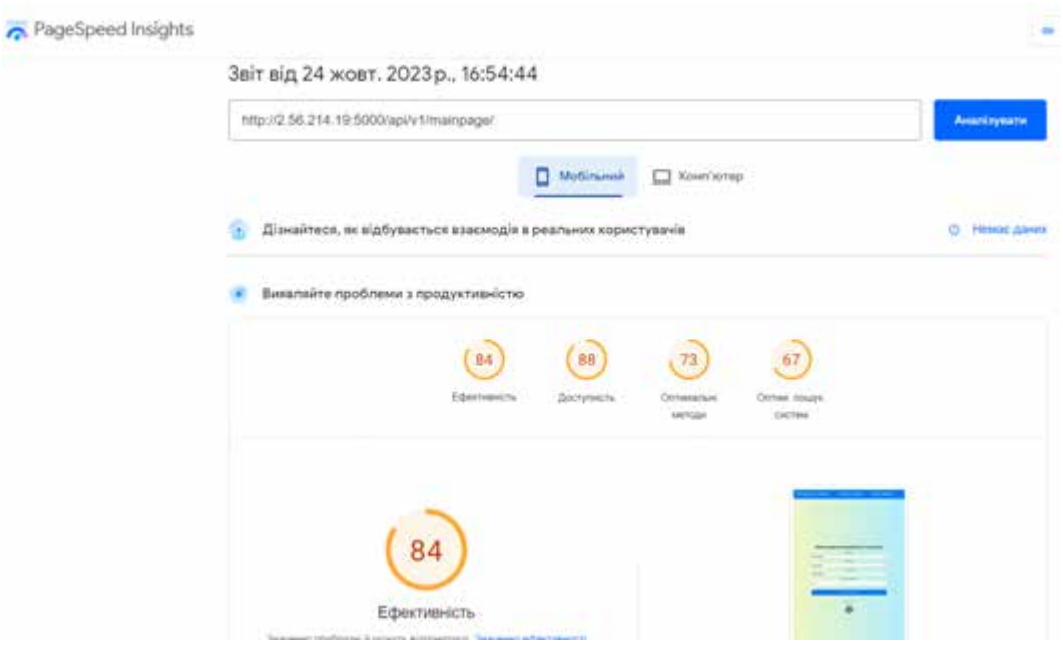

Рис. 4.13. Ефективність роботи сайту з мобільного пристрою

Наступним кроком є аналіз таких показників як: перше малювання змісту, найбільшого малюнку, загальний час блокування, зміщення макета та індекс швидкості. Як видно з результатів на Рис.25 всі показники є в межах норми, окрім найбільшого малюнку, час на його завантаження є досить великим.

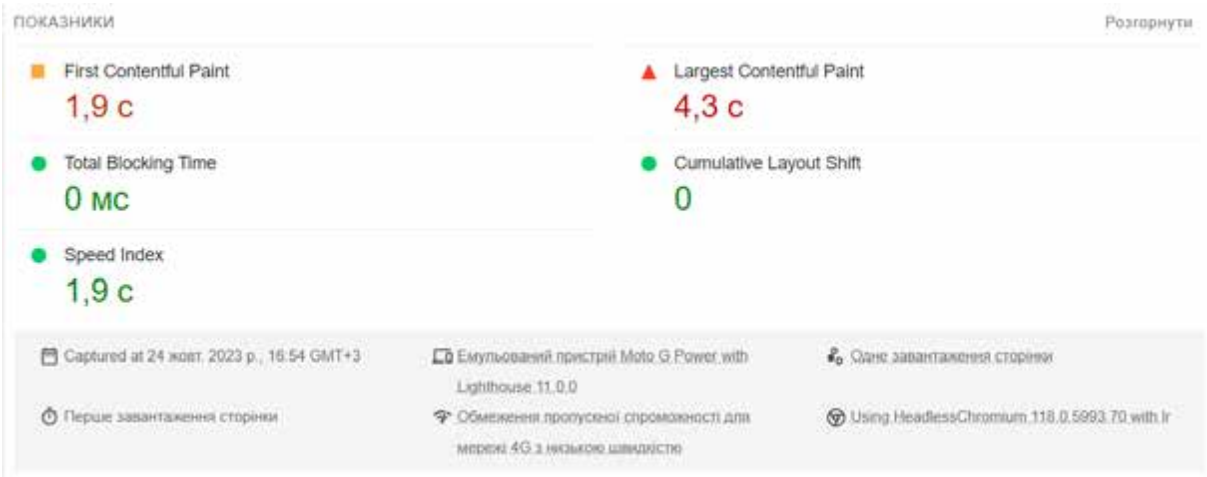

Рис. 4.14. Основні показники системи

Наступними кроками тестування є тестування можливостей, надано рекомендації щодо розміру зображень, формату, оптимізації коду, стиснення тексту та іншого. Також проводиться діагностика коду, яка дозволяє виявляти відсутність певних елементів, результати тестування представлені на Рис.26

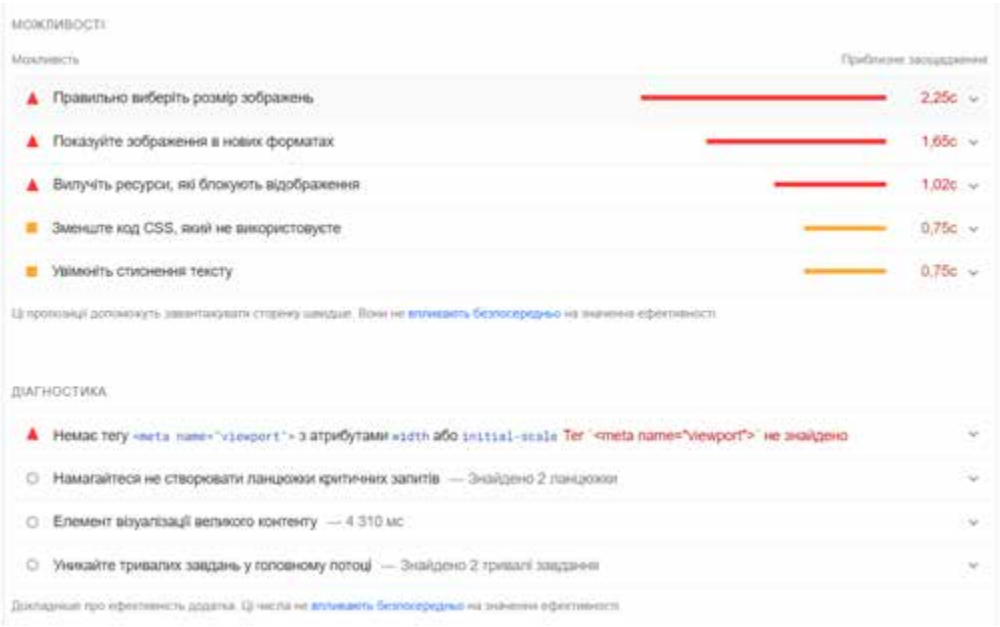

Рис. 4.15. Діагностика та основні можливості системи

Протестувавши систему для мобільних пристроїв, перейдемо для тестування робото спроможності системи використовуючи її на комп’ютері, та порівняємо отримані результати.

На Рис.27 побачимо загальну ефективність системи:

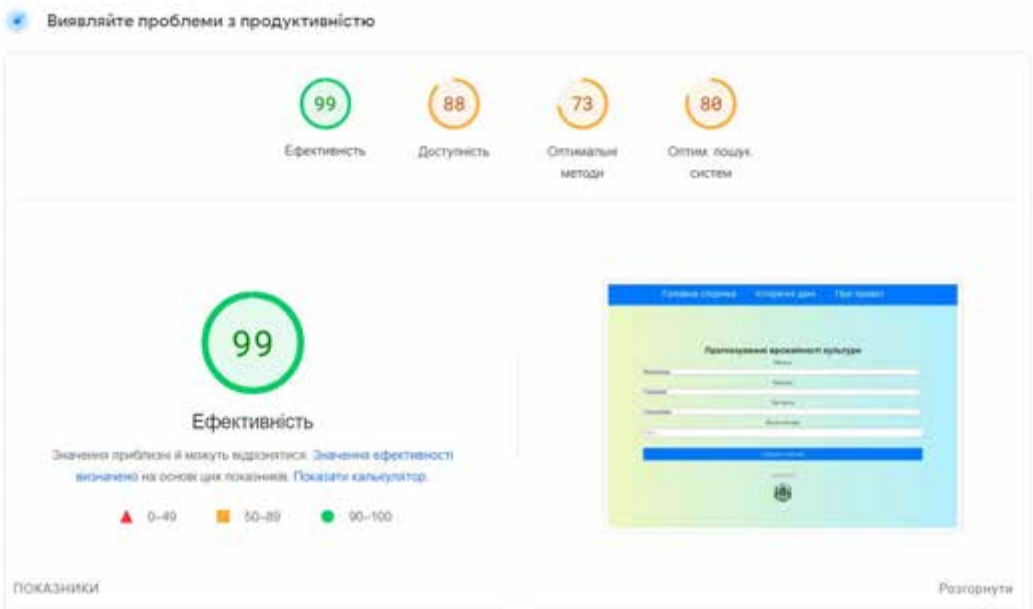

Рис. 4.16. Загальна ефективність системи на ПК

Виміряємо основні показники роботи системи, результат зображено на Рис.28.

Рис. 4.17. Основні показники роботи системи

Надалі було перевірено можливості системи, протестовано час завантаження зображень, та досліджено ресурси які блокують зображення. Розділ діагностика надає ті самі рекомендації що і на мобільній версії, оскільки код програми в залежності від пристрою яким користується користувач не змінюється.

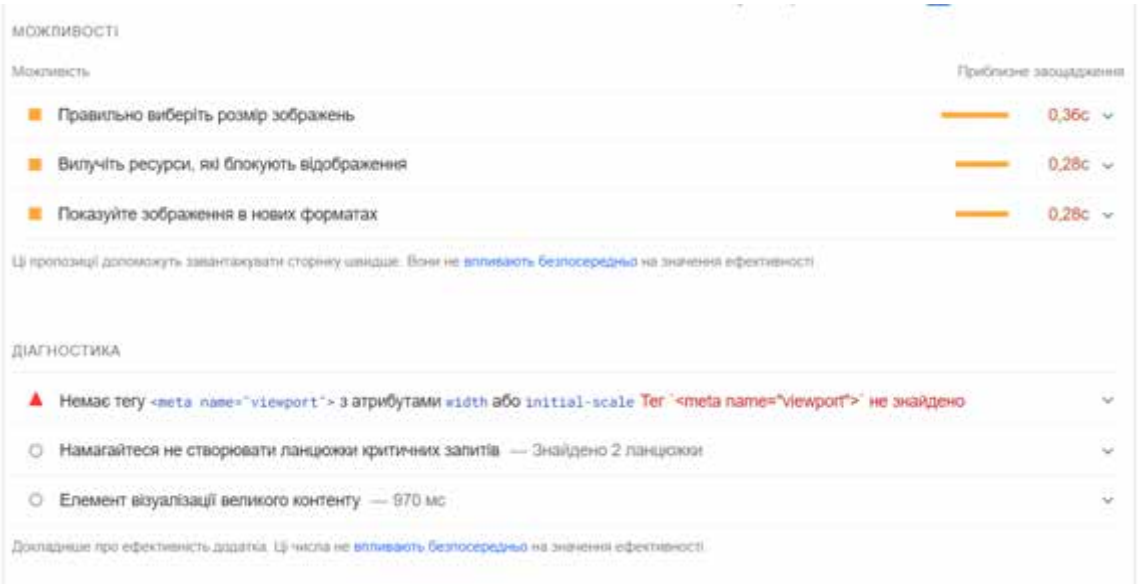

Рис. 4.18. Можливості да діагностика системи

Виконавши тестування системи, можна зробити певні висновки. Порівнюючи ручне та автоматичне тестування, можна відразу сказати, що автоматичне тестування є більш повноцінним, охоплює набагато більше аспектів, а сам процес тестування проходить набагато швидше та зручніше.

Порівнюючи автоматичне тестування для мобільних пристроїв та комп’ютерів, з отриманих результатів можна сказати, що ефективність системи при роботі з комп’ютера є більшою, проте і з мобільного пристрою вона є цілком задовільною, для нормальної роботи з системою. В основному при роботі з ПК, швидше завантажуються елементи, такі як зображення та графіки, що робить роботу з програмою більш комфортною, проте це не є критично.

Отже протестувавши систему, можна з впевненістю сказати, що вона пройшла всі перевірки, працює коректно та швидко, підходить для роботи на різних пристроях та системах, що задовольняє поставлені вимоги до впровадження системи.

4.4 Дослідження використання методу асоціативних правил

Метод асоціативних правил - це аналітичний метод, який використовується для виявлення зв'язків між різними атрибутами або подіями у даних. Цей метод широко використовується в галузі аналізу даних та машинного навчання і застосовується у багатьох різних галузях. Вони особливо корисні для виявлення закономірностей в ситуаціях, коли дані здавалися незалежними, таких як у реляційних базах даних і базах даних транзакцій. Процес використання правил асоціації часом називається "видобутком асоціативних правил" або "асоціаційним аналізом". Правила асоціації формуються з наборів, що складаються з двох чи більше елементів. У зв'язку з великою кількістю можливих правил, завдання може стати дуже складним. Тому правила асоціації зазвичай створюються на основі даних, де зв'язки між елементами чітко виражені і можуть бути зрозуміло інтерпретовані.

Метод асоціативних правил використовується для виявлення складних зв'язків у великих наборах даних та може бути корисним інструментом в аналізі даних, прогнозуванні та прийнятті рішень. У контексті системи прогнозування врожайності сільськогосподарських культур, метод асоціативних правил може

допомогти виявити залежності між різними факторами та покращити точність прогнозу.[28]

Отже, проаналізувавши метод асоціативних правил можна сказати, що він є корисним засобом для розкриття зв'язків в даних і розробки результативних бізнес-стратегій.

Розглянемо застосування цього методу на практиці, для цього було використано середовище Visual Studio, та інструменти для проведення інтелектуального аналізу даних. На основі створеного OLAP-кубу, реалізуємо метод асоціативних правил, зображений на Рис. 4.19.

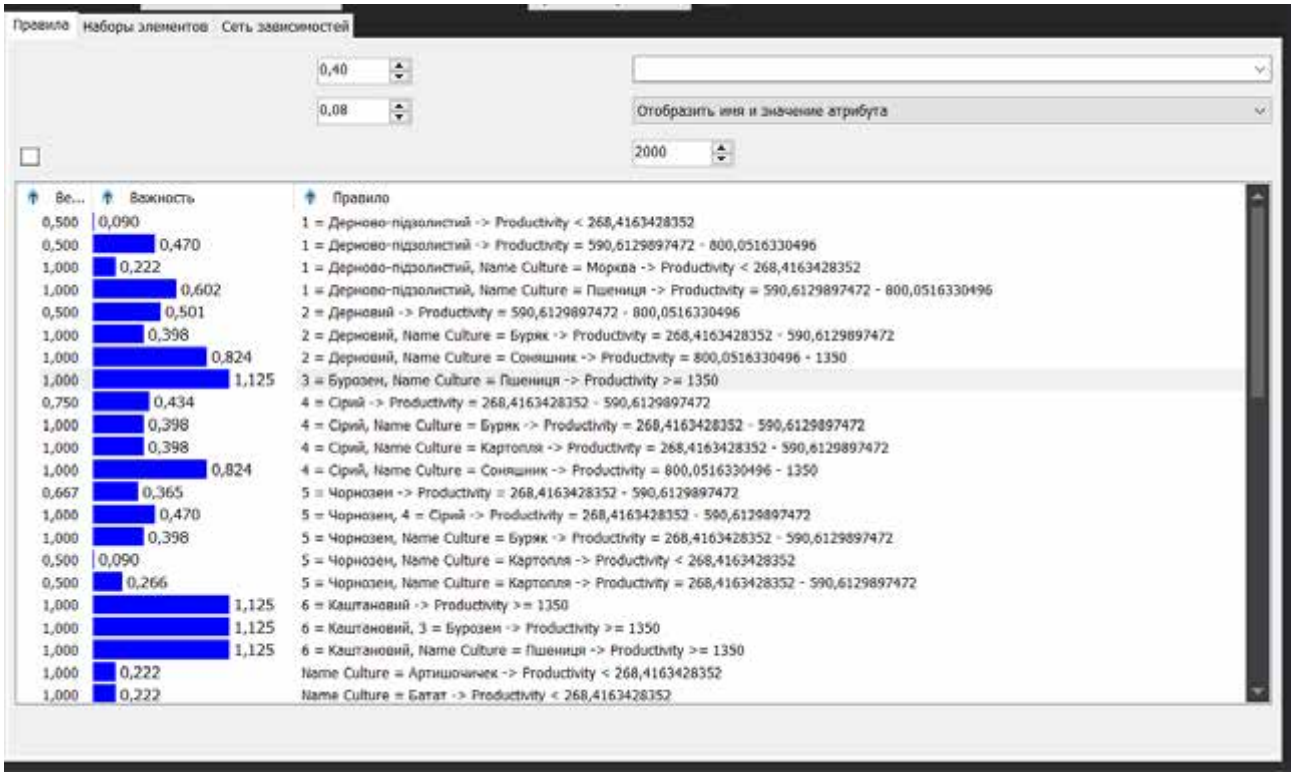

Рис. 4.19 Знайдені правила на основі взаємозалежностей

Як видно з отриманих результатів, було знайдено досить багато взаємозв’язків між даними. Наприклад як в залежності від типу грунту, змінюється врожайність культури.

Також було побудовано мережу залежностей, на якій графічно представлено зв’язки між даними(Рис.4.20).

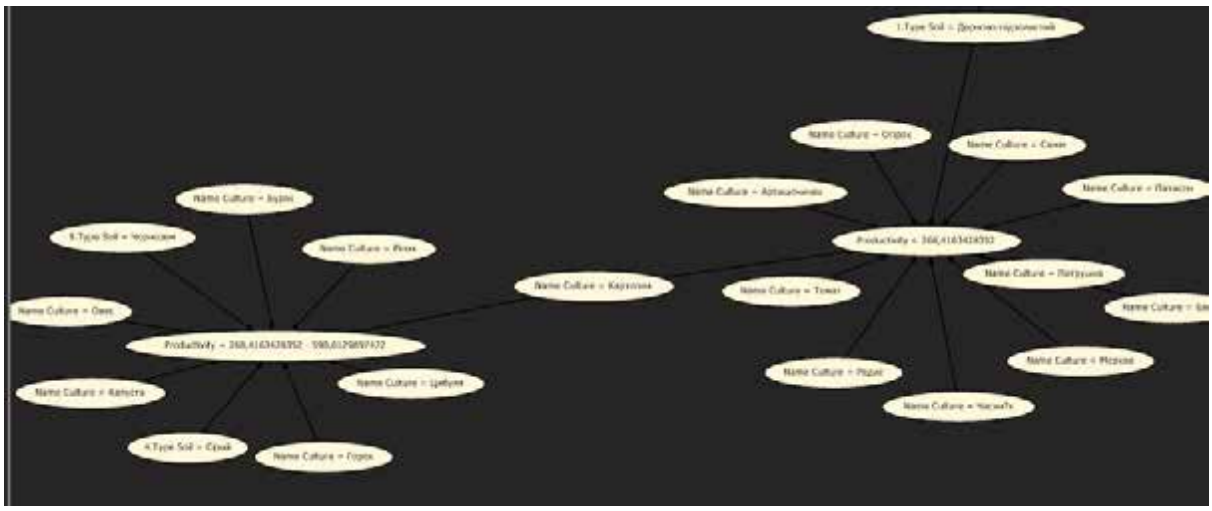

Рис.4.20 Сітка залежностей між даними

4.5 Можливості подальшого розвитку та розширення функціональності системи

Система має потенціал для подальшого розвитку та розширення функціональності, що дозволить відповідати змінюваним потребам користувачів та враховувати розвиток технологій. Система може розширюватися шляхом додавання нових функцій та модулів. Такими модулями як, внесення інформації про поле. Планується додати інтерактивну карту, на якій користувач зможе обирати територію, де розміщене його поле, і після чого зможе вносити дані про цю ділянку, таких як історія обробки, використані ресурси, та інші. Внесена інформація може відображатись прямо на карті, що покращить сприйняття користувачу, особливо коли у нього багато ділянок. Це в свою чергу дасть можливість точно відстежувати погодні умови, кількість опадів, якість ґрунту, вести облік врожайності. Користувачі можуть використовувати дані про свої ділянки для планування сільськогосподарських робіт, розрахунку витрат та оптимізації виробництва, що буде важливим модулем у системі в майбутньому.

Перегляд історії дощу та спеки: Функція перегляду історії погодних умов може бути корисною для аграрних підприємств, що прагнуть вивчати та аналізувати погодні тенденції та їх вплив на сільське господарство. Цей модуль може допомогти користувачам отримувати статистику та графіки стосовно дощів та спеки протягом певного періоду. Непогано було б у цьому модулі

отримувати дані з інших джерел прогнозу погоди, та надавати користувачу повідомлення про сильну спеку, або сильну зливу, аби він міг завчасно прийняти певні дії для збереження врожаю.

Модуль відображення актуальної ціни на продукцію може стати важливим доповненням до системи. Він надасть користувачам доступ до інформації про поточні ціни на продукцію. Для підприємців це відкриє багато нових можливостей, посприяє прийняттю рішень щодо виробництва. Також можливість відстеження коливання ціни може допомогти фермеру та підприємствам у плануванні своєчасного збуту своєї продукції, переглянувши наприклад історичні дані по ціні культури, або ціну у даний момент часу. Важливо буде додати можливість налаштування сповіщень про зміни цін на конкретну продукцію. Це дозволить користувачам бути завжди в курсі актуальних пропозицій. Використання візуальних елементів, таких як графіки і діаграми, для представлення динаміки цін може робити інформацію більш зрозумілою та привабливою для користувачів. З додаванням нових модулів, важливо буде покращувати дизайн додатку, тому , що це може значно покращити його привабливість та зручність для майбутніх користувачів. Планується додавати нові високоякісні графічні елементи, картинки , значки. Також в майбутньому буде додано анімації та плавні переходи між сторінками, адже це покращить комфорт користування , та зробить його більш плавним.

У майбутньому можна буде розширити функціональність системи, розробивши власний мобільний додаток, який повинен бути доступний на платформах Android та IOS, і надасть користувачам зручний спосіб перегляду інформації про свої поля, навіть не маючи доступу до інтернету в даний момент часу. Це буде зручно для фермерів, які знаходячись прямо на полі, захочуть внести дані у систему, або занотувати щось важливе, що потім можна буде опрацювати. Найголовніше це дасть офлайн доступ, що дуже зручно, бо на багатьох полях погане покриття мережі. Слід також відзначити про взаємодію з апаратними функціями смартфону, наприклад такими як камера , або геолокація, яка може бути використана для більш точної навігації. Також з мобільного

застосунку зручно буде відправляти користувачам сповіщення, що допоможе залучити користувача до роботи.

Для розробника такий додаток може бути цікавим в плані монетизації, адже може приносити додатковий прибуток через рекламу, та інші види монетизації. Це в свою чергу дасть можливість дізнаватись кількість активної аудиторії, яка працює у мобільному додатку. Загалом, розробка мобільного додатку може суттєво підвищити якість взаємодії зі споживачами та забезпечити їм зручний інструмент для взаємодії з вашими послугами або продуктом.

ВИСНОВКИ

У ході виконання кваліфікаційної магістерської роботи було розроблено інформаційно-аналітичну систему прогнозування врожайності сільськогосподарських культур, досліджено та проаналізовано предметну область, поставлено технічні вимоги, зібрано необхідні дані для проведення прогнозування, створено та розроблено модель системи. На етапі планування було обрано інструменти для розробки системи, зберігання даних та проведення тестування системи. Дослідження полягали у визначенні впливу параметрів ґрунту та кліматичних умов, на майбутню врожайність сільськогосподарських культур.

Застосунок був реалізований за допомогою використання клієнт-серверної архітектури. Серверна частина була написана за допомогою мови Python та мікрофреймворка Flask, а клієнтська частина за допомогою мови HTML та CSS. Для зберігання даних була використана СУБД PostgreSQL.

В процесі дослідження було проведено глибокий аналіз та обробку великого обсягу даних, що стосуються вирощування різних видів сільськогосподарських культур. Цей аналіз включав в себе збір та агрегацію даних про врожайність, кліматичні умови, якість ґрунту, та багато інших факторів, що впливають на результати вирощування сільськогосподарських культур. Результати дослідження були представлені у розділі 4.

Було впроваджено алгоритм лінійної регресії, який став ключовою складовою інформаційно-аналітичної системи для прогнозування врожайності сільськогосподарських культур. Під час розробки алгоритму лінійної регресії, було враховано різні аспекти, такі як вагові коефіцієнти для кожного фактора, які дозволили підібрати найкращу модель для прогнозування врожайності.

В результаті, інформаційно-аналітична система прогнозування врожайності сільськогосподарських культур виконує поставленні завдання, використовує сучасні методи аналізу даних, включаючи лінійну регресію та статистичні методи, для прогнозування врожайності. Система була ретельно

перевірена та піддана тестуванню з використанням реальних даних, що дозволило підтвердити її ефективність та точність. У перспективі система має можливості подальшого розвитку та розширення функціональності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бабенко М.Д. Активізація і підвищення ефективності розвитку агропромислової інтеграції та кооперації в АПК регіону: дис. канд. екон. наук : 08.07.02. Дніпропетровськ, 2012. 220с.
2. Каленська, С. М. Прогнозування і програмування врожаїв сільськогосподарських культур / С. М. Каленська, В. А. Мокрієнко, М. Я. Дмитришак, А. В. Юник, Є. В. Качура. – Київ: Видавничий Центр НУБіП України. – 28 с.
3. Smith, J. K., & Johnson, A. B. (2020). Factors Affecting Crop Yield: A Comprehensive Review. Agricultural Science Journal, 45(2), 123-138.
4. Hatfield, J. L., & Prueger, J. H. (2015). Temperature extremes: Effect on plant growth and development. Weather and Climate Extremes, 10, 4-10
5. Brady, N. C., & Weil, R. R. (2016). The Nature and Properties of Soils (15th ed.). Pearson.
6. Device requirements for using web applications [Електронний ресурс] – Режим доступу до ресурсу: https://support.microsoft.com/uk-ua/topic/030f95d4-9589-5190-619d-1f6ea218569b
7. James Rumbaugh, Ivar Jacobson, Grady Booch (1999). The unified modeling language reference manual Addison Wesley Longman Inc. – 8 с.
8. Антонов В.М. Сучасні комп’ютерні мережі –К.: «МК-Прес», 2005–480с.
9. Ситник Н. В. Проектування баз та сховищ даних: Навч. посібник. – К.: КНЕУ, 2004. – 348 с.
10. Державна служба статистики України [Електронний ресурс] –Режим доступу до ресурсу: https://www.ukrstat.gov.ua/
11. Eric Sperley, Enterprise Data Warehouses: Planning, Building, and Implementation, Volume 1 "M.: Williams Publishing House, 2001. – 400 pages"
12. What is OLAP? [Електронний ресурс] –Режим доступу до ресурсу:https://www.ibm.com/topics/olap

13. What is SSAS (SQL Server Analysis Services)? ? [Електронний ресурс]–Режим доступу до ресурсу:https://www.solarwinds.com/resources/it-glossary/ssas
14. SQL Server Integration Services [Електронний ресурс] –Режим доступу до ресурсу:https://learn.microsoft.com/en-us/sql/integration-services/sql-server-integration-services?view=sql-server-ver15
15. Why VS Code? [Електронний ресурс]–Режим доступу до ресурсу:https://code.visualstudio.com/learn
16. pgAdmin [Електронний ресурс] –Режим доступу до ресурсу:https://www.pgadmin.org/
17. "The Importance of Using Frameworks in Software Development" [Електронний ресурс]–Режим доступу до ресурсу:https://www.geeksforgeeks.org/the-importance-of-using-frameworks-in-software-development/
18. Flask [Електронний ресурс]–Режим доступу до ресурсу:https://flask.palletsprojects.com/en/2.1.x/
19. Mozilla Developer Network (MDN)- HTTP [Електронний ресурс]–Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview
20. Psycopg – PostgreSQL database adapter for Python [Електронний ресурс]–Режим доступу до ресурсу: https://www.psycopg.org/docs/
21. Using Matplotlib- [Електронний ресурс]–Режим доступу до ресурсу:https://matplotlib.org/stable/users/index.html
22. Learn to style HTML using CSS [Електронний ресурс]–Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Learn/CSS
23. Introduction to SQL [Електронний ресурс]–Режим доступу до ресурсу:https://www.w3schools.com/sql/sql_intro.asp
24. Пєтух А.М. Бази даних. Мови запитів, управління транзакціями, розподілена обробка даних [Електронний навчальний посібник] /А.М. Пєтух, О.В. Романюк, О.Н. Романюк. – Режим доступу: https://web.posibnyky.vntu.edu.ua/fitki/11petuh_bazdanyh_movy_zalitiv/

25. PostgreSQL: The World's Most Advanced Open Source Relational Database [Електронний ресурс]–Режим доступу до ресурсу:https://www.postgresql.org/
26. PageSpeed Insights [Електронний ресурс]–Режим доступу до ресурсу:https://pagespeed.web.dev/
27. Гончарук А.Г.Основи статистики: Навч. посібник.- К.: Центр навчальної літератури, 2004.-125с.
28. Agrawal, R., Imielinski, T., & Swami, A. N. (1993). Mining association rules between sets of items in large databases."

ДОДАТОК А

КОД РЕАЛІЗАЦІЇ ВЕБ-ДОДАТКУ ЗА ДОПОМОГОЮ FLASK

Сторінок-3

```
from flask import Flask, render_template, request
from methods import psycopg as psql
from methods import plot_drawing as plt

app = Flask(__name__)

@app.route('/api/v1/mainpage/',  methods=['GET'])
def mainpage():
    error = None
    if request.method == 'GET':
        culture_dict = psql.ret_culture_yield()
        region_dict = psql.ret_region_prod()
        soil_dict = ['Чорноземи', 'Дерново-підзолисті', 'Сірі лісові']

        return render_template('main.html', culture_dict = culture_dict, region_dict =
region_dict, soil_dict = soil_dict)

@app.route('/api/v1/about/',  methods=['GET'])
def about():
    error = None
    if request.method == 'GET':
        return render_template('about.html')

@app.route('/api/v1/mainpage_bar/',  methods=['GET'])
def mainpage_bar():
    error = None
    if request.method == 'GET':
```

```
        culture_dict = psql.ret_culture_yield()
        region_dict = psql.ret_region_prod()
        soil_dict = ['Чорноземи', 'Дерново-підзолисті', 'Сірі лісові']

        return render_template('main_bar.html', culture_dict = culture_dict, region_dict
= region_dict, soil_dict = soil_dict)

@app.route('/api/v1/results/',  methods=['GET'])
def results():
    error = None
    if request.method == 'GET':
        culture_dict = psql.ret_culture_yield()
        region_dict = psql.ret_region_prod()

        culture_name = request.args.get('culture_name','')
        region = request.args.get('region', '')

        ga = float(request.args.get('ga', 100.0))

        cond = psql.ret_cond(region)

        plt_dict = plt.gen_plot(culture_name, region)
        print(plt_dict)
        return render_template('results.html', culture_dict = culture_dict, region_dict =
region_dict, res = plt_dict, ga = ga, cond = cond, prob = plt.rd_prognoz())

@app.route('/api/v1/results_bar/',  methods=['GET'])
def results_bar():
    error = None
```

```
    if request.method == 'GET':
        culture_dict = psql.ret_culture_yield()
        region_dict = psql.ret_region_prod()

        culture_name = request.args.get('culture_name','')
        region = request.args.get('region', '')

        ga = float(request.args.get('ga', 100.0))

        cond = psql.ret_cond(region)

        plt_dict = plt.gen_plot_bar(culture_name, region)

        return render_template('results_bar.html', culture_dict = culture_dict,
region_dict = region_dict, res = plt_dict, ga = ga, cond = cond, prob =
plt.rd_prognoz())


@app.route('/api/v1/get_predict/', methods=['GET'])
def get_predict():
    error = None
    if request.method == 'GET':
        culture_name = request.args.get('culture_name','')
        region = request.args.get('region', '')

        predict_method = request.args.get('predict_method', 'ARIMA')

        data_dict = psql.ret_culture_yield()

        return f'<p>reg: {region}, cult: {culture_name}, method: {predict_method}</p>'
```

ДОДАТОК Б

КОД РЕАЛІЗАЦІЇ ПОБУДОВИ ГРАФІКІВ

```
import matplotlib.pyplot as plt
import random as rd
import math
from methods import psycopg as psql
from methods import hashtable as ht
hash_table = []
def rd_prognoz():
    return '90%'
def gen_plot_bar(culture_name=None, region=None):
    ht_table = ht.ret_hash()
    f_y = math.floor(float(psql.ret_culture_yield_single(culture_name))) +
ht_table[len(region)] * 2.0
    f_y_perc = ht_table[len(culture_name):len(culture_name)+7]
    # f_y_perc = -1
    print(f_y_perc)
    data = {
        '2017':math.ceil(f_y + f_y_perc[0]),
        '2018':math.ceil(f_y + f_y_perc[1]),
        '2019':math.ceil(f_y + f_y_perc[2]),
        '2020':math.ceil(f_y + f_y_perc[3]),
        '2021':math.ceil(f_y + f_y_perc[4]),
        '2022':math.ceil(f_y + f_y_perc[5]),
        '2023':f_y}
     # for i in data.items():
    #     if i < 0:
    #         i = rd.randint(1,3)
    years = list(data.keys())
    values = list(data.values())
    fig, ax = plt.subplots()
    bar_container = ax.bar(years, values)
```

```
            ax.bar_label(bar_container, fmt=lambda x: f'{x}')
               # creating the bar plot
            plt.bar(years, values, color ='blue',
                    width = 0.4)
            plt.xlabel("Рік")
            plt.ylabel("Врожайність (ц / га)")
            plt.title(f"Врожайність ({culture_name}, {region} обл.) по рокам")
            # plt.show()
            plt.savefig('./static/plot_bar.png', bbox_inches='tight')
        def gen_plot(culture_name=None, region=None):
            plt.figure().set_figwidth(15)
            ht_table = ht.ret_hash()
            f_y = math.floor(float(psql.ret_normalized_predict(culture=culture_name,
region=region).get('c_prognoz')))
            f_y_perc = ht_table[len(culture_name):len(culture_name)+7]
               c_last = psql.ret_culture_yield_single(culture_name)
            print(f_y)
            avg_sample=[0, 0, 0, 0, 0, 0]
            x=[2019,2020,2021,2022,2023,2024]
            plt.axvline(x = 2023, color = 'b')
           #rd
            avg = [c_last + f_y_perc[0], c_last + f_y_perc[1], c_last + f_y_perc[2],
c_last + f_y_perc[3], math.floor(c_last), f_y]
            for i, item in enumerate(avg):
               if item < 0:
                  avg[i] = 1
            avg[-1] = f_y
            worst = avg.copy()
            worst[-1] = math.floor(worst[-1]*0.9)
            # print(avg)
```

```
            # print(worst)
            best = avg.copy()
            best[-1] = math.floor(best[-1]*1.1)
            w = plt.plot(x, worst, 'ro--', label = 'Найгірший сценарій')
            b = plt.plot(x, best, 'go--', label = 'Найкращий сценарій')
            a = plt.plot(x, avg, 'bo-', label = 'Прогнозуємий сценарій')
            for i in range(len(avg)):
                plt.annotate(avg[i], (x[i], avg[i] + 0.25))
            for i in range(len(worst[:1])):
                i = i + len(worst) - 1
                plt.annotate(worst[i], (x[i], worst[i] + 0.25))
            for i in range(len(best[:1])):
                i = i + len(best) - 1
                plt.annotate(best[i], (x[i], best[i] + 0.25))
            # plt.xticks(range(len(x)), x)
            plt.legend(loc="upper left")
            plt.ylim(min(worst) - 5, max(best) + 5)

            # plt.plot(x,z)
            plt.xlabel('Рік')
            plt.ylabel('Врожайність (ц / га)')
            plt.title(f"{culture_name} - очікуваний урожай ({region} обл.)")
            # plt.show()

            plt.savefig('./static/plot.png', bbox_inches='tight')
            return {'avg': avg, 'c_prognoz_text' :
psql.ret_normalized_predict(culture=culture_name,
region=region).get('c_prognoz_text')}

        # gen_plot_bar()
```