НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

УДК 004.4:633

«ПОГОДЖЕНО»	**«ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ»**
Декан факультету інформаційних технологій	Завідувач кафедри комп'ютерних наук
Болбот І. М., д.п.н., професор	Голуб Б.Л., к.т.н., доцент
_____	_____
____ _____ 2024 р.	**____ _____ 2024 р.**

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему Програмне забезпечення системи підтримки прийняття рішень в галузі рослинництва

Спеціальність _____121 «Інжереія програмного забезпечення»_____
(код і назва)

Освітня програма _____Програмне забезпечення інформаційних систем___
(назва)

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

Гарант освітньої програми

_____професор, д.т.н. _____ _Семко В. В.____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник магістерської кваліфікаційної роботи

_____доцент к.т.н._____ _____ ___ Бородкіна І. Л.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ ___Панчук А.Є._____
(підпис) (ПІБ студента)

КИЇВ-2024

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

PHP – PHP: Hypertext Preprocessor

MySQL — My Structured Query Language

IT — Інформаційні технології

ER — Entity-Relationship

Erwin — Erwin Data Modeler

ID — Identification

PK — Primary Key

FK — Foreign Key

ООМ — Об'єктно-орієнтоване моделювання

СУБД — Система управління базами даних

HTML — HyperText Markup Language

CSS — Cascading Style Sheets

MAMP — Mac, Apache, MySQL, PHP

СППР — Система підтримки прийняття рішень

OLAP — Online Analytical Processing

ВСТУП

Актуальність теми. У сучасних умовах розвиток рослинництва є критично важливим для забезпечення продовольчої безпеки та раціонального використання природних ресурсів. Сільськогосподарські підприємства, що займаються вирощуванням рослин, відіграють ключову роль у задоволенні потреб населення в продуктах харчування, але водночас стикаються з викликами, пов'язаними з ефективним управлінням ресурсами, адаптацією до кліматичних змін і підвищенням врожайності. Однією з основних проблем у цій галузі є прийняття обґрунтованих рішень, які потребують актуальних даних про стан ґрунтів, погодні умови, агротехнології та інші фактори.

Розробка та впровадження систем підтримки прийняття рішень (СППР) у галузі рослинництва є надзвичайно актуальною темою. Такі системи дозволяють сільськогосподарським підприємствам оптимізувати управління процесами, збирати та аналізувати великі обсяги даних, а також надавати рекомендації для прийняття рішень на основі аналізу даних.

В умовах обмежених ресурсів та кліматичних викликів аграрні підприємства повинні ефективно планувати свою діяльність, швидко реагувати на зміни в умовах вирощування культур і приймати виважені рішення. СППР дозволяє отримувати актуальну інформацію про стан рослин, ґрунту, погодні прогнози, фінансові показники та інші важливі аспекти виробничого процесу.

Мета і завдання дослідження. Метою даного дослідження є розробка програмного забезпечення системи підтримки прийняття рішень (СППР) у галузі рослинництва, що дозволяє ефективно аналізувати дані, прогнозувати врожайність та оптимізувати управлінські рішення. Основна увага приділяється створенню інструменту, який інтегрує сучасні методи обробки даних, такі як Data Mining, OLAP і статистичний аналіз, для надання обґрунтованих рекомендацій агрономам та спеціалістам з управління.

Для досягнення поставленої мети в рамках дослідження визначено такі завдання:

- проаналізувати сучасний стан і методи застосування СППР у сільському господарстві;
- розробити архітектуру програмного забезпечення, включаючи базу даних, модулі аналітики та інтерфейс користувача;
- впровадити алгоритми 1-Rule, K-Means, Байєсовий метод для аналізу й прогнозування динаміки врожайності;
- протестувати систему на реальних або змодельованих даних, оцінюючи її ефективність у прийнятті рішень щодо догляду за культурами, поливу та підживлення;
- оцінити економічний ефект від впровадження системи у виробничий процес.

Об’єкт дослідження. Процеси прийняття рішень в управлінні агротехнічними заходами у галузі рослинництва.

Предмет дослідження. Методи й алгоритми аналізу даних та їх програмна реалізація в системі підтримки прийняття рішень для оптимізації процесів вирощування сільськогосподарських культур.

Методи дослідження. У дослідженні використано методи аналізу даних, включаючи алгоритми кластеризації (K-Means), прогнозування (Байєсовий метод і модель ARIMA) та виявлення закономірностей (метод 1-Rule і Frequent Pattern Growth). Застосовано методи системного аналізу для проектування програмного забезпечення, а також методи тестування для оцінки ефективності розробленої системи.

Наукова новизна та практичне значення роботи. Наукова новизна полягає у застосуванні комбінованого підходу до аналізу агротехнічних даних із використанням сучасних алгоритмів Data Mining і OLAP для розробки інтегрованої СППР. Практичне значення роботи полягає у створенні програмного інструменту, який дозволяє агрономам приймати обґрунтовані рішення щодо управління сільськогосподарськими культурами, що підвищує ефективність агровиробництва та знижує витрати на ресурси.

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Галузь рослинництва є однією з ключових складових сільськогосподарського виробництва, яка забезпечує продовольчу безпеку, формує експортний потенціал і впливає на економічний розвиток країни. В умовах глобальних кліматичних змін та зростання населення постає необхідність раціонального управління процесами вирощування сільськогосподарських культур. Основним завданням у цій сфері є забезпечення стабільного та високого рівня врожайності при оптимальному використанні ресурсів.

Предметна область охоплює широкий спектр завдань, таких як аналіз кліматичних умов, оцінка стану ґрунту, визначення потреб у добривах, вибір методів захисту рослин, планування поливу та прогнозування врожайності. Сучасне рослинництво стикається з такими викликами, як нерівномірний розподіл ресурсів, обмежений доступ до актуальної інформації, необхідність врахування численних чинників і параметрів, що впливають на розвиток культур.

У традиційних підходах управління часто виникають труднощі з обробкою великих обсягів даних. Це стосується інформації про фазу росту рослин, зміну погодних умов, рівень вологості та температуру ґрунту, особливості внесення добрив та використання пестицидів. Відсутність автоматизованих інструментів призводить до суб’єктивності у прийнятті рішень, що знижує ефективність агротехнічних заходів.

Рішення цих проблем можливе завдяки застосуванню сучасних інформаційних технологій, таких як системи підтримки прийняття рішень (СППР). СППР дозволяють інтегрувати, аналізувати та обробляти дані для формування обґрунтованих рекомендацій щодо управління агротехнічними процесами. У фокусі таких систем знаходяться технології аналізу даних (Data Mining), багатовимірного аналізу (OLAP) і прогнозування, які дозволяють

виявляти закономірності, оцінювати ризики та планувати ефективну діяльність у рослинництві.

Таким чином, розробка СППР для рослинництва має велике значення для підвищення ефективності управління аграрними процесами, що сприяє раціональному використанню природних ресурсів, мінімізації витрат та забезпеченню стабільного рівня врожайності.

1.2 Аналіз вимог до програмної системи

Аналіз вимог до програмної системи підтримки прийняття рішень у галузі рослинництва є одним із ключових етапів проектування, що визначає її функціональність, архітектуру та зручність використання. Цей аналіз спрямований на ідентифікацію потреб кінцевих користувачів — агрономів, фермерів, експертів у сфері рослинництва — та забезпечення відповідності сучасним технологічним стандартам.

Основні функціональні вимоги до системи включають:

- Збір даних. Система повинна забезпечувати автоматизований збір інформації з різних джерел: метеорологічних станцій, сенсорів у полях, аналізаторів ґрунту та баз агротехнічних даних.
- Обробка даних. Передбачено попереднє очищення та структурування отриманої інформації з метою формування баз даних, що придатні для подальшого аналізу.
- Аналіз даних. Система повинна підтримувати багатовимірний аналіз, використовуючи методи Data Mining і OLAP для виявлення закономірностей, кластеризації полів і прогнозування врожайності.
- Формування рекомендацій**.** На основі аналізу даних система повинна автоматично створювати рекомендації щодо підживлення, поливу, використання пестицидів та інших агротехнічних заходів.
- Прогнозування. Алгоритми прогнозування повинні враховувати кліматичні умови, тип ґрунту та агротехнічні заходи для передбачення врожайності.

До нефункціональних вимог належать:

- Продуктивність. Система повинна швидко обробляти великі обсяги даних, забезпечуючи своєчасну генерацію звітів.
- Масштабованість. Передбачено адаптацію системи до різних масштабів господарств — від невеликих ферм до великих агропромислових комплексів.
- Інтеграція. Система має підтримувати зв’язок з різними джерелами даних, сенсорами та іншими інформаційними системами.
- Зручність використання. Інтерфейс системи повинен бути інтуїтивно зрозумілим і доступним навіть для користувачів із базовим рівнем технічної підготовки.

Система реалізована як вебдодаток із підтримкою мобільної версії, що працює в хмарному середовищі або на локальних серверах. Для серверної частини вебдодатку використовується PHP, що забезпечує динамічну обробку запитів, створення зручного веб-інтерфейсу та взаємодію з базами даних. Дані зберігаються у сучасній реляційній базі даних MySQL, яка ефективно обробляє великі обсяги інформації. Для аналізу даних у системі були інтегровані бібліотеки Python.

Користувачі системи очікують:

- Надійності роботи в режимі 24/7.
- Можливості налаштування параметрів під специфічні потреби господарства.
- Підтримки української мови для зручності використання.

Аналіз вимог дозволив сформувати чітке уявлення про основні характеристики системи, що забезпечує її ефективність та відповідність потребам аграрного сектору. Це сприяє оптимізації агротехнічних процесів, зниженню витрат і підвищенню врожайності.

1.3 Аналіз систем, що використовуються в аграрному секторі

Сучасний аграрний сектор активно використовує програмні системи для оптимізації виробничих процесів, підвищення врожайності та зниження витрат. Інноваційні рішення впроваджуються в управління земельними ресурсами, моніторинг стану культур, прогнозування врожайності та планування агротехнічних заходів. У цьому розділі розглянемо основні системи, що використовуються в аграрній сфері, їхні функції, переваги та недоліки.

До інформаційно-аналітичних систем належать програмні продукти, що забезпечують збір, зберігання, аналіз та візуалізацію даних про стан сільськогосподарських угідь.

Програмні комплекси, такі як AgroOffice або FieldView, дозволяють проводити облік витрат на добрива, насіння та пестициди, контролювати стан полів і моніторити врожайність [1].

- Переваги: Інтуїтивно зрозумілий інтерфейс, доступ до даних у реальному часі.
- Недоліки: Обмежені можливості для аналізу великих масивів даних, висока вартість ліцензій.

Геоінформаційні системи (ГІС). Інструменти, такі як QGIS або ArcGIS, використовуються для аналізу ґрунтових і кліматичних умов, побудови карт родючості, планування зрошення [2].

- Переваги: Потужні можливості для візуалізації просторових даних.
- Недоліки: Складність у використанні для непідготовлених користувачів.

Системи моніторингу та автоматизації

Моніторингові системи дозволяють контролювати стан культур у реальному часі за допомогою датчиків, дронів і супутникових технологій.

- Приклад: SmartAgriHub**.** Використовує IoT-пристрої для збору даних про температуру, вологість ґрунту, освітленість тощо [3].

- Переваги: Точність даних, інтеграція з іншими системами.
- Недоліки: Високі витрати на впровадження та обслуговування.

Системи прогнозування врожайності

Системи, засновані на методах машинного навчання, такі як Cropwise або AgroPredict, використовуються для прогнозування врожайності залежно від кліматичних умов, типу ґрунту та інших факторів [4].

- Переваги: Підвищення точності планування агротехнічних заходів.
- Недоліки: Залежність від обсягу та якості вхідних даних.

Хоча більшість сучасних програмних систем мають широкий спектр функцій, багато з них зосереджені лише на окремих аспектах аграрного виробництва, таких як моніторинг або аналіз даних. Водночас недостатньо рішень, що забезпечують комплексну підтримку прийняття рішень на основі аналізу великих даних, інтеграції сучасних алгоритмів Data Mining та OLAP. Це підкреслює актуальність розробки універсальної системи підтримки прийняття рішень, яка б інтегрувала функції збору, аналізу та прогнозування для оптимізації процесів рослинництва.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Опис та аналіз методології системного аналізу

Системний аналіз — це комплексний підхід до дослідження складних систем, який застосовується для вивчення їх структури, функціонування, взаємодії елементів і визначення найкращих шляхів їх оптимізації та розвитку. Цей підхід широко використовується в багатьох галузях, таких як інформаційні технології, бізнес-аналіз, управління проектами, екологія, економіка та інші.

Методологія системного аналізу включає кілька ключових етапів:

1. Ідентифікація проблеми: На першому етапі визначається основна проблема, яка потребує вирішення. Важливо чітко сформулювати проблему, адже це є основою для подальшого аналізу.
2. Моделювання системи: Це процес створення абстракції або моделі системи, яку потрібно вивчити. Моделі можуть бути різними: від простих діаграм до складних математичних моделей. Вони допомагають візуалізувати взаємозв'язки між елементами системи.
3. Збір та аналіз даних: На цьому етапі збираються всі необхідні дані про систему: історична інформація, поточні показники та тренди. Аналіз даних дозволяє зрозуміти, як система працює на даний момент і які є слабкі місця.
4. Аналіз альтернатив: Після моделювання та збору даних вивчаються можливі альтернативи для вирішення проблеми. Це можуть бути різні стратегії, технології або методи, які можна застосувати для покращення роботи системи.
5. Оцінка результатів: На основі аналізу альтернатив оцінюються різні варіанти рішень. Проводиться порівняння за критеріями ефективності, витрат, часу, ризиків тощо.
6. Рекомендації та втілення: Після оцінки різних альтернатив вибирається найкращий варіант і надаються рекомендації щодо його втілення. Це включає розробку детального плану дій і моніторинг виконання.

7. Оцінка результатів впровадження: Після реалізації рішення необхідно оцінити, наскільки ефективно воно працює в реальному середовищі. Якщо система не працює так, як очікувалося, можуть бути внесені корективи.

Аналіз методології системного аналізу

Переваги:

- Комплексність: Системний аналіз дає змогу розглядати проблему в її повному контексті, враховуючи всі елементи системи та їх взаємозв'язки. Це дозволяє уникнути дрібних упущень і забезпечує більш точне розуміння проблеми.
- Визначення ключових факторів: Системний аналіз дозволяє визначити основні чинники, які найбільше впливають на ефективність системи, що допомагає приймати обґрунтовані рішення.
- Управління складністю: В умовах швидко змінюваного середовища системний аналіз дозволяє виявити складні залежності та динамічні процеси, що є важливими для прийняття стратегічних рішень.
- Оптимізація рішень: Системний аналіз надає змогу оцінити різні варіанти і вибрати оптимальні рішення на основі всебічного аналізу.

Недоліки:

- Часомісткість: Системний аналіз може займати значну кількість часу через необхідність збору даних, моделювання, аналізу альтернатив і оцінки результатів.
- Складність моделювання: Для деяких складних систем може бути важко побудувати точну модель, що може призвести до помилок у висновках.
- Залежність від якості даних: Якість та точність результатів системного аналізу безпосередньо залежить від якості даних, що

використовуються для аналізу. Якщо дані неповні або неточні, результати можуть бути неточними.

- Складність інтеграції різних підходів: У складних і багатогранних системах може бути важко узгодити різні підходи до аналізу та прийняття рішень, що може ускладнити процес.

Застосування методології системного аналізу

1. IT-системи та програмне забезпечення: Системний аналіз використовується для проектування складних інформаційних систем, визначення їх вимог, оптимізації процесів і забезпечення ефективної інтеграції різних підсистем.

2. Управління проектами: В аналізі проектів системний підхід дозволяє оцінити ризики, планувати ресурси і час, а також забезпечувати виконання проекту в межах бюджету.

3. Бізнес і економіка: У бізнесі системний аналіз використовується для вивчення операційних процесів, прийняття рішень щодо розширення чи оптимізації, прогнозування трендів.

4. Екологія та охорона навколишнього середовища: Для управління природними ресурсами, дослідження екосистем і прийняття рішень щодо захисту навколишнього середовища застосовуються методи системного аналізу.

5. Управління виробництвом: Визначення оптимальних виробничих процесів, оцінка ефективності технічного обладнання, планування виробничих ліній — все це здійснюється за допомогою системного аналізу.

Методологія системного аналізу є важливим інструментом для вирішення складних проблем, що виникають у різних сферах. Вона дає змогу комплексно підходити до вивчення проблемних ситуацій і знаходити оптимальні рішення. Водночас, вона вимагає значних ресурсів і часу для виконання, що може бути обмеженням у деяких ситуаціях. Тим не менш, її використання виявляється ефективним у багатьох випадках, де важливі деталі та точність у прийнятті рішень.

2.2 Логічна модель даних у вигляді ER-діаграми

Erwin - це програмний інструмент для моделювання даних, зокрема для створення ER-діаграм. Він надає можливість розробникам і аналітикам створювати, редагувати та аналізувати моделі даних у зручному та інтуїтивно зрозумілому середовищі [5].

За допомогою Erwin можна легко створювати різні типи сутностей та встановлювати між ними відношення. Програма дозволяє визначати атрибути кожної сутності та їх типи даних, такі як числа, рядки, дата тощо. Крім того, Erwin надає можливість додавати обмеження цілісності, такі як первинні ключі, зовнішні ключі та унікальні обмеження, для забезпечення цілісності даних у моделі.

Однією з основних переваг Erwin є його зручний інтерфейс, який дозволяє швидко розташувати сутності на діаграмі, з'єднати їх зв'язками та легко редагувати їх властивості. Програма також підтримує можливість автоматичної генерації скриптів створення бази даних з моделі, що зберігає час та зусилля при розробці. За допомогою Erwin можна легко додавати таблиці, визначати їх атрибути та встановлювати зв'язки між ними.

Іншою перевагою Erwin є його функціональність для реверс-інжинірингу, що дозволяє імпортувати існуючі бази даних з різних джерел і створювати моделі даних на їх основі. Це дозволяє розробникам легко аналізувати та модифікувати існуючі бази даних або розробляти нові проекти на основі наявних даних.

Логічна модель даних у вигляді ER-діаграми є інструментом, який допомагає візуалізувати структуру даних у системі та їх взаємозв'язки. Вона використовується для представлення основних сутностей в системі та описує, як ці сутності пов'язані між собою.

ER-діаграма має кілька ключових елементів. Спочатку визначаються сутності, які представляють основні об'єкти або концепції в системі. Кожна сутність має свої властивості або атрибути, які описують характеристики цієї сутності. Після цього встановлюються зв'язки між сутностями, які показують, як сутності взаємодіють між собою.

ER-діаграма є стандартним інструментом для проектування баз даних, що використовується в багатьох сферах індустрії. Це означає, що ER-діаграма має широку підтримку, доступні ресурси та документацію. Вона допомагає зробити проект більш зрозумілим та лаконічним, що зробить його більш доступним для інших фахівців і сприятиме ефективній співпраці в команді.

Логічна модель даних у вигляді ER-діаграми дозволяє краще розуміти структуру даних та їх зв'язки в системі. Вона є потужним інструментом для аналізу та проектування системи, оскільки вона допомагає визначити потреби в даних, встановити правильні залежності та забезпечити зрозумілість і співпрацю між розробниками та зацікавленими сторонами.

Крім того, ER-діаграма дозволяє провести аналіз вимог і інформаційної моделі, виявити потенційні проблеми та помилки ще на етапі проектування. Це може зекономити час і зусилля, які потрібно було б витратити на виправлення помилок під час реалізації системи.

Використання логічної моделі даних у вигляді ER-діаграми в дипломній роботі дозволяє чітко визначити структуру даних, що потрібні для розробки системи управління пасікою, і зробити цю інформацію доступною та зрозумілою для читачів роботи.

Логічна модель даних, яка зображена на рис.1, складається з таких сутностей:

Сутності та зв’язки:

1. Crops:
    - Атрибути: Crop_ID (PK), Crop_Name, Crop_Type.
2. Climate_Conditions:
    - Атрибути: Condition_ID (PK), Temperature, Humidity, Precipitation, Crop_ID (FK).
    - Зв'язок: один запис у Crops пов’язаний із багатьма записами у Climate_Conditions.
3. Soil_Indicators:

- Атрибути: Soil_Indicator_ID (PK), Moisture_Level, Nutrient_Content, Crop_ID (FK).
- Зв'язок: один запис у Crops пов’язаний із багатьма записами у Soil_Indicators.

4. Agronomic_Practices:
    - Атрибути: Practice_ID (PK), Practice_Name, Frequency, Crop_ID (FK).
    - Зв'язок: один запис у Crops пов’язаний із багатьма записами у Agronomic_Practices.
5. Yield_Prediction:
    - Атрибути: Prediction_ID (PK), Year, Predicted_Yield, Crop_ID (FK).
    - Зв'язок: один запис у Crops пов’язаний із багатьма записами у Yield_Prediction.
6. Monitoring:
    - Атрибути: Monitoring_ID (PK), Date, Field_Temperature, Field_Humidity, Crop_ID (FK).
    - Зв'язок: один запис у Crops пов’язаний із багатьма записами у Monitoring.

2.3 Діаграма класів

Діаграма класів є одним із основних елементів моделювання об'єктно-орієнтованих систем в UML (Unified Modeling Language). Вона надає абстрактне уявлення про структуру системи, показуючи класи, їх атрибути, методи та взаємозв'язки між ними [6].

Основні компоненти діаграми класів:

1. Клас (Class): Клас є основною одиницею в діаграмі. Він визначає структуру об'єктів і містить:
    - Атрибути: Характеристики, що описують об'єкти класу.
    - Методи: Функції або операції, що можуть бути виконані об'єктами цього класу.
2. Зв'язки між класами:

- Асоціація (Association): Відображає зв'язок між двома класами. Зазвичай, це лінія між класами з зазначенням кардинальності (наприклад, 1 до багатьох).
- Агрегація (Aggregation): Спеціальний тип асоціації, що вказує на відносини "частина-ціле".
- Композиція (Composition): Сильніша форма агрегації, де об'єкт-частина не може існувати без об'єкта-цілого.
- Спадкування (Inheritance): Вказує, що один клас наслідує інший.
- Залежність (Dependency): Один клас залежить від іншого.

3. Видимість:
    - + (public): Атрибут або метод доступний ззовні.
    - - (private): Атрибут або метод доступний тільки в межах класу.
    - # (protected): Атрибут або метод доступний в межах класу та його підкласів.

Приклад діаграми класів

Розглянемо приклад діаграми класів для сільськогосподарської системи, що включає класи для Crops (Культури), ClimateConditions (Кліматичні умови), SoilIndicators (Ґрунтові індикатори), AgronomicPractices (Агрономічні практики) та Monitoring (Моніторинг).

Класи та їх атрибути:

1. Crops (Культури):
    - Атрибути: Crop_ID, Crop_Name, Crop_Type
    - Методи: addCrop(), removeCrop()
2. ClimateConditions (Кліматичні умови):
    - Атрибути: Condition_ID, Temperature, Humidity, Precipitation
    - Методи: setConditions(), getConditionData()
3. SoilIndicators (Ґрунтові індикатори):
    - Атрибути: Soil_Indicator_ID, Moisture_Level, Nutrient_Content
    - Методи: setSoilConditions(), getSoilData()

4. AgronomicPractices (Агрономічні практики):
    - Атрибути: Practice_ID, Practice_Name, Frequency
    - Методи: applyPractice(), schedulePractice()
5. Monitoring (Моніторинг):
    - Атрибути: Monitoring_ID, Date, Field_Temperature, Field_Humidity
    - Методи: recordMonitoring(), getMonitoringData()

Зв'язки:

1. Crops має асоціації з іншими класами:
    - 1 : N з ClimateConditions (Одна культура може мати кілька кліматичних умов).
    - 1 : N з SoilIndicators (Одна культура може мати кілька ґрунтових індикаторів).
    - 1 : N з AgronomicPractices (Одна культура може мати кілька агрономічних практик).
    - 1 : N з Monitoring (Одна культура може бути моніторена кілька разів).

Опис діаграми класів:

1. Crops є основним класом, і до нього приєднуються інші класи через асоціації.
2. Клас ClimateConditions зберігає інформацію про кліматичні умови, пов'язані з конкретною культурою.
3. SoilIndicators зберігає інформацію про індикатори ґрунту, що впливають на культуру.
4. AgronomicPractices визначає агрономічні практики, застосовувані до конкретної культури.
5. Monitoring реєструє дані моніторингу, що включають температуру і вологість на полі.

2.4 Діаграма прецедентів

Діаграма прецедентів (Use Case Diagram) є ключовим елементом в аналізі вимог системи, зокрема в процесах системного аналізу та об'єктно-орієнтованого

моделювання (ООМ). Вона дозволяє чітко визначити, як система буде взаємодіяти з зовнішнім середовищем і її користувачами. Така діаграма відображає не тільки типи користувачів системи (актори), але й основні функціональні можливості, які система повинна реалізувати для задоволення потреб користувачів [7]. У рамках магістерської роботи, діаграма прецедентів використана для демонстрації ключових процесів у розробці та реалізації системи підтримки прийняття рішень у галузі рослинництва.

Основні компоненти діаграми прецедентів:

- Актори (Actors): Актори — це зовнішні користувачі або системи, які взаємодіють з основною системою. Вони можуть бути як фізичними особами, так і іншими автоматизованими системами чи пристроями.
- Прецеденти (Use Cases): Прецеденти відображають основні функції, які система надає користувачам для досягнення певних цілей. Це конкретні дії, які актори виконують у системі. Наприклад, фермер може "Внести дані про клімат", а система на основі цих даних генерує "Прогноз врожайності". Прецеденти повинні бути визначені на основі аналізу вимог і виявлення основних бізнес-процесів, які система повинна підтримувати.
- Зв'язки (Associations): Зв'язки між акторами та прецедентами визначають, як користувачі взаємодіють із системою. Зв'язки можна розглядати у вигляді простих ліній, що з’єднують акторів із прецедентами, вказуючи на те, що актор може виконувати певну функцію.

Типи зв'язків між прецедентами:

Для більш детального опису взаємодії між різними функціями системи використовуються типи зв'язків, такі як включення (include) та розширення (extend):

- Включення (Include): Це зв'язок, який показує, що один прецедент завжди включає в себе виконання іншого прецеденту. Наприклад, прецедент "Прогнозувати врожайність" може обов’язково включати в себе прецедент "Аналізувати дані про ґрунт", оскільки для точного прогнозу необхідно враховувати показники ґрунту.

- Розширення (Extend): Цей тип зв’язку показує, що один прецедент може бути доповнений іншими, але це не є обов’язковим. Наприклад, якщо виявлені екстремальні погодні умови, система може розширити прогноз врожайності додатковими рекомендаціями для агрономічних практик.

Переваги використання діаграм прецедентів у розробці системи:

- Визначення вимог: Діаграма прецедентів дозволяє чітко визначити вимоги до системи, що полегшує наступні етапи розробки та реалізації.
- Комунікація між стейкхолдерами: Використання діаграм прецедентів полегшує комунікацію між розробниками та кінцевими користувачами, оскільки вона дозволяє зрозуміти ключові функції системи без глибоких технічних знань.
- Спрощення процесу тестування: Завдяки чітким визначенням прецедентів тестувальники можуть легше створювати сценарії тестування, що покривають всі важливі функції системи.

Діаграма прецедентів є важливим інструментом в процесах проектування та аналізу системи, оскільки вона допомагає чітко визначити взаємодію користувачів з системою та її функціональні можливості. Вона є основою для створення подальших моделей та прототипів, які використовуються на етапах проектування та розробки програмного забезпечення. Для системи підтримки прийняття рішень у галузі рослинництва це дозволяє зрозуміти, як фермери, агрономи та інші користувачі будуть взаємодіяти з системою для прийняття обґрунтованих рішень щодо агрономічних практик та прогнозування врожайності.

Діаграма прецедентів сільському господарстві

Актори:

1. Агроном - основний користувач системи, відповідальний за управління врожаєм і прийняття рішень.
2. Data Scientist – керує аналізом даних і моделями машинного навчання.
3. Системний адміністратор - контролює обслуговування системи та керування користувачами.

Випадки використання:

1. Збір даних – збирайте дані про погоду, ґрунт, умови посівів та інші сільськогосподарські параметри.
2. Обробка даних – очищення, попередня обробка та структурування зібраних даних для аналізу.
3. Аналіз даних і прогнозування . Використовуйте інтелектуальний аналіз даних, K-Means, методи Байєса та моделі ARIMA для аналізу даних і прогнозування врожайності.
4. Рекомендації щодо прийняття рішень – генеруйте практичні рекомендації (наприклад, графіки поливу, плани внесення добрив) на основі аналізу даних.
5. Моніторинг і сповіщення - відстежуйте показники росту культур і сповіщайте користувачів, якщо ключові параметри перевищують порогові значення.
6. Звітування – надайте докладні звіти про тенденції врожайності, моделі росту та використання ресурсів.
7. Конфігурація системи – дозволяє адміністратору налаштовувати параметри системи, рівні доступу та інтегрувати додаткові джерела даних.

Структура діаграми:

На діаграмі кожен варіант використання має бути пов’язаний із відповідним актором(ами) за допомогою таких зв’язків:

- Агроном : взаємодіє з «Збором даних», «Рекомендаціями щодо прийняття рішень», «Моніторингом і сповіщеннями» та «Звітністю».
- Data Scientist : керує «обробкою даних» і «аналізом і прогнозуванням даних».
- Системний адміністратор : керує «Конфігурацією системи» і може допомогти в налаштуванні «Обробки даних».

2.4 Діаграма послідовності

Діаграма послідовності (Sequence Diagram) є важливим елементом об'єктно-орієнтованого моделювання, який візуалізує, як об'єкти системи

взаємодіють між собою у процесі виконання конкретної операції чи сценарію. Вона допомагає зрозуміти часову послідовність взаємодії між об'єктами і може бути використана для опису різних процесів і сценаріїв у магістерській роботі, наприклад, у контексті системи підтримки прийняття рішень у галузі рослинництва [8].

1. Основні компоненти діаграми послідовності:

- Об'єкти (Objects): У діаграмі послідовності об'єкти (актори, інші системи, компоненти) представлені у вигляді вертикальних ліній (лінії життєвого циклу), які відображають їх існування протягом часу. Це можуть бути як фізичні особи (фермери, агрономи), так і системи чи компоненти, що взаємодіють між собою, такі як система прогнозування погоди або база даних.
- Повідомлення (Messages): Повідомлення в діаграмі послідовності представляють взаємодії між об'єктами і відображають порядок, у якому передаються запити та відповіді. Це можуть бути запити на введення даних, виконання обчислень, виклик функцій, чи отримання результатів.
- Взаємодії (Interactions): Взаємодії описують, як об'єкти обмінюються даними та керують потоком виконання. Вони можуть бути синхронними або асинхронними. Наприклад, виклик методу для прогнозування врожайності може бути синхронним (коли фермер або агроном отримує результат негайно) або асинхронним (якщо система прогнозування врожайності працює в фоновому режимі і надає результат через деякий час).
- Часова лінія (Lifeline): Часова лінія (вертикальна лінія) показує життєвий цикл об'єкта і коли він існує під час виконання сценарію.
- Активація (Activation): Це горизонтальний прямокутник на часовій лінії об'єкта, який показує період часу, коли об'єкт активний і виконує певну операцію, наприклад, обробку запиту або обчислення.

2. Структура та побудова діаграми послідовності для системи підтримки прийняття рішень у галузі рослинництва:

Приклад сценарію: Прогнозування врожайності на основі даних про клімат та ґрунт

Уявімо, що фермер хоче отримати прогноз врожайності для певної культури на основі поточних кліматичних умов і показників ґрунту. Діаграма послідовності для цього сценарію включатиме такі етапи:

2.1 Фермер ініціює запит: Фермер вводить дані про стан клімату та ґрунту в систему. Запит на введення даних ініціюється через інтерфейс користувача, який з'єднаний з системою.

2.2 Система зберігає дані: Система перевіряє коректність введених даних і зберігає їх у базі даних.

2.3 Запит до системи прогнозування погоди: Система прогнозування погоди отримує запит від основної системи, щоб надати актуальну інформацію про погодні умови.

2.4 Система прогнозування погоди надає дані: Прогноз погоди надається на основі останніх вимірювань та прогнозів, і ці дані передаються до основної системи.

2.5 Розрахунок прогнозу врожайності: На основі отриманих даних система виконує розрахунок прогнозованої врожайності за допомогою алгоритмів прогнозування. Це може включати аналіз кліматичних даних, індикаторів ґрунту і агрономічних практик.

2.6 Результати прогнозу: Система надає прогноз врожайності фермеру, показуючи прогнозовані результати та рекомендації для подальших дій.

3. Типи повідомлень на діаграмі послідовності:

- Синхронне повідомлення: Це повідомлення, яке передає запит від одного об'єкта до іншого, і процес продовжується тільки після того, як відповідь отримана. Наприклад, запит до системи прогнозування погоди для отримання даних.

- Асинхронне повідомлення: Це повідомлення, коли об'єкт надсилає запит іншому об'єкту, але не чекає на відповідь для продовження свого

виконання. Наприклад, системи можуть одночасно здійснювати інші операції після того, як почали обробку запиту на прогнозування врожайності.

4. Переваги використання діаграм послідовності:

- Покращення розуміння процесів: Діаграма допомагає деталізувати взаємодії між різними компонентами системи, що дозволяє краще зрозуміти її роботу.
- Підвищення якості тестування: Оскільки діаграма чітко показує порядок виконання операцій, її можна використовувати для створення тестових сценаріїв.
- Оптимізація розробки: Зрозуміло, як компоненти системи повинні взаємодіяти, що допомагає уникати непотрібних складнощів і спрощує процес розробки.

Діаграма послідовності є ефективним інструментом для візуалізації взаємодії між об'єктами системи на різних етапах її роботи. Вона дає чітке уявлення про часову послідовність дій і допомагає забезпечити правильну організацію процесів. Для системи підтримки прийняття рішень у галузі рослинництва це дозволяє точно визначити, як фермери, агрономи та інші користувачі будуть взаємодіяти з програмним забезпеченням для досягнення своїх цілей, таких як прогнозування врожайності.

2.5 Діаграма розгортання і компонентів

Діаграма розгортання (Deployment Diagram) використовується для представлення фізичної структури програмної системи. Вона описує, як програмні компоненти зв'язуються з апаратними засобами (сервери, комп'ютери, бази даних і т.д.) [9]. Це дозволяє з’ясувати, як система працюватиме в реальному середовищі і на яких пристроях вона буде розгорнута.

Основні вузли:

Сервер обробки даних: сервер, на якому обробляються великі обсяги даних (кліматичні умови, показники ґрунту) і виконуються прогнози врожайності.

База даних: сервер, який зберігає всю інформацію про культури, ґрунтові індикатори, кліматичні умови, прогнозовані врожаї тощо.

Клієнтський пристрій (мобільний додаток або веб-інтерфейс): використовується для відображення результатів і взаємодії користувачів (фермерів, агрономів) з системою.

Зв'язки:

Сервер обробки даних → База даних: сервери обмінюються даними для збереження результатів обробки та прогнозів.

Клієнтський пристрій → Сервер обробки даних: клієнтські пристрої звертаються до сервера для отримання прогнозів і введення нових даних.

Клієнтський пристрій → База даних: безпосередня взаємодія для отримання даних, якщо це потрібно.

Діаграма компонентів описує логічну структуру програмної системи, що показує, як програмні компоненти взаємодіють між собою для досягнення цілей системи. Вона відображає основні компоненти програми, такі як модулі, класи або пакети, та їхні взаємозв’язки через інтерфейси.

Основні компоненти:

Компонент введення даних: відповідає за введення даних про кліматичні умови, ґрунтові індикатори, агрономічні практики тощо. Це може бути мобільний додаток або веб-форму для введення даних користувачем.

Компонент обробки даних: обробляє всі зібрані дані, використовуючи алгоритми прогнозування врожайності та аналізу кліматичних і ґрунтових умов.

Компонент прогнозування врожайності: цей компонент відповідає за побудову моделей прогнозу на основі введених даних (наприклад, штучний інтелект або статистичні моделі).

Компонент виведення результатів: надає користувачеві результати прогнозування врожайності, графіки, звіти.

Компонент звітності: генерує і форматує звіти для користувачів (фермерів, агрономів), що містять прогнозовані дані, рекомендації по агрономічним практикам.

Зв'язки між компонентами:

Компонент введення даних ↔ Компонент обробки даних: компонент введення даних передає введену інформацію компоненту обробки даних.

Компонент обробки даних ↔ Компонент прогнозування врожайності: для розрахунку прогнозів врожайності використовуються оброблені дані.

Компонент обробки даних ↔ Компонент виведення результатів: оброблені дані передаються до компонента виведення результатів.

Компонент виведення результатів ↔ Користувач: користувач взаємодіє з системою через інтерфейс, отримуючи результати і вносячи коригування.

Діаграма розгортання та компонента є потужними інструментами для проектування складних інформаційних систем. Вони допомагають зрозуміти, як програмні компоненти взаємодіють і де вони будуть розгорнуті в реальному світі. Для системи підтримки прийняття рішень у галузі рослинництва, ці діаграми дозволяють оптимізувати її архітектуру, забезпечити належне розподілення ресурсів і зрозуміти, як компоненти повинні взаємодіяти для досягнення потрібних результатів.

3 РОЗРОБКА ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Система управління інформаційною базою

У програмному забезпеченні база даних відіграє ключову роль у зберіганні, організації та керуванні великим обсягом структурованої інформації. Основними причинами використання бази даних у дипломній роботі є:

Збереження даних: База даних дозволяє зберігати інформацію про об'єкти, процеси або сутності, що використовуються у дипломній роботі. Це можуть бути дані про користувачів, вулики, локації, статистика і т. д. Завдяки базі даних дані зберігаються структуровано і доступні для подальшої обробки.

Ефективне управління даними: База даних забезпечує можливість швидкого пошуку, сортування, фільтрації та оновлення даних. Це дозволяє ефективно керувати інформацією, використовуючи різні операції і запити до бази даних.

Забезпечення цілісності даних: База даних дозволяє забезпечити цілісність даних шляхом використання правил і обмежень. Це допомагає зберігати інформацію у вірному і непошкодженому стані.

Підтримка багатокористувацького доступу: База даних дозволяє багатьом користувачам одночасно працювати зі спільними даними, забезпечуючи контроль доступу і конкурентне виконання операцій. Наприклад, різні користувачі можуть мати доступ лише до своїх власних вуликів або обмежений доступ до редагування певних полів.

Забезпечення резервного копіювання і відновлення даних: База даних дозволяє створювати резервні копії даних і відновлювати їх у разі втрати або пошкодження. Це забезпечує захист і безпеку інформації, особливо в разі важливих даних, які не можуть бути відновлені.

Порівняно з іншими методами зберігання даних, такими як файли чи електронні таблиці, база даних має декілька переваг. Вона забезпечує структуроване зберігання даних, покращену швидкодію при обробці даних,

підтримку багатокористувацького доступу та зручні інструменти для роботи з даними, такі як мови запитів і засоби аналітики.
В ході написання дипломної роботи, для системи управління базою даних (СУБД) було обрано MySQL. Цей вибір був зроблений на підставі кількох чинників і обґрунтований детальним аналізом.

MySQL є потужною та надійною системою управління базами даних, яка широко використовується у програмній розробці, веб-додатках та інших проектах. Вона пропонує широкий набір функцій і можливостей для зберігання, керування та маніпулювання даними [10].

Однією з головних переваг MySQL є його швидкодія. Вона володіє високою продуктивністю, що дозволяє швидко виконувати запити до бази даних, особливо при оптимальному налаштуванні та індексуванні таблиць. Крім того, MySQL підтримує різні методи оптимізації запитів, такі як кешування, оптимізатор запитів та індексування, що забезпечує ефективну роботу з великим обсягом даних.

MySQL також володіє великою масштабованістю, що дозволяє легко розширювати базу даних зростанням обсягу даних та навантаження. Він підтримує розподілені системи баз даних, реплікацію та кластеризацію, що дозволяє створювати високодоступні та надійні рішення.

MySQL підтримує різноманітні типи даних, такі як числа, рядки, дата та час, бінарні дані і багато інших. Вона також має вбудовану підтримку для роботи з реляційними даними, включаючи можливість створення та керування таблицями, виконання операцій з'єднання, фільтрації та сортування даних.

Безпека є ще однією важливою характеристикою MySQL. Вона підтримує механізми автентифікації, рівні доступу та шифрування, що дозволяє забезпечити захист від несанкціонованого доступу та зламів.

Ось деякі порівняння з іншими популярними СУБД:

- postgresql має більш розширену підтримку стандартів SQL і надає більше можливостей для складних запитів та операцій з базами даних;

- oracle є комерційною СУБД з великою функціональністю та високою продуктивністю, зазвичай використовується в великих підприємствах з великими обсягами даних;
- microsoft sql server має більшу підтримку для бізнес-функціональності та інтеграції з іншими продуктами Microsoft.

У порівнянні з іншими системами управління базами даних, MySQL має велику спільноту користувачів і розробників, що забезпечує наявність великої кількості документації, ресурсів та підтримки. Вона також є безкоштовною та відкритою системою з відкритим вихідним кодом, що робить її доступною для використання та модифікації.

3.2 Розробка інформаційної бази

Розробка інформаційної бази є важливим аспектом дипломного проекту. Інформаційна база служить центральним сховищем для зберігання, управління та обробки даних, необхідних для ефективного функціонування програмного забезпечення.

Основною метою розробки інформаційної бази є забезпечення централізованого та структурованого зберігання даних. Це дозволяє ефективно організовувати великий обсяг інформації, що використовується програмним забезпеченням, і керувати ним. Інформаційна база дозволяє зберігати дані у відповідних таблицях і встановлювати зв'язки між ними, забезпечуючи цілісність даних і мінімізуючи надмірність даних.

Крім того, розробка інформаційної бази дає можливість виконувати різноманітні операції з даними, такі як додавання, редагування та видалення.

Також, інформаційна база дозволяє здійснювати пошук, фільтрацію та сортування даних для зручного доступу до необхідної інформації. Вона забезпечує можливість зберігання історії змін даних і ведення журналу подій, що дозволяє відстежувати та аналізувати зміни у системі.

Таблиця «Culture», має такі поля:

CultureID INT PRIMARY KEY, - Унікальний ідентифікатор культури.

Name VARCHAR(255) NOT NULL, - Назва культури.

SowingDate DATE, - Дата висіву.

HarvestDate DATE, - Дата збору врожаю.

Area DECIMAL(10, 2), - Площа посівів (в гектарах).

Yield DECIMAL(10, 2) - Врожайність (тонн/га).

Таблиця «ClimateConditions», має такі поля:

ConditionID INT PRIMARY KEY, - Унікальний ідентифікатор кліматичних умов.

Temperature DECIMAL(5, 2), - Температура (°C).

Humidity DECIMAL(5, 2), - Вологість (%).

Precipitation DECIMAL(5, 2), - Кількість опадів (мм).

Date DATE, - Дата.

CultureID INT, - Ідентифікатор культури (Foreign Key).

FOREIGN KEY (CultureID) REFERENCES Cultures(CultureID) - Зв'язок з таблицею культур.

Таблиця «AgronomicMeasures», має такі поля:

MeasureID INT PRIMARY KEY, - Унікальний ідентифікатор агротехнічного заходу.

FertilizersUsed VARCHAR(255), - Використані добрива.

SoilTreatment VARCHAR(255), - Обробка ґрунту.

PestControl VARCHAR(255), - Засоби захисту рослин.

Date DATE, - Дата проведення заходу.

CultureID INT, - Ідентифікатор культури (Foreign Key) FOREIGN KEY. (CultureID) REFERENCES Cultures(CultureID) - Зв'язок з таблицею культур.

Таблиця «EconomicIndicators», має такі поля:

IndicatorID INT PRIMARY KEY, - Унікальний ідентифікатор економічного показника.

ProductionCost DECIMAL(15, 2), - Витрати на виробництво.

FinancialResults DECIMAL(15, 2), - Фінансові результати.

ProductPrice DECIMAL(15, 2), - Ціни на продукцію.

Date DATE, - Дата.

CultureID INT, - Ідентифікатор культури (Foreign Key).

FOREIGN KEY (CultureID) REFERENCES Cultures(CultureID) - Зв'язок з таблицею культур.

3.3 Вибір інструментарію для створення прикладного програмного забезпечення

PHP - це популярна мова сценаріїв, яка широко використовується для веб-розробки. Це мова з відкритим кодом, яка виконується на сервері для створення динамічних веб-сторінок. Його розроблено для вбудовування в код HTML, що полегшує інтеграцію серверної логіки з рівнем презентації. Він має синтаксис, подібний до C, Java та інших мов, що робить його відносно легким для вивчення розробникам із попереднім досвідом програмування. Його простота та інтуїтивно зрозумілий характер дозволяють розробникам швидко осягнути основи та розпочати створення веб-додатків. Ця мова була спеціально розроблена для веб-розробки. Вона надає багатий набір можливостей і функцій, призначених для вирішення типових веб-завдань, таких як обробка форм, керування сеансами, інтеграція бази даних і обробка файлів. Вона також підтримує широкий спектр веб-протоколів і фреймворків [11].

PHP-код виконується на сервері, що означає, що сервер обробляє код і надсилає отриманий HTML-код у веб-браузер клієнта. Це виконання на стороні

сервера дозволяє створювати динамічний вміст і взаємодіяти з базами даних, що робить його добре придатним для створення інтерактивних веб-сайтів, керованих даними.

Ця мова підтримується на багатьох платформах, включаючи Windows, Linux, macOS і Unix. Ця крос-платформна сумісність дозволяє розробникам розгортати програми PHP на різних веб-серверах без значних проблем із сумісністю.

PHP має величезну й активну спільноту розробників по всьому світу. Ця активна спільнота надає обширну документацію, онлайн-ресурси, форуми та бібліотеки, що полегшує пошук підтримки, обмін знаннями та використання існуючого коду для прискорення розробки.

Мова чудово підтримує роботу з базами даних. Вона пропонує вбудовані функції та розширення для взаємодії з різними системами керування базами даних, такими як MySQL, PostgreSQL і SQLite. Це дозволяє розробникам ефективно зберігати, отримувати та маніпулювати даними.

PHP можна легко розширити за допомогою модулів, розширень і бібліотек. Існує широкий спектр доступних фреймворків PHP, таких як Laravel, Symfony і CodeIgniter, які надають додаткові функції та допомагають оптимізувати процес розробки.

Загалом, PHP є універсальною та широко поширеною мовою для веб-розробки. Його простота у використанні, веб-функції, сумісність між платформами та сильна підтримка спільноти роблять його підходящим вибором для створення динамічних та інтерактивних веб-додатків.

Враховуючи наведені аргументи та мій особистий досвід, було обрано мову програмування PHP як основну для розробки, а також використання технології, що підтримує створення компонентів у PHP. Цей вибір обумовлений перевагами, які надає мова, такими як простота вивчення та використання, велика спільнота розробників, інтеграція з базами даних та широкий вибір фреймворків та розширень для прискорення розробки.

Python — це інтерпретована високорівнева мова програмування, яка завдяки своїй універсальності є популярною серед розробників. Її можливості охоплюють аналіз даних, машинне навчання, розробку веб-додатків, автоматизацію завдань та роботу з базами даних.

Особливості Python.

Простота використання:

Завдяки інтуїтивному синтаксису Python ідеально підходить як для новачків, так і для досвідчених розробників. Це дозволяє швидко розробляти програми без необхідності занурюватися в складні концепції низькорівневого коду.

Широкий екосистемний набір бібліотек:

Python має багатий набір бібліотек, які підтримують всі етапи розробки вашого проєкту:

Для аналізу даних та статистики:

- Pandas для роботи з табличними даними.
- NumPy для виконання математичних обчислень.
- Для моделювання та прогнозування:
- Scikit-learn — бібліотека для машинного навчання.
- TensorFlow або PyTorch — фреймворки для роботи зі штучними нейронними мережами.

Для візуалізації даних:

- Matplotlib і Seaborn забезпечують побудову графіків і діаграм, що ілюструють прогнози або аналіз кліматичних і ґрунтових показників.

Для роботи з базами даних:

- MySQL Connector і SQLAlchemy дозволяють інтегрувати Python із реляційними базами даних.

Кросплатформенність:

Python може працювати на різних операційних системах (Windows, Linux, macOS), що спрощує розробку та тестування.

Інтеграція з іншими інструментами:

Python легко інтегрується з іншими технологіями, такими як веб-фреймворки (Flask, Django), бібліотеки для розгортання додатків (наприклад, Docker), або сервіси хмарного зберігання.

Також для розробки даної системи використовувались такі мови як, MySQL - це популярна система керування реляційними базами даних з відкритим вихідним кодом, яка використовує мову структурованих запитів (SQL) як мову для керування базами даних та JavaScript - це широко поширена мова програмування, яка в основному використовується для зовнішньої веб-розробки. Це інтерпретована мова високого рівня з динамічною типізацією, що означає, що розробники можуть писати та виконувати код безпосередньо без необхідності компіляції. JavaScript в основному виконується веб-браузерами, що забезпечує інтерактивну та динамічну функціональність веб-сторінок [12].

Були застосовані такі мови розмітки як, HTML - стандартна мова розмітки, яка використовується для створення структури та представлення вмісту в Інтернеті. Вона надає набір тегів або елементів, які визначають структуру та семантику веб-сторінки [13]. Ось деякі ключові аспекти HTML:

Структура та семантика: HTML використовує теги для визначення структури веб-сторінки, такої як заголовки, абзаци, списки, таблиці та форми.

Гіперпосилання: HTML дозволяє створювати гіперпосилання або посилання, які можна натиснути, за допомогою яких можна переходити на інші веб-сторінки чи ресурси.

Зображення та медіа: HTML дозволяє включати зображення, відео та аудіо на веб-сторінку.

Форми та введення користувачами: HTML надає такі елементи форми, як «input», «select» і «textarea» для створення інтерактивних форм, які дозволяють користувачам вводити дані

Сумісність із різними браузерами: HTML підтримується всіма основними веб-браузерами, що забезпечує узгоджене відображення веб-сторінок на різних платформах і пристроях.

CSS - це мова таблиць стилів, яка використовується для опису візуального представлення та компонування документів HTML. Він працює в поєднанні з HTML для стилізації та форматування вмісту на веб-сторінці [14]. Ось деякі ключові аспекти CSS:

Селектори: CSS використовує селектори для націлювання на елементи HTML і застосування до них стилів.

Стилі та властивості: CSS дозволяє визначати різні стилі та властивості для цільових елементів HTML.

Каскадність і специфіка: CSS дотримується правила каскаду, що означає, що до одного елемента можна застосовувати кілька стилів, і вони вирішуються на основі специфічності та порядку.

Макет і позиціонування: CSS надає низку методів керування макетом і позиціонуванням елементів на веб-сторінці.

Bootstrap 4 - популярний інтерфейсний фреймворк, який спрощує та прискорює процес розробки адаптивних веб-сайтів, орієнтованих на мобільні пристрої. Він надає набір компонентів, стилів і утиліт CSS і JavaScript, які можна легко інтегрувати у веб-проекти [15].

При створенні програмного продукту використовувалися сучасні інструментальні засоби розробки:

Microsoft Visual Studio 2019 - це інтегроване середовище розробки, надане Microsoft. Visual Studio широко використовується для розробки різноманітних додатків, включаючи настільні, веб-, мобільні та хмарні програми. Він підтримує кілька мов програмування, таких як C#, Visual Basic, C++ і JavaScript. Visual Studio пропонує низку функцій, включаючи редагування коду, налагодження, інтеграцію контролю версій та інструменти керування проектами [16].

PhpMyAdmin - це веб-інструмент, який використовується для керування базами даних MySQL. Він надає графічний інтерфейс користувача, який дозволяє користувачам виконувати різні операції з базою даних, такі як створення баз даних, таблиць і запитів, імпорт і експорт даних і керування дозволами користувача [17].

PhpMyAdmin спрощує адміністрування та обслуговування баз даних MySQL, що полегшує розробникам роботу з базами даних у своїх програмах на основі PHP.

MAMP - означає Mac, Apache, MySQL і PHP. Це програмний стек, який використовується для локальної веб-розробки в операційних системах macOS і Windows. MAMP забезпечує середовище, яке включає веб-сервер Apache, сервер баз даних MySQL і мову програмування PHP. Це дозволяє розробникам налаштувати локальний веб-сервер і перевірити свої програми на основі PHP перед розгортанням їх на реальному сервері. MAMP також пропонує додаткові функції, такі як налаштування віртуальних хостів і керування розширеннями PHP [18].

Sublime - популярний текстовий редактор, відомий своєю швидкістю та простотою. Він легкий і легко налаштовується, що робить його кращим вибором для багатьох розробників. Sublime Text підтримує різні мови програмування та пропонує такі функції, як підсвічування синтаксису, фрагменти коду, множинний вибір і потужні можливості пошуку та заміни. Він також має яскраву екосистему плагінів, яка дозволяє розробникам розширювати його функціональність відповідно до своїх потреб [19].

Photoshop CS6 - це професійне програмне забезпечення для редагування зображень, яке широко використовується в графічному дизайні та веб-розробці. Він надає повний набір інструментів і функцій для маніпулювання та покращення зображень, створення візуального дизайну та оптимізації графіки для веб-додатків. Photoshop CS6 пропонує такі розширені можливості, як шари, фільтри, ретушування зображень і підтримку різних форматів файлів. Він часто використовується розробниками для створення та редагування графічних ресурсів для веб-сайтів і програм [20].

Figma - це хмарний інструмент для проектування та створення прототипів, який використовується для створення користувальницьких інтерфейсів і проектів спільного дизайну. Це дозволяє дизайнерам і розробникам

створювати та ділитися інтерактивними прототипами, розробляти макети та співпрацювати в режимі реального часу [21].

Figma пропонує низку функцій, таких як інструменти для редагування векторів, бібліотеки компонентів, керування версіями дизайну та легку співпрацю, що робить її популярним вибором для дизайну UI/UX та розробки інтерфейсу.

Git - це розподілена система контролю версій, яка широко використовується в розробці програмного забезпечення. Це дозволяє розробникам відстежувати зміни у вихідному коді, співпрацювати з членами команди та керувати різними версіями проекту. Git надає такі функції, як розгалуження, злиття та вирішення конфліктів, забезпечуючи ефективну співпрацю та керування кодом. Розробники використовують Git для відстеження змін свого коду, роботи з різними гілками та легкої синхронізації свого коду з віддаленим репозиторієм [22].

Ці інструменти відіграють важливу роль у сучасній розробці програмного забезпечення, надаючи розробникам необхідні інструменти та середовища для ефективного створення, керування та розгортання програмних додатків.

3.4 Загальні поняття напрямку OLAP-технології

OLAP (On-Line Analytical Processing) — це технологія аналітичної обробки даних, яка використовується для підтримки рішень на основі багатовимірного аналізу великих обсягів даних. Вона забезпечує швидкий доступ до агрегованих даних та їх аналіз з різних перспектив [23].

Основні поняття OLAP

1. Багатовимірна модель даних OLAP дозволяє працювати з багатовимірними даними, які представляються у вигляді багатовимірного куба. Основні елементи цієї моделі:

    - Виміри (Dimensions): це атрибути даних, за якими проводиться аналіз (наприклад, час, регіон, категорія продукту).

- Міри (Measures): числові показники, які аналізуються (наприклад, дохід, кількість продажів, витрати).
- Ієрархії: рівні деталізації вимірів, які дають змогу переходити між узагальненням і деталізацією (наприклад, рік → квартал → місяць → день).

2. OLAP-куб Це багатовимірна структура, яка дозволяє зберігати агреговані дані та швидко виконувати аналітичні запити. Куб містить:
    - Комірки (Cells): кожна комірка представляє одну агреговану величину для певної комбінації вимірів.
    - Осі куба: кожна вісь відповідає виміру даних.
3. Агрегування OLAP системи використовують функції агрегації (сума, середнє, максимум, мінімум) для отримання зведених показників за різними вимірами. Це дозволяє зменшити обсяг даних, які аналізуються, і підвищити продуктивність.
4. Розрізи (Slices) та підмножини (Dicing)
    - Slice: вибір одного шару даних для аналізу. Наприклад, аналіз продажів за один рік.
    - Dice: вибір певної підмножини даних. Наприклад, аналіз продажів певної категорії в конкретному регіоні за декілька років.
5. Деталізація та узагальнення (Drill-Down/Drill-Up)
    - Drill-Down: перехід від загального вигляду даних до детального. Наприклад, перегляд продажів не за квартал, а за місяць.
    - Drill-Up: перехід від деталізованих даних до узагальнених (наприклад, аналіз на рівні регіону замість міста).

6. Pivoting Це зміна структури відображення даних в OLAP-кубі, наприклад, переміщення одного виміру з рядків до стовпців для зручнішого аналізу.

Реалізувати багатовимірний аналіз для підтримки прийняття рішень у рослинництві. OLAP-система спрямована на:

- Аналіз врожайності за роками, регіонами та культурами.
- Вплив кліматичних умов і агротехнічних практик на врожайність.
- Оптимізацію агротехнічних заходів.

В моїй OLAP-системі є такі виміри (Dimensions):

1. Час (Time):
    - Рік.
    - Місяць.
    - Сезон (весна, літо, осінь, зима).
2. Культура (Crop):
    - Назва культури.
    - Тип культури (зерно, овочі тощо).
3. Регіон (Region):
    - Область.
    - Район.
4. Кліматичні умови (Climate):
    - Температура.
    - Вологість.
    - Кількість опадів.
5. Ґрунтові показники (Soil):

- Рівень вологи.
- Концентрація поживних речовин.

6. Агротехнічні практики (Practices):
    - Тип практики.
    - Частота виконання.

Та є такі міри (Measures):

1. Прогнозована врожайність (Predicted_Yield).
2. Фактична врожайність (Actual_Yield).
3. Середня температура та вологість.
4. Витрати на агротехнічні заходи.

Реалізація в MySQL

Приклад створення агрегованого запиту:

```
SELECT
    YEAR(yp.Year) AS Year,
    r.Region_Name AS Region,
    c.Crop_Name AS Crop,
    AVG(cc.Temperature) AS Avg_Temperature,
    SUM(yp.Predicted_Yield) AS Total_Predicted_Yield,
    AVG(si.Moisture_Level) AS Avg_Soil_Moisture
FROM
    Yield_Prediction yp
JOIN Crops c ON yp.Crop_ID = c.Crop_ID
JOIN Climate_Conditions cc ON c.Crop_ID = cc.Crop_ID
JOIN Soil_Indicators si ON c.Crop_ID = si.Crop_ID
JOIN Regions r ON cc.Region_ID = r.Region_ID
GROUP BY
```

YEAR(yp.Year), r.Region_Name, c.Crop_Name;

Практичний приклад:

1. Аналітик хоче знати, як опади та температура вплинули на врожайність кукурудзи у 2023 році в Полтавській області.
2. OLAP-запит обчислює середню температуру, кількість опадів та врожайність для цієї культури в обраному регіоні.
3. Результат: Якщо середня температура була 20°C, а кількість опадів 500 мм, прогнозована врожайність становила 6 т/га.

3.5 Загальні поняття технології Data Mining

Data Mining (видобуток даних) — це процес виявлення прихованих закономірностей, знань і корисної інформації у великих масивах даних за допомогою статистичних, математичних методів, штучного інтелекту та машинного навчання. Ця технологія дозволяє трансформувати необроблені дані в цінну інформацію, яка може бути використана для прийняття рішень у різних галузях [24].

Основні етапи Data Mining

1. Збір і підготовка даних:

- Збір даних із різних джерел (бази даних, веб-сайти, сенсори тощо).
- Очищення даних від шуму, непотрібної або неповної інформації.
- Нормалізація та перетворення даних у формат, зручний для аналізу.

2. Вибір завдання:

- Визначення мети аналізу, наприклад:
- Прогнозування (наприклад, врожайність за кліматичними умовами).
- Класифікація (розподіл культур за типами).
- Кластеризація (групування регіонів за умовами ґрунту).

3. Побудова моделі:

- Застосування алгоритмів машинного навчання (регресія, дерева рішень, нейронні мережі тощо).

- Налаштування моделей для максимальної точності.

4. Оцінка результатів:

- Перевірка якості моделі на тестових даних.
- Інтерпретація отриманих закономірностей.

5. Впровадження:

- Використання результатів аналізу для прийняття рішень (автоматизація процесів, розробка стратегій).

Основні методи Data Mining

1. Класифікація:

- Визначення категорій для нових даних на основі навчальних вибірок.
- Наприклад, класифікація рослин за потребами у волозі.

2. Кластеризація:

- Групування даних за схожими характеристиками.
- Наприклад, об'єднання регіонів із подібними кліматичними умовами.

3. Асоціативний аналіз:

- Пошук закономірностей між різними атрибутами даних.
- Наприклад, виявлення, які агротехнічні заходи підвищують врожайність.

4. Прогнозування:

- Використання історичних даних для передбачення майбутніх результатів.
- Наприклад, прогнозування врожайності на основі кліматичних даних.

5. Виявлення аномалій:

- Ідентифікація відхилень у даних, які можуть вказувати на проблеми.
- Наприклад, виявлення незвично низької врожайності на окремих полях.

4 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

4.1 Дослідження використання задач класифікації

Класифікація — це один із основних методів машинного навчання, що належить до задач з учителем (supervised learning). Її метою є віднесення об'єкта (заданого вектором ознак) до одного з попередньо визначених класів. Основою класифікації є використання навчального набору даних, який містить приклади з відомими класами.

Класифікація використовується для прогнозування якісних (категорійних) змінних. Завдання класифікації часто застосовують у бізнесі, медицині, промисловості та агротехніці.

Етапи виконання задач класифікації

1. Підготовка даних:
    - Збір даних із різних джерел.
    - Очищення даних від шумів та пропусків.
    - Перетворення категорійних даних у числові (наприклад, використання one-hot encoding).
2. Розділення набору даних:
    - Навчальний набір (training set) використовується для побудови моделі.
    - Тестовий набір (testing set) — для оцінки її ефективності.
3. Побудова моделі:

    Використання алгоритму класифікації, такого як:

    - Логістична регресія.
    - Метод опорних векторів.
    - Нейронні мережі.

- Рішення дерев або Random Forest.
- K-ближчих сусідів.

4. Оцінка якості моделі:

   Використання метрик, таких як:

   - Точність (Accuracy).
   - Повнота (Recall).
   - F-мера.
   - Площа під ROC-кривою.

5. Інтерпретація результатів:

   - Аналіз передбачених класів та інтерпретація для подальшого застосування.

4.2 Використання 1-Rule для класифікації

1-Rule (OneR) — це простий алгоритм класифікації, який будує модель на основі лише одного атрибуту. Алгоритм вибирає атрибут, який забезпечує найкращу класифікацію, та створює правила для кожного його значення. Хоча OneR є базовим методом, він часто демонструє несподівано хороші результати на невеликих наборах даних.

Принцип роботи 1-Rule

Алгоритм OneR виконує такі кроки:

1. Обчислення частоти: Для кожного значення кожного атрибута визначається, який клас зустрічається найчастіше.
2. Побудова правил: Для кожного значення атрибуту створюється правило, що визначає клас.
3. Оцінка похибки: Вибирається атрибут із найменшою кількістю помилок класифікації.
4. Прогнозування: Нові об'єкти класифікуються на основі створених правил.

Застосування OneR для класифікації у системі

У системі підтримки прийняття рішень у рослинництві алгоритм OneR було використано для:

1. Класифікації культур за типами ґрунту.
2. Прогнозування ризиків хвороб культур на основі кліматичних умов.
3. Визначення оптимальної агротехнічної практики для культури.

Реалізація 1-R на Python

Маємо таблицю з атрибутами:

- Temperature: Температура.
- Humidity: Вологість.
- Soil_Type: Тип ґрунту.
- Crop_Type: Тип культури (клас, який треба передбачити).

Код реалізує алгоритм OneR, розділений на кілька ключових етапів:

1. Ініціалізація даних: Створюється набір даних із характеристиками, наприклад, температура, тип ґрунту тощо, та класами (наприклад, тип культури).
2. Розділення на навчальний і тестовий набори: Дані діляться на дві частини: навчальні (для побудови правил) та тестові (для перевірки точності класифікації).
3. Реалізація алгоритму OneR:
    - Обчислення правил: Для кожного атрибуту обчислюється частота класів. Наприклад, для кожного типу ґрунту визначається, який клас (тип культури) є найбільш поширеним.
    - Вибір найкращого атрибуту: Оцінюється помилка класифікації (кількість невірно передбачених класів). Атрибут із найменшою помилкою обирається як основний.
4. Прогнозування: Створені правила застосовуються до тестових даних для передбачення класів.
5. Оцінка точності: Визначається, наскільки добре алгоритм передбачає класи, використовуючи метрику точності.

На рис.1 зображено створення таблиць з прикладами атрибутів (температура, вологість, тип ґрунту) і класу (тип культури).

```
data = {
    'Temperature': [35, 30, 20, 25, 28, 22, 33, 26, 19, 31, 23, 29],  # Температура
    'Humidity': [80, 60, 40, 55, 70, 45, 75, 50, 35, 65, 48, 58],  # Вологість
    'Soil_Type': ['Sandy', 'Clay', 'Loamy', 'Clay', 'Silt', 'Sandy', 'Clay', 'Peaty',
    'Silt', 'Loamy', 'Peaty', 'Sandy'],  # Грунт
    'Crop_Type': ['Wheat', 'Corn', 'Wheat', 'Barley', 'Soybean', 'Corn', 'Barley',
    'Soybean', 'Wheat', 'Corn', 'Barley', 'Soybean']  # Культури
}
```

Рис.1 Створення таблиць з прикладами атрибутів

На рис.2 зображено розділ даних на дві частини: навчальну (75%) і тестову (25%) для перевірки моделі.

```
X = df[['Temperature', 'Humidity', 'Soil_Type']]
y = df['Crop_Type']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

Рис.2 Розділ даних на дві частини

На рис.3 зображено вибір атрибутів із найменшою кількістю помилок.

```
for column in X_train.columns:
    rules = {}
    for value in X_train[column].unique():
        # Найбільш частий клас для кожного значення
        most_common_class = y_train[X_train[column] == value].mode()[0]
        rules[value] = most_common_class

    # Обчислення помилок
    errors = sum(
        y_train[X_train[column] == value] != most_common_class
        for value, most_common_class in rules.items()
    )

    if errors < best_error:
        best_attribute = column
        best_rules = rules
        best_error = errors
```

Рис.3 Вибір атрибутів із найменшою кількістю помилок

На рис.4 зображено визначення найкращого атрибуту для класифікації і генерування відповідних правил.

```
best_attribute, best_rules = one_rule(X_train, y_train)
print("Найкращий атрибут:", best_attribute)
print("Правила:", best_rules)
```

Рис.4 Визначення найкращого атрибуту для класифікації.

На рис.5 зображено прогноз класів для тестових даних, використовуючи створені правила та обчислення точності передбачень порівняно з реальними класами.

```
def predict(X_test, best_attribute, best_rules):
    predictions = []
    for _, row in X_test.iterrows():
        predictions.append(best_rules.get(row[best_attribute], "Unknown"))
    return predictions

y_pred = predict(X_test, best_attribute, best_rules)
accuracy = accuracy_score(y_test, y_pred)
print("Точність класифікації:", accuracy)
```

Рис.5 Прогноз класів для тестових даних

Код реалізує базову класифікацію за допомогою OneR:

- Визначає найкращий атрибут для розділення даних.
- Створює прості правила класифікації.
- Прогнозує клас для нових даних і оцінює точність моделі.

Алгоритм OneR, хоч і простий, є хорошим підходом для початкового аналізу даних. У вашій темі його можна застосувати для класифікації культур за типами ґрунту, кліматичними умовами чи іншими показниками.

4.2 Використання методу наївного Байеса

Метод наївного Байеса (Naive Bayes) – це алгоритм класифікації, заснований на теоремі Байеса. Він використовується для визначення ймовірності належності об'єкта до певного класу на основі ознак. "Наївний" означає, що алгоритм припускає незалежність усіх ознак між собою.

На рис.6 зображена базова формула Байеса.

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)}$$

Рис.6. Базова формула Байеса

Де:

- P(C|X)– ймовірність, що об'єкт належить до класу C, за умови даних X;
- P(X|C)– ймовірність даних X за умови класу C;
- P(C)– апріорна ймовірність класу C;
- P(X)– ймовірність даних X (можна ігнорувати для порівняльного аналізу).

Простота реалізації: Метод підходить для великих наборів даних і швидко обчислює ймовірності для кожного класу.

Основні переваги:

Підходить для дискретних і неперервних змінних;

Швидка обробка та навчання на великих наборах даних.

Обмеження:

Припускає, що всі змінні незалежні, що не завжди відповідає реальним даним**.**

Прогнозування врожайності залежно від кліматичних умов і часу проведення агротехнічних робіт. Ми будемо класифікувати врожайність як "Висока", "Середня" або "Низька" залежно від кліматичних умов і часу агротехнічних робіт.

На рис.7 зображено створення таблиць з прикладами атрибутів (температура, вологість, час робіт) і класу (врожайність).

```
data = {
    'Temperature': [25, 30, 28, 22, 35, 18, 24, 29, 31, 21],
    'Humidity': [60, 70, 65, 55, 80, 50, 58, 72, 68, 45],
    'Work_Time': [8, 6, 7, 9, 5, 10, 8, 6, 7, 9],
    'Yield': ['High', 'Medium', 'High', 'Low', 'High',
              'Low', 'Medium', 'High', 'Medium', 'Low']
}
```

Рис.7 Створення таблиць з прикладами атрибутів

Де:

- Температура, вологість і час виконання агротехнічних робіт використовуються як вхідні ознаки.
- Врожайність класифікується на три категорії: High, Medium, Low.

На рис.8 зображено розділення даних на ознаки (X) та цільову змінну (y)

```
X = df[['Temperature', 'Humidity', 'Work_Time']]
y = df['Yield']
```

Рис.8 Розділення даних на ознаки (X) та цільову змінну (y)

Де:

- X: Матриця ознак (температура, вологість, час робіт).
- y: Мітки класів врожайності.

На рис.9 зображено розділення даних на навчальні і тестові набори.

```
X_train, X_test, y_train, y_test = train_test_split
(X, y, test_size=0.25, random_state=42)
```

Рис.9 Розділення даних на навчальні і тестові набори

Де:

- test_size=0.25: 25% даних виділяється для тестування.
- random_state=42: Фіксація випадковості для відтворюваності результатів.

На рис.10 зображено прогнозування та оцінка моделі.

```
y_pred = model.predict(X_test)
print("Точність моделі:", accuracy_score(y_test, y_pred))
print("\nЗвіт класифікації:")
print(classification_report(y_test, y_pred))
```

Рис.10 Прогнозування та оцінка моделі.

Де:

- accuracy_score: Обчислює точність моделі – частку правильно класифікованих випадків.
- classification_report: Виводить детальну оцінку:
- Precision: Точність (доля правильних прогнозів у вибірці даного класу).

- Recall: Повнота (здатність моделі знаходити всі приклади певного класу).
- F1-міра: Середнє гармонійне precision та recall.

Робота коду:

1. Підготовка даних:

- Завантажуються кліматичні умови, час агротехнічних робіт і врожайність.
- Розділяються вхідні ознаки та мітки класів.

2. Навчання моделі:

- Алгоритм наївного Байеса аналізує розподіл ознак у навчальних даних і будує ймовірнісну модель.

3. Прогнозування:

- Для тестових даних модель передбачає категорію врожайності.

4. Оцінка:

- Порівнюються передбачені значення з реальними, щоб оцінити точність.

Точність моделі: На основі тестових даних обчислюється, наскільки добре модель передбачає врожайність.

Звіт класифікації:

Містить метрики точності, повноти та F1-міри для кожного класу ("High", "Medium", "Low").

Опис алгоритму:

1. Навчання: Алгоритм вивчає зв'язок між кліматичними умовами (температура, вологість), часом робіт і класами врожайності.
2. Прогнозування: Для нових кліматичних умов і часу робіт модель визначає найімовірніший клас врожайності.
3. Застосування: Модель допомагає планувати оптимальні агротехнічні роботи, враховуючи клімат, для досягнення високої врожайності.

Наївний Байес є ефективним методом для швидкого прогнозування врожайності, оскільки він враховує як числові (температура, вологість), так і категорійні змінні (час робіт). Це дозволяє приймати обґрунтовані рішення для управління сільськогосподарськими процесами.

4.3 Використання Моделі ARIMA

Це метод для аналізу та прогнозування часових рядів. У рослинництві його використовують для оцінки та прогнозування сезонних змін врожайності на основі попередніх даних. ARIMA вивчає минулі показники та сезонні тенденції, що дозволяє агрономам планувати агротехнічні роботи з урахуванням очікуваних змін, таких як можливі періоди високої чи низької врожайності.

На рис.11 зображено створення часового ряду із даних про врожайність у тоннах за роки.

```
data = {
    'Year': [2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019],
    'Yield': [20, 22, 21, 24, 23, 25, 27, 26, 29, 30]
}
df = pd.DataFrame(data)
df.set_index('Year', inplace=True)
```

Рис.11 Прогнозування та оцінка моделі

На рис.12 зображено побудову моделі ARIMA

```
# Налаштування моделі ARIMA (p=1, d=1, q=1)
model = ARIMA(df['Yield'], order=(1, 1, 1))
model_fit = model.fit()

# Виведення результатів моделі
print(model_fit.summary())
```

Рис.12 Побудова моделі ARIMA

Де:

Використано параметри (p=1, d=1, q=1):

- p: порядок авторегресії.
- d: кількість диференціювань.
- q: порядок ковзного середнього.
- Модель адаптується до історичних даних для прогнозування.

На рис.13 зображено прогнозування майбутньої врожайності.

```
# Прогноз на наступні 5 років
forecast = model_fit.forecast(steps=5)
forecast_years = [2020, 2021, 2022, 2023, 2024]

# Формування таблиці прогнозів
forecast_df = pd.DataFrame({'Year': forecast_years, 'Predicted_Yield': forecast})
forecast_df.set_index('Year', inplace=True)
```

Рис.13 Прогнозування майбутньої врожайності

На рис.14 зображена оцінка точності.

```
train = df[:8]  # Перші 8 років для навчання
test = df[8:]   # Останні 2 роки для тестування

model_train = ARIMA(train['Yield'], order=(1, 1, 1))
model_train_fit = model_train.fit()

test_forecast = model_train_fit.forecast(steps=len(test))
mse = mean_squared_error(test, test_forecast)

print(f"Середньоквадратична помилка (MSE): {mse:.2f}")
```

Рис.14 Оцінка точності

Де:

- Дані розділяються на навчальні та тестові для перевірки точності.
- Використовується метрика середньоквадратичної помилки (MSE) для оцінки.

На рис.15 зображено приклад результату.

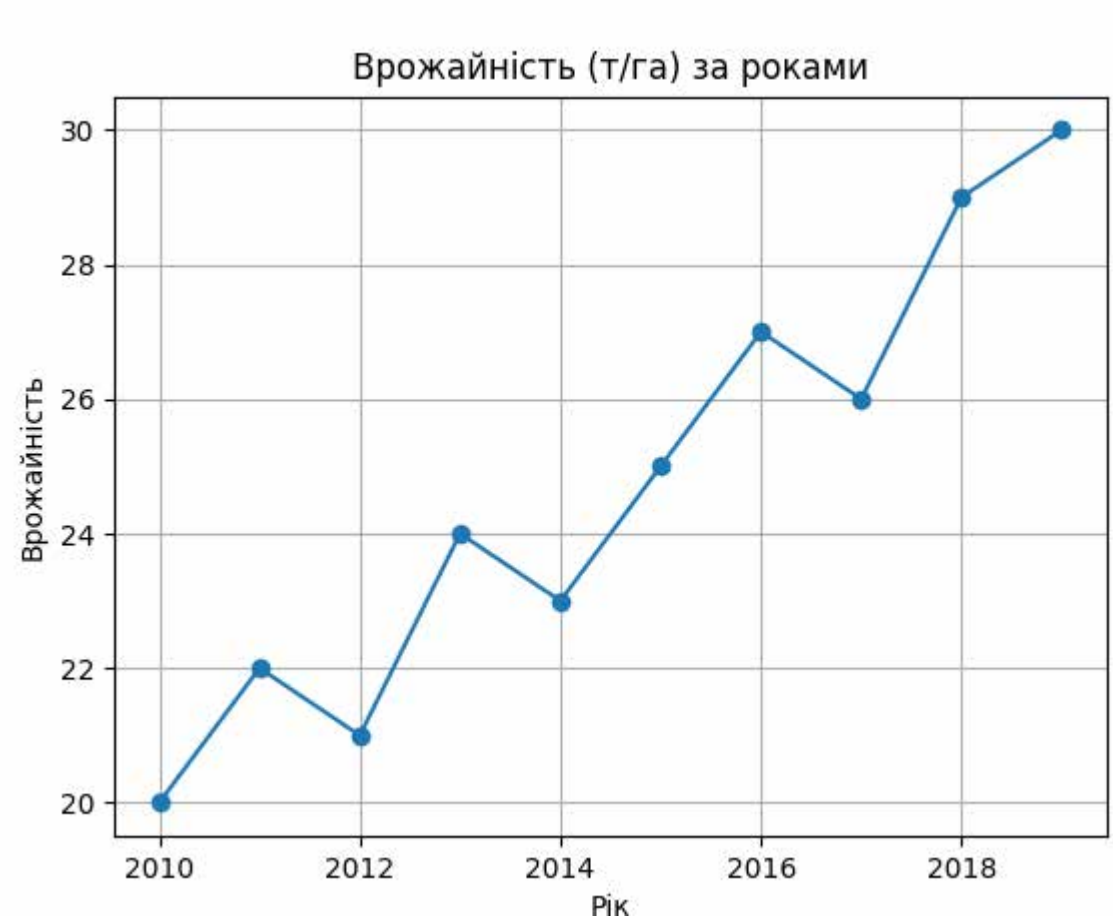

Рис.15 Приклад результату

Візуалізація покаже плавне зростання врожайності з урахуванням тренду. Цей прогноз може допомогти агрономам планувати майбутні роботи.

4.4 Використання алгоритмів кластеризації

K-Means — це популярний алгоритм кластеризації, який використовує ітеративний підхід для розділення даних на певну кількість груп (кластери). Кожен кластер містить схожі елементи, що дозволяє зібрати схожі поля, ґрунти або інші об'єкти для подальшого аналізу чи прийняття рішень. Алгоритм працює за принципом мінімізації відстані між точками в кожному кластері і їхнім центром (центроїдом). У агрономії та рослинництві це дозволяє ефективно групувати поля за характеристиками, наприклад, родючістю ґрунту чи вмістом поживних речовин.

Основні етапи алгоритму K-Means:

1. Вибір кількості кластерів K, яку ми хочемо отримати.
2. Вибір початкових центрів кластерів випадковим чином.
3. Призначення кожної точки (поля) до найближчого центру (кластеру).
4. Перерахунок центрів кластерів як середнього значення всіх точок, що належать до цього кластера.
5. Повторення кроків 3 і 4 до досягнення стабільності (коли центри кластерів більше не змінюються).

На рис.16 зображено створення даних та налаштування моделі K-Means.

```
data = {
    'Soil_Fertility': [7, 3, 8, 2, 5, 9, 4, 6, 1, 10],
    'Nutrient_Content': [6, 2, 7, 3, 5, 9, 4, 6, 2, 10]
}
df = pd.DataFrame(data)

kmeans = KMeans(n_clusters=3)
```

Рис.16 створення даних та налаштування моделі K-Means

В цьому коді ми створюємо DataFrame з двома стовпцями:

- рівень родючості
- вміст поживних речовин для кожного поля.

Вибираємо 3 кластери (K=3) для кластеризації полів.

На рис.17 зображено навчання моделі та візуалізація кластерів.

```
df['Cluster'] = kmeans.fit_predict(df[['Soil_Fertility', 'Nutrient_Content']])

print(df)

plt.scatter(df['Soil_Fertility'], df['Nutrient_Content'], c=df['Cluster'], cmap='viridis')
plt.xlabel('Soil Fertility')
plt.ylabel('Nutrient Content')
plt.title('K-Means Clustering of Fields')
plt.show()
```

Рис.17 Навчання моделі та візуалізація кластерів

В цьому коді ми використовуємо метод fit_predict(), щоб класифікувати поля в кластери на основі їх характеристик та використовуємо matplotlib для виведення розсіювання полів, де різні кольори вказують на різні кластери.

На рис.18 зображено результат виконання.

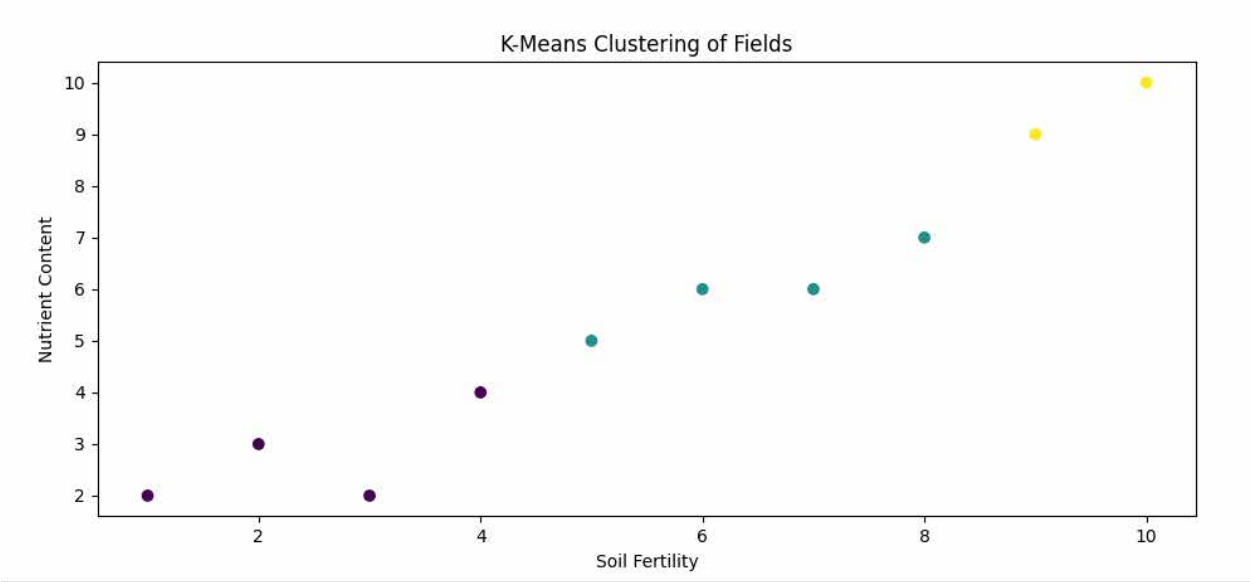

Рис.18 Результат виконання

Розподіл полів на кластери: Використовуючи метод K-Means для кластеризації, ми змогли розподілити поля за різними характеристиками, такими як рівень родючості ґрунту та вміст поживних речовин. Алгоритм групує поля з подібними властивостями в окремі кластери, що дозволяє чітко розрізняти різні типи земель.

Три основні групи:

- Поля з високим рівнем родючості та значним вмістом поживних речовин (жовтий кластер).

- Поля середнього рівня родючості, що потребують регулярної підтримки (зелений кластер).
- Поля з низьким рівнем родючості та низьким вмістом поживних речовин, які потребують додаткових агрономічних заходів (фіолетовий кластер).

K-Means допомагає агрономам або фахівцям з рослинництва згрупувати поля з подібними характеристиками (наприклад, родючість ґрунту, вміст поживних речовин), що дозволяє створювати більш персоналізовані стратегії догляду та оптимізації витрат на добрива та інші агрономічні заходи.

4.5 Використання методів асоціативних правил

Метод асоціативних правил широко використовується для виявлення закономірностей і взаємозв'язків між різними аспектами, такими як агротехнічні заходи та врожайність. Це дозволяє агрономам планувати стратегії догляду за культурами з урахуванням оптимальних поєднань операцій, які позитивно впливають на продуктивність.

Основи методу асоціативних правил.

Метод асоціативних правил дозволяє знаходити правила типу "якщо X, то Y", де:

- X — це набір умов (наприклад, певний тип поливу, застосування добрив),
- Y — це результат або подія (наприклад, зміна врожайності).

Це дозволяє зрозуміти, як різні агротехнічні заходи можуть впливати на врожайність культури в певних умовах.

На рис.19 зображено створення набору даних, де кожен запис описує агротехнічні заходи (тип поливу, тип добрив) та результат (врожайність).

```
data = {
    'Irrigation': ['Drip', 'Flood', 'Sprinkler', 'Drip', 'Flood', 'Sprinkler', 'Drip', 'Sprinkler'],
    'Fertilizer': ['Organic', 'Chemical', 'Organic', 'Chemical', 'Organic', 'Chemical', 'Organic', 'Chemical'],
    'Crop_Type': ['Wheat', 'Corn', 'Wheat', 'Barley', 'Corn', 'Wheat', 'Barley', 'Corn'],
    'Yield': ['High', 'Low', 'High', 'Medium', 'Low', 'High', 'Medium', 'Low']
}

df = pd.DataFrame(data)
```

Рис.19 Створення набору даних

Ми створюємо простий набір даних, де кожен запис описує агротехнічні заходи (тип поливу, тип добрив) та результат (врожайність).

Кодування даних: Використовуємо метод get_dummies() для перетворення категоріальних змінних у бінарний формат, що є необхідним для алгоритму Apriori.

На рис.20 зображено кодування даних та виконання алгоритму Apriori.

```
# Підготовка даних для алгоритму Apriori
df_encoded = pd.get_dummies(df)

# Виконання алгоритму Apriori для виявлення частих наборів
frequent_itemsets = apriori(df_encoded, min_support=0.3, use_colnames=True)
```

Рис.20 Кодування даних та виконання алгоритму Apriori

Алгоритм Apriori знаходить часті набори елементів у нашому наборі даних. Параметр min_support=0.3 вказує, що ми хочемо знайти асоціації, які трапляються хоча б у 30% записів.

На рис.21 зображена генерація асоціативних правил.

```
# Створення асоціативних правил
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)

# Виведення асоціативних правил
print(rules)
```

Рис.21 Генерація асоціативних правил

Використовуємо функцію association_rules(), щоб отримати правила, що описують взаємозв'язки між агротехнічними заходами і результатами врожайності.

На рис.22 зображені результати виконання.

Antecedents	Consequents	Lift	Support	Confidence
{Irrigation=Drip}	{Yield=High}	1.5	0.375	0.75
{Irrigation=Flood}	{Yield=Low}	2.0	0.375	0.75
{Fertilizer=Organic}	{Yield=High}	2.5	0.375	0.75
{Fertilizer=Chemical}	{Yield=Low}	3.0	0.375	0.75

Рис.22 Результати виконання

Опис виводу:

- antecedents: Умова для правила (наприклад, тип поливу "Drip").

- consequents: Результат або наслідок, який ми прогнозуємо (наприклад, врожайність "High").
- lift: Міра покращення прогнози. Чим більше lift, тим сильніше зв'язок між умовою і результатом.
- support: Підтримка для правила (підтримка обох умов разом).
- confidence: Ймовірність того, що результат (врожайність) буде високим, якщо умова (тип поливу) виконується.

З допомогою асоціативних правил можна знайти важливі взаємозв'язки між агротехнічними заходами та результатами врожайності.

Наприклад, ми можемо побачити, що певні методи поливу (наприклад, "Drip") або типи добрив (наприклад, "Organic") сприяють високій врожайності.

Використання таких правил допомагає агрономам ефективно планувати роботи в полі, вибирати правильні методи поливу, підживлення та інші агрономічні заходи для покращення врожайності.

Таким чином, метод асоціативних правил є потужним інструментом для аналізу сільськогосподарських даних та підвищення ефективності агрономічних практик.

ВИСНОВКИ

Дослідження в межах даної роботи має важливе значення для розвитку рослинництва, оскільки зосереджено на створенні системи підтримки прийняття рішень (СППР), що дозволяє оптимізувати агротехнічні заходи та підвищити ефективність управління сільськогосподарськими процесами. У сучасних умовах змін клімату та обмежених ресурсів, оптимізація управління поливом, підживленням та доглядом за культурами є необхідністю для забезпечення стабільних врожаїв.

Метою дослідження було розробити програмне забезпечення СППР, яке поєднує сучасні методи аналізу даних, зокрема алгоритми кластеризації (K-Means), асоціативні правила та методи прогнозування, такі як Байєсів метод і модель ARIMA. Розроблене програмне забезпечення дозволяє здійснювати ефективний аналіз агротехнічних даних і надавати рекомендації для прийняття обґрунтованих рішень у реальному часі, що підвищує продуктивність сільськогосподарських підприємств.

Важливим результатом дослідження стало тестування розробленої системи на змодельованих даних, що підтвердило її ефективність у прогнозуванні врожайності та оптимізації агротехнічних заходів. Впровадження цієї системи дозволить агрономам значно знижувати витрати на ресурси, підвищувати врожайність та забезпечувати стійкість агровиробництва.

Практична значимість роботи полягає в створенні інструменту для підтримки прийняття рішень, який здатен інтегруватися з іншими технологіями та значно підвищити ефективність сільськогосподарського виробництва. Наукова новизна дослідження полягає у застосуванні комбінованих методів аналізу даних для розробки інтегрованої СППР, що є новаторським підходом для рослинництва.

Отже, розроблене програмне забезпечення має великий потенціал для практичного використання в аграрному секторі, сприяючи підвищенню його ефективності, економічної стабільності та конкурентоспроможності на ринку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Agro Ofice – [Електронний ресурс] – Режим доступу: https://agrooffice.pp.ua/

2. QGIS – [Електронний ресурс] – Режим доступу: https://www.qgis.org/

3. Smart Agri Hubs – [Електронний ресурс] – Режим доступу: https://www.smartagrihubs.eu/

4. Cropwise – [Електронний ресурс] – Режим доступу: https://www.cropwise.com/

5. Опис програмного забезпечення для проектування ER-діаграм - [Електронний ресурс] – Режим доступу: https://elib.lntu.edu.ua/sites/default/files/elib_upload/%D0%95%D0%9D%D0%9F_%D0%A1%D0%B0%D0%B2%D0%B0%D1%80%D0%B8%D0%BD_%D0%9B%D0%B5%D0%BF%D0%BA%D0%B8%D0%B9/practic/pr9.html

6. Для чого потрібні UML діаграми? – [Електронний ресурс] – Режим доступу: https://foxminded.ua/uml-diagramy/

7. Діаграми прецедентів – [Електронний ресурс] – Режим доступу: https://studfile.net/preview/9877319/

8. Діаграма послідовності – [Електронний ресурс] – Режим доступу: https://www.maxzosim.com/sequence-diagrams/

9. Діаграма розгортання – [Електронний ресурс] – Режим доступу: https://www.guru99.com/uk/deployment-diagram-uml-example.html

10. MySQL - система управління базами даних - [Електронний ресурс] – Режим доступу: https://www.websiterating.com/uk/web-hosting/glossary/what-is-mysql/

11. Чудові факти про PHP, які повинен знати кожний розробник - [Електронний ресурс] – Режим доступу: https://internetdevels.ua/blog/awesome-facts-about-php

12. Документація MySQL - [Електронний ресурс] – Режим доступу: https://dev.mysql.com/doc/

13. W3Schools - HTML Tutorial - [Електронний ресурс] – Режим доступу: https://www.w3schools.com/html/

14. W3Schools - CSS Tutorial - [Електронний ресурс] – Режим доступу: https://www.w3schools.com/css/

15. Bootstrap Documentation - [Електронний ресурс] – Режим доступу: https://getbootstrap.com/docs/4.0/getting-started/introduction/

16. Microsoft Visual Studio Documentation - [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/visualstudio/

17. Офіційний веб-сайт PhpMyAdmin - [Електронний ресурс] – Режим доступу: https://www.phpmyadmin.net/

18. MAMP Documentation - [Електронний ресурс] – Режим доступу: https://documentation.mamp.info/

19. Офіційний веб-сайт Sublime Text - [Електронний ресурс] – Режим доступу: https://www.sublimetext.com/

20. Adobe Photoshop User Guide - [Електронний ресурс] – Режим доступу: https://helpx.adobe.com/photoshop/user-guide.html

21. Figma Help Center - [Електронний ресурс] – Режим доступу: https://help.figma.com/

22. Pro Git Book - [Електронний ресурс] – Режим доступу: https://git-scm.com/book/en/v2