ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ, УПРАВЛІННЯ, ПРАВА ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Пояснювальна записка

до кваліфікаційної роботи на здобуття ступеня вищої освіти Бакалавр

на тему: «Розробка інтелектуальної системи поливу»

Виконав: здобувач вищої освіти
за освітньо-професійною програмою
Інформаційні управляючі системи
спеціальності 126 Інформаційні
системи та технології
ступеня вищої освіти Бакалавр
групи 126ІСТбд21[1](стн)
Даншин І.В.
Керівник: Слюсар В.І.
Рецензент: Брикун О.М.

Полтава – 2024 року

ВСТУП

Актуальність теми кваліфікаційної роботи підтверджується необхідністю зменшення витрат води шляхом оптимізації режимів поливу, що сприяє збереженню цього цінного ресурсу. Правильне зволоження ґрунту є ключовим фактором для забезпечення високої врожайності. Системи, які можуть автоматично налаштовуватися в залежності від погодних умов, типу ґрунту та потреб рослин, допомагають досягти оптимальних умов для їх росту. Це сприяє розвитку технологічного сектору та впровадженню інновацій у повсякденне життя. Однак питання інтеграції штучного інтелекту (AI) та інтернету речей (IoT) потребує додаткових досліджень. Все це свідчить про актуальність теми роботи.

Метою кваліфікаційної роботи є удосконалення системи автоматичного поливу на основі застосування штучного інтелекту.

Завданнями кваліфікаційної роботи є:

– аналіз особливостей використання штучного інтелекту в аграрному секторі;

– розробка моделі прогнозування на основі штучного інтелекту;

– обґрунтування рекомендацій щодо практичної реалізації системи інтелектуального поливу.

Об'єктом дослідження є процес автоматичного поливу в умовах змінних кліматичних та ґрунтових умов.

Предметом дослідження є алгоритми та методи інтелектуальної обробки даних для регулювання поливу

Методами дослідження є аналіз даних, моделювання, машинне навчання, оптимізація алгоритмів та економічний аналіз.

Інформаційна база кваліфікаційної роботи сформована з ресурсів, що містять інформацію про системи інтелектуального поливу на основі Інтернету речей та нейронних мереж.

Практична значущість роботи полягає у розробці система автоматичного поливу на основі штучного інтелекту – можуть бути

використані для подальших досліджень за даною тематикою та при проектуванні розумної ферми

Апробація результатів відбувалася в рамках науково-практичної конференції за підсумками проходження виробничої практики здобувачів вищої освіти спеціальності 126 Інформаційні системи та технології (26 лютого 2024 р., м. Полтава).

За результатами досліджень здійснено публікації тез доповідей.

Структура кваліфікаційної роботи логічно пов'язана з завданнями досліджень і містить вступ, три розділи основної частини, висновки, список використаних джерел, додатки. Загальний обсяг пояснювальної записки кваліфікаційної роботи складає 57 сторінок формату А4. Вона містить 29 рисунків 10 таблиць.

РОЗДІЛ 1
АНАЛІЗ ОСОБЛИВОСТЕЙ ВИКОРИСТАННЯ ШТУЧНОГО ІНТЕЛЕКТУ В АГРАРНІЙ СФЕРІ

1.1 Інтернет речей (IoT)

Інтернет речей (IoT) є концепцією, що стала центральною в сучасній інформаційній технології. За своєю суттю, IoT визначається як мережа фізичних пристроїв, які обмінюються даними та взаємодіють один з одним через Інтернет без прямого втручання людини. Ця концепція створює нові можливості для збору, обробки та аналізу великих обсягів даних з метою оптимізації процесів у різних галузях, включаючи промисловість, медицину, транспорт, побут та сільське господарство.

У промисловості IoT відіграє важливу роль у підвищенні ефективності та оптимізації виробничих процесів. Широке застосування знаходять «розумні» пристрої, обладнання та машини, що підключені до мережі, які забезпечують збір та обробку даних в режимі реального часу. Це дозволяє здійснювати моніторинг стану обладнання, прогнозувати виникнення несправностей та забезпечувати своєчасне обслуговування. Більше того, IoT сприяє автоматизації виробничих процесів та оптимізації ресурсів, що призводить до зниження витрат та підвищення продуктивності.

Загалом, розвиток Інтернету речей відкриває широкі можливості як для побутового, так і промислового секторів, прискорюючи цифрову трансформацію та сприяючи створенню більш ефективних та зручних середовищ життя та роботи [1].

Архітектура IoT зазвичай включає в себе три рівні.

Рівень датчиків та пристроїв. На цьому рівні знаходяться фізичні пристрої, які збирають дані з навколишнього середовища. Рівень мережі: Дані, зібрані на рівні пристроїв, передаються через мережеве обладнання до центральних серверів або хмарних сервісів для подальшої обробки.

Рівень додатків та сервісів. На цьому рівні знаходяться програмні засоби та алгоритми, які аналізують та обробляють дані, а також надають користувачам доступ до функціоналу IoT через вебінтерфейси або мобільні додатки.

Рівень збору та передачі даних. На цьому рівні здійснюється збір даних від сенсорів та їх передача через мережеве обладнання. Цей рівень включає в себе різноманітні протоколи зв'язку та стандарти передачі даних, такі як MQTT (Message Queuing Telemetry Transport), CoAP (Constrained Application Protocol), а також технології бездротового зв'язку, такі як Wi-Fi, Bluetooth, Zigbee, LoRa і NB-IoT.

Рівень аналізу та обробки даних. Дані, які були зібрані та передані на центральні сервери або хмарні сервіси, піддаються аналізу та обробці. На цьому рівні використовуються різноманітні алгоритми та моделі машинного навчання для виявлення патернів, прогнозування подій та прийняття рішень на основі цих даних. Також тут може здійснюватися агрегація даних з різних джерел та їх кореляція для отримання більш повного зображення ситуації.

Рівень управління та контролю. На цьому рівні відбувається управління фізичними пристроями на основі отриманих аналітичних даних. Це включає в себе віддалене керування сенсорами, актуалізаторами та іншими пристроями, відповідно до виявлених патернів та цілей системи. Управління може бути автоматизованим або здійснюватися операторами через відповідні інтерфейси [2].

Архітектура IoT може бути централізованою, де всі дані обробляються на центральних серверах, або розподіленою, де обробка даних відбувається на близьких до джерела даних пристроях (edge computing), що дозволяє зменшити затримки в передачі даних та заощадити пропускну здатність мережі.

Основними складовими IoT є фізичні пристрої, датчики та актуалізатори, які здатні сприймати інформацію з навколишнього середовища та взаємодіяти з ним. Ці пристрої можуть бути підключені до Інтернету і

надсилати зібрані дані на центральний сервер для подальшої обробки. Потім дані можуть бути використані для різноманітних цілей, таких як моніторинг стану об'єктів, аналіз та прогнозування різних явищ, автоматизація процесів та управління системами [3].

Складові IoT включають в себе наступні компоненти.

Фізичні пристрої. Це сенсори, актуалізатори та інші обладнання, що взаємодіють з фізичним середовищем. Сенсори здатні збирати дані про температуру, вологість, рівень освітленості, тиск та інші параметри навколишнього середовища. Актуалізатори здатні впливати на середовище, наприклад, регулювати температуру або включати/вимикати пристрої.

Мережеве обладнання. Для забезпечення зв'язку між фізичними пристроями та центральними серверами використовуються мережеві засоби, такі як маршрутизатори, комутатори та безпровідні точки доступу.

Центральні сервери та хмарні сервіси. Зібрані дані надсилаються на центральні сервери або зберігаються в хмарних сервісах для подальшої обробки, аналізу та зберігання. Тут можуть застосовуватися різноманітні алгоритми та моделі штучного інтелекту для отримання корисної інформації зі зібраних даних.

ПЗ та алгоритми. На цьому рівні використовуються різноманітні програмні засоби та алгоритми для керування та аналізу даних, що надходять від фізичних пристроїв. До програмного забезпечення входять інструменти для моніторингу, управління та взаємодії з IoT пристроями, а також платформи для збору, зберігання та обробки даних. Алгоритми машинного навчання та штучного інтелекту використовуються для аналізу великих обсягів даних та виявлення в них патернів та закономірностей.

Хмарні сервіси та обчислення в хмарі. Хмарні сервіси грають важливу роль у збереженні, обробці та аналізі великих обсягів даних, які збираються від пристроїв IoT. Вони забезпечують гнучкість та масштабованість, необхідні для розвитку великих IoT систем. Крім того, хмарні сервіси можуть забезпечувати доступ до різних сервісів та додатків через мережу Інтернет.

Безпека та конфіденційність. Забезпечення безпеки та конфіденційності даних є критичною складовою IoT. Це включає в себе захист від несанкціонованого доступу до пристроїв та даних, шифрування даних в перехідних мережах, аутентифікацію користувачів та пристроїв, а також застосування стандартів та протоколів безпеки.

Інтерфейси користувача та додатки. Для взаємодії з користувачами та операторами систем IoT використовуються різноманітні інтерфейси користувача та додатки. Це можуть бути веб-інтерфейси, мобільні додатки або спеціалізовані програмні інтерфейси (API), які дозволяють отримувати доступ до функціоналу та даних системи.

Інтеграція з існуючими системами. Багато систем IoT потребують інтеграції з існуючими платформами та системами. Це можуть бути системи управління виробництвом, CRM-системи, системи управління енергоефективністю та інші. Інтеграція забезпечує спільну роботу та обмін даними між різними системами для досягнення цілей бізнесу.

Ці складові разом створюють комплексну інфраструктуру для реалізації IoT систем, які забезпечують збір, обробку та аналіз даних в реальному часі, автоматизацію процесів та управління підключеними пристроями.

Інтернет речей включає в себе різноманітні застосування. В промисловості, він може використовуватися для моніторингу та підтримки обладнання, прогнозування збоїв та оптимізації виробничих процесів. У медицині, IoT дозволяє стежити за станом пацієнтів у реальному часі та надавати їм необхідну допомогу. У транспорті, він може використовуватися для відстеження руху транспортних засобів та управління транспортними потоками. У побуті, IoT може контролювати різні побутові пристрої та забезпечувати їх взаємодію для підвищення комфорту та безпеки.

Одним із головних викликів, який стоїть перед IoT, є забезпечення безпеки та конфіденційності зібраних даних. З огляду на великий обсяг інформації, що обробляється в мережі IoT, важливо забезпечити захист від несанкціонованого доступу та зловживань [4].

1.2 Штучний інтелект в агросекторі

Розвиток штучного інтелекту (ШІ) в агросекторі представляє собою захоплюючий шлях інновацій та технологічного прогресу. Початки використання ШІ в сільському господарстві сягають далекого минулого, але справжній прорив стався з наростанням доступності обчислювальних ресурсів та розвитком алгоритмів машинного навчання в останні десятиліття.

Одним із перших напрямів використання ШІ в агросекторі було застосування експертних систем для діагностики захворювань рослин та прогнозування врожайності. За допомогою правил, зібраних від експертів сільськогосподарської галузі, комп'ютерні програми могли робити висновки про стан рослин та рекомендувати відповідні заходи.

Протягом останніх десятиліть значний прогрес було зроблено у напрямку розвитку систем моніторингу та управління засобами механізації в агросекторі. Завдяки поєднанню сучасних технологій IoT та сенсорів, фермери тепер можуть в реальному часі отримувати дані про стан поля, погодні умови та врожайність [5].

Одним із ключових аспектів сучасного сільського господарства є ефективне моніторинг росту та стану рослин. Завдяки розвитку технологій ШІ, сільські господарства тепер мають доступ до різноманітних методів та інструментів для автоматизованого моніторингу та аналізу даних про рослини.

Один з найбільш важливих засобів моніторингу є використання дронів. Дрони обладнані високоякісними камерами та сенсорами, які дозволяють збирати великі обсяги даних з високою роздільною здатністю. Ці дані можуть бути використані для створення зображень високої якості, а також для визначення різноманітних параметрів рослин, таких як розмір, колір та стан листя.

Крім того, сучасні технології ШІ дозволяють використовувати супутникові знімки для моніторингу росту рослин. За допомогою

спеціалізованих алгоритмів обробки зображень, які базуються на штучних нейронних мережах, можна автоматично визначати площу посівів, оцінювати врожайність та виявляти ознаки захворювань або стресу у рослин.

Нарешті, датчики росту використовуються для моніторингу різних параметрів рослин, таких як вологість ґрунту, температура повітря, рівень CO2 тощо. Ці дані збираються в реальному часі та надсилаються на спеціальні платформи для аналізу та прийняття рішень [6].

Автоматизація процесів виробництва у сільському господарстві є одним із ключових напрямків впровадження ШІ та новітніх технологій. Сучасні технології дозволяють фермерам та агрономам автоматизувати ряд процесів, що раніше вимагали значних зусиль та ресурсів.

Один з перших етапів автоматизації полягає в використанні автономних тракторів та машин для обробки ґрунту та вирощування культур. Ці технічні засоби обладнані системами навігації та управління, що дозволяють їм самостійно виконувати різноманітні операції на полі, такі як внесення добрив, обробка ґрунту та посів насіння.

Однак автоматизація не обмежується лише механізацією робіт на полі. Великий внесок роблять системи вирощування рослин в контрольованих умовах, такі як теплиці та вертикальні ферми. Ці системи автоматизовані від початкової стадії розвитку рослин до збору врожаю, що дозволяє забезпечити оптимальні умови для росту та максимальний врожай.

Одним з переваг автоматизації є збільшення продуктивності та зниження витрат на ресурси. За допомогою систем ШІ, фермери можуть ефективно використовувати ресурси, такі як вода, добрива та енергія, що дозволяє підвищити врожайність та знизити витрати [7].

Прогнозування погоди є критично важливою складовою сільського господарства, оскільки воно допомагає фермерам приймати обґрунтовані рішення щодо вирощування та догляду за рослинами. Використання ШІ у цій сфері дозволяє покращити точність та достовірність прогнозів, що відіграє ключову роль у забезпеченні високого рівня врожайності та ефективності

сільського господарства. Одним з методів використання ШІ для прогнозування погоди є аналіз великих обсягів даних з різних джерел, таких як супутникові знімки, метеорологічні станції, датчики та інші джерела. Ці дані обробляються за допомогою алгоритмів машинного навчання, які навчаються розпізнавати патерни та залежності у погодних явищах.

Також системи ШІ можуть використовувати прогностичні моделі для прогнозування погоди у майбутньому на основі поточних даних та історичних трендів. Ці моделі можуть оцінювати ризики пов'язані з погодними умовами, такі як засуха, негода чи заморозки, що дозволяє фермерам приймати вчасні заходи для захисту своїх посівів та врожаю.

Важливою складовою прогнозування погоди є також його інтеграція з іншими системами управління сільським господарством, такими як системи автоматизації поливу чи вибір сортів рослин. Це дозволяє фермерам оптимально використовувати доступні ресурси та максимально зменшити вплив негативних погодних умов на вирощування культур [8].

Ефективне управління ресурсами у сільському господарстві є ключовим аспектом забезпечення стійкого розвитку та високої продуктивності господарства. Використання ШІ у цій сфері дозволяє фермерам та агрономам оптимізувати використання води, добрив, енергії та інших ресурсів з максимальною ефективністю.

Одним із ключових напрямків використання ШІ є моніторинг вологості ґрунту та автоматизоване управління поливом. За допомогою сенсорів та систем моніторингу, фермери можуть в реальному часі отримувати дані про вологість ґрунту та визначати оптимальний час та об'єм поливу для кожної ділянки поля.

Крім того, системи ШІ можуть бути використані для розробки прогностичних моделей для управління водними ресурсами. Алгоритми машинного навчання можуть аналізувати історичні дані про споживання води та погодні умови для прогнозування оптимальних режимів поливу та зменшення втрат води.

Також системи ШІ можуть бути використані для оптимізації використання добрив та інших агрохімікатів. Аналіз даних про ґрунт, погодні умови та врожайність дозволяє розробляти індивідуальні плани добрив для кожної ділянки поля, що дозволяє зменшити витрати та мінімізувати вплив на навколишнє середовище [9].

Машинне навчання в сільському господарстві відіграє ключову роль у вирішенні складних завдань аналізу даних та прийнятті рішень. Застосування алгоритмів машинного навчання дозволяє фермерам та агрономам отримувати цінні інсайти з великих обсягів даних та приймати обґрунтовані рішення для оптимізації виробництва та підвищення врожайності.

Одним із застосувань машинного навчання є прогнозування врожайності та визначення оптимальних стратегій вирощування культур. За допомогою аналізу історичних даних про врожайність, погодні умови, використання ресурсів та інших факторів, можна створювати прогностичні моделі, які допомагають передбачати потенційні врожаї та розробляти стратегії їх вирощування.

Крім того, машинне навчання використовується для аналізу даних з сільськогосподарських сенсорів та IoT-пристроїв. За допомогою спеціалізованих алгоритмів можна виявляти патерни та залежності у зібраних даних, що дозволяє оптимізувати використання ресурсів, виявляти проблеми та шукати шляхи їх вирішення.

Також машинне навчання використовується для автоматизації процесів контролю якості та виявлення захворювань у рослин. За допомогою навчених моделей, які аналізують зображення рослин та виявляють ознаки захворювань або стресу, можна вчасно втручатися та запобігати поширенню хвороб, що дозволяє зберегти врожай та знизити втрати [10].

Обробка природних мов (Natural Language Processing, NLP) є важливою галуззю штучного інтелекту, яка знаходить широке застосування в сільському господарстві. За допомогою методів NLP, фермери та агрономи можуть аналізувати та використовувати тексти, що містять інформацію про погодні

умови, агрономічні рекомендації, ринкові дані та інші важливі аспекти для прийняття рішень.

Одним з застосувань NLP є аналіз новинних статей та медіа-звітів для отримання актуальної інформації про погодні умови, ринкові тренди та інші фактори, що можуть впливати на сільське господарство. За допомогою автоматичного збору та аналізу даних з різних джерел, фермери можуть бути в курсі останніх подій та приймати обґрунтовані рішення.

NLP може бути використана для аналізу агрономічних рекомендацій та наукових публікацій з метою виявлення нових технологій, методів та підходів у вирощуванні культур. За допомогою автоматичного обробки та класифікації текстів, можна швидко знаходити корисну інформацію та використовувати її для вдосконалення сільського господарства [11].

1.3 Інтеграція систем інтернету речей та штучного інтелекту

Інтернет речей стає все більш поширеним і важливим компонентом в сільському господарстві, дозволяючи фермерам моніторити, керувати та оптимізувати різні аспекти виробництва. Інтеграція систем IoT з технологіями ШІ відкриває нові можливості для автоматизації та управління господарством.

Одним з основних застосувань IoT в сільському господарстві є моніторинг умов середовища, таких як температура, вологість, рівень CO2 тощо. За допомогою датчиків, які встановлені на полі, у теплицях та на інших об'єктах, фермери можуть отримувати реальний час дані про стан навколишнього середовища та реагувати на зміни вчасно.

Системи IoT можуть бути використані для автоматизації процесів поливу, годівлі та управління врожаєм. За допомогою спеціальних пристроїв, які з'єднуються з мережею Інтернет, фермери можуть програмувати та

контролювати поливні системи, дозування добрив та інші аспекти вирощування культур.

Крім того, IoT може бути використана для створення "розумних" сільськогосподарських машин та обладнання, таких як трактори, комбайни та інші. Інтеграція датчиків та засобів зв'язку в машини дозволяє збирати дані про їх роботу та стан, що дозволяє вчасно виявляти та усувати проблеми [12].

У сучасному сільському господарстві, де все більше процесів стає автоматизованими та з'єднаними з мережею Інтернет, забезпечення кібербезпеки стає надзвичайно важливим аспектом. Зламані або скомпрометовані системи можуть призвести до великих втрат врожаю, фінансових втрат та порушення нормального функціонування господарства.

Одним з ключових аспектів кібербезпеки в сільському господарстві є захист мережевих пристроїв та систем IoT від несанкціонованого доступу. Це може включати в себе застосування сучасних методів аутентифікації та авторизації, шифрування комунікацій, а також використання програмних та апаратних засобів захисту.

Важливою складовою кібербезпеки є навчання персоналу господарства щодо правил безпеки та виявлення потенційних загроз. Регулярні навчання та оновлення навичок допомагають уникнути типових помилок та виявити підозрілу активність.

Забезпечення кібербезпеки також включає в себе регулярну оцінку потенційних загроз та вразливостей системи, а також вжиття заходів для їх усунення. Аудити безпеки, пентестування та інші методи дозволяють виявити слабкі місця в існуючій системі та прийняти заходи для їх запобігання [13].

Аналіз даних та виведення звітності є важливою складовою управління сільським господарством, оскільки дозволяє фермерам та агрономам отримувати цінні інсайти щодо ефективності вирощування, використання ресурсів та інших аспектів виробництва. Застосування методів аналізу даних та створення звітів допомагає приймати обґрунтовані рішення та покращувати продуктивність господарства. Одним з ключових етапів аналізу

даних є збір та підготовка даних для подальшого аналізу. Це може включати в себе збір даних з різних джерел, таких як сільськогосподарські сенсори, погодні станції, облікові системи господарства тощо, а також їх очищення, перевірку на достовірність та обробку для подальшого використання.

Після збору та підготовки даних проводиться їх аналіз за допомогою різних методів, таких як статистичний аналіз, машинне навчання, глибинне навчання тощо. Ці методи дозволяють виявляти патерни, залежності та тенденції у даних, що допомагає розуміти фактори, які впливають на виробництво та приймати обґрунтовані рішення.

Одним із важливих аспектів аналізу даних є виведення звітності та візуалізація результатів. Створення звітів та графіків дозволяє представити отримані дані у зрозумілій та доступній формі, що сприяє прийняттю рішень та спільному розумінню ситуації [14].

Оптимізація процесів управління господарством є ключовим аспектом сучасного сільського господарства, оскільки дозволяє підвищити ефективність виробництва, знизити витрати та підвищити прибутковість господарства. Застосування сучасних методів та технологій управління дозволяє фермерам та агрономам швидко реагувати на зміни у виробничих процесах та приймати обґрунтовані рішення для оптимізації виробництва.

Одним із аспектів оптимізації управління господарством є використання систем управління ресурсами (Enterprise Resource Planning, ERP). Ці системи дозволяють автоматизувати бізнес-процеси, такі як облік виробничих ресурсів, фінансовий облік, управління складами тощо, що сприяє підвищенню ефективності та зниженню витрат.

Крім того, оптимізація управління господарством передбачає використання аналітичних та прогностичних моделей для прогнозування попиту, виробництва та інших аспектів діяльності господарства. За допомогою цих моделей фермери можуть приймати обґрунтовані рішення щодо виробництва та збуту продукції, що дозволяє оптимізувати виробництво та знижувати витрати. Однією з важливих складових

оптимізації управління господарством є впровадження систем моніторингу та контролю за виробничими процесами. За допомогою сучасних датчиків, IoT-технологій та систем аналізу даних фермери можуть отримувати реальний час інформацію про стан об'єктів виробництва та реагувати на зміни вчасно [15].

ШІ – це галузь комп'ютерних наук, що вивчає створення програм та систем, які демонструють інтелект або розумність, подібні до тих, які характерні для людини. Це включає в себе розвиток алгоритмів та моделей, які дозволяють комп'ютерам навчатися з досвіду, робити прогнози, розпізнавати образи та приймати рішення.

В агросекторі ШІ знаходить широке застосування, допомагаючи сільськогосподарським підприємствам та фермерам покращити ефективність та продуктивність своєї діяльності. Завдяки аналізу великої кількості даних про погоду, ґрунт, рослини та інші фактори, системи на основі штучного інтелекту можуть здійснювати прогнозування врожаю, оптимізувати полив та внесення добрив, виявляти хвороби та шкідників, що дозволяє фермерам приймати обґрунтовані рішення та максимізувати врожайність.

Агроіндустрія 4.0 визначається як використання передових технологій, таких як штучний інтелект, інтернет речей, автоматизація та цифрові технології, для оптимізації виробництва та управління сільськогосподарськими підприємствами.

Ця концепція передбачає впровадження інноваційних підходів у всі аспекти агропромислового виробництва, від сівозміни та вирощування до збирання та обробки урожаю.

Це дозволяє підвищити ефективність та продуктивність галузі, зменшити вплив на навколишнє середовище та забезпечити стабільність у виробництві харчових продуктів.

Ключовими аспектами Агроіндустрії 4.0 є автоматизація сільськогосподарських процесів, використання даних для прийняття рішень,

впровадження цифрових технологій у виробництві та збір інформації з великої кількості джерел для аналізу та оптимізації.

Агроіндустрія 4.0 відкриває нові можливості для розвитку сільського господарства та підвищення його конкурентоспроможності на ринку. Вона створює умови для впровадження інновацій та підвищення якості та кількості виробництва сільськогосподарських продуктів.

Аналізуючи концепції штучного інтелекту, IoT та Агроіндустрії 4.0, ми можемо зробити декілька важливих висновків.

По-перше, сучасні технології мають великий потенціал для підвищення продуктивності та ефективності сільського господарства. Впровадження інноваційних рішень, таких як системи моніторингу та управління, дозволяє фермерам оптимізувати виробничі процеси та знижувати витрати.

По-друге, використання аналітики даних та штучного інтелекту дозволяє фермерам робити обґрунтовані рішення щодо вирощування культур, управління ресурсами та прогнозування врожайності. Це сприяє підвищенню прибутковості та стійкості господарства.

По-третє, концепція Агроіндустрії 4.0 відкриває нові можливості для розвитку сільського господарства у цифрову еру. Інтеграція передових технологій дозволяє створювати сучасні та ефективні системи виробництва, що відповідають сучасним вимогам та потребам ринку.

Загалом, можна підкреслити важливість інновацій у сільському господарстві та їх потенціал для подальшого розвитку галузі. Далі в роботі будуть розглянуті конкретні приклади впровадження передових технологій у практиці сільськогосподарського виробництва [16].

РОЗДІЛ 2
ДОСЛІДЖЕННЯ МЕТОДІВ ТА МОДЕЛІ ПРОГНОЗУВАННЯ

2.1. Методи для прогнозування погоди та вологості ґрунту

Методи регресії в контексті прогнозування погоди та вологості ґрунту є важливими інструментами для аналізу та передбачення метеорологічних явищ. Основні методи включають у себе просту лінійну регресію, поліноміальну регресію, логістичну регресію та нелінійну регресію [17].

Проста лінійна регресія – це метод, який передбачає залежність між однією незалежною змінною (наприклад, часом) та однією залежною змінною (наприклад, температурою). Вона базується на припущенні, що ця залежність може бути апроксимована прямою лінією.

Поліноміальна регресія використовується для моделювання залежностей, які не можуть бути апроксимовані простою лінією. Вона дозволяє використовувати поліноми більшого степеню для апроксимації складних залежностей між змінними.

Логістична регресія використовується для прогнозування ймовірності виникнення події на основі даних про певні змінні. Вона особливо корисна для передбачення категоріальних змінних, таких як наявність опадів або тип погоди.

Нелінійна регресія дозволяє моделювати складні залежності між змінними, які не можуть бути виражені за допомогою простих лінійних або поліноміальних функцій. Вона використовує нелінійні функції для апроксимації залежностей між змінними.

У виборі методу регресії для конкретної задачі прогнозування погоди та вологості ґрунту важливо враховувати як характер даних, так і структуру залежностей між ними [18].

Регресійні моделі широко використовуються для прогнозування погоди через їх здатність моделювати залежності між метеорологічними змінними.

Під час застосування регресійних моделей до прогнозування погоди спочатку збираються великі обсяги даних, такі як температура, атмосферний тиск, вологість, вітер та ін., які виміряні в різні моменти часу.

Наступним кроком є аналіз цих даних, включаючи виявлення залежностей між різними метеорологічними змінними. Регресійні моделі дозволяють побудувати математичні вирази, які описують ці залежності. Наприклад, можна використовувати лінійну регресію для передбачення температури на основі часу доби або поліноміальну регресію для моделювання залежності між температурою та вітром.

Після побудови регресійних моделей їх перевіряють на валідаційних даних, щоб переконатися в їхній точності та ефективності. Після цього моделі можуть бути використані для прогнозування погоди на майбутні періоди, використовуючи вхідні дані, такі як поточні метеорологічні умови та прогнози атмосферних змін.

Важливо відзначити, що точність прогнозів погоди залежить від якості та кількості вхідних даних, а також від складності змінних та вибраного методу регресії. Також враховується постійна нестабільність атмосферних умов, що впливає на точність прогнозів [19].

Побудова регресійних моделей для прогнозування вологості ґрунту базується на ключових принципах, спрямованих на врахування фізичних та географічних факторів, що впливають на вологість ґрунту. Деякі з цих принципів та можливостей включають:

1. Вибір вхідних змінних. Для побудови регресійної моделі необхідно вибрати вхідні змінні, які мають відомий вплив на вологість ґрунту. Це можуть бути, наприклад, опади, температура повітря, вологість повітря, тип ґрунту, нахил території тощо.

2. Обробка та аналіз даних. Зібрані дані піддаються обробці та аналізу для виявлення залежностей між вхідними змінними та вологості ґрунту. Цей етап може включати в себе статистичний аналіз, виявлення кореляцій та інші методи обробки даних.

3. Вибір моделі. На основі аналізу даних вибирається тип регресійної моделі, яка найкраще підходить для моделювання залежностей між вхідними змінними та вологості ґрунту. Це може бути лінійна, поліноміальна, нелінійна чи інша форма регресії, в залежності від характеру даних.

4. Навчання та валідація моделі. Обрана регресійна модель навчається на доступних даних, після чого її ефективність перевіряється на незалежних валідаційних даних. Цей етап дозволяє оцінити точність та надійність прогнозів моделі.

5. Оцінка результатів. Результати прогнозів моделі оцінюються з урахуванням їхньої точності та придатності для практичних застосувань. При необхідності модель може бути вдосконалена або перебудована з використанням додаткових даних чи інших методів регресії.

Ці принципи дозволяють побудувати ефективні регресійні моделі для прогнозування вологості ґрунту, що можуть бути використані для вирішення різних завдань у сфері сільського господарства, екології та ін. галузях [20].

2.2. Моделі прогресії для аналізу та оптимізації поливу

Прогресійні моделі в аналізі даних є невід'ємною складовою для прогнозування та моделювання залежностей у часі. Ці моделі дозволяють нам встановити зв'язок між змінними та їхніми тенденціями в часі, що дозволяє нам прогнозувати майбутні значення або зміни на основі наявних даних [21].

Лінійна регресія є однією з найпоширеніших прогресійних моделей. Цей метод полягає у побудові лінійної функції, яка найкращим чином відповідає залежності між незалежною та залежною змінною. Вона дозволяє нам аналізувати, як змінюється залежна змінна при зміні незалежної змінної, і застосовується для прогнозування майбутніх значень на основі цього зв'язку.

Для випадків, коли залежність між змінними не може бути описана лінійною функцією, використовується нелінійна регресія. Цей підхід

дозволяє нам моделювати залежності за допомогою нелінійних функцій, таких як квадратичні, експоненціальні або логарифмічні. Він дозволяє краще враховувати складні тенденції у даних та забезпечує точніші прогнози в разі нелінійних залежностей [22].

Часові ряди – це інший важливий метод прогресії, який використовується для аналізу даних з часовою структурою. Він дозволяє нам прогнозувати майбутні значення на основі попередніх даних та враховує зміни у часі. Цей метод особливо корисний для прогнозування тенденцій та сезонності у даних.

Байєсівська регресія використовує статистичний підхід для моделювання залежностей та розподілів у даних. Вона дозволяє нам враховувати невизначеність у прогнозах та робити більш точні прогнози, враховуючи різноманітність факторів та їхній стохастичний характер.

Нарешті, машинне навчання, таке як метод опорних векторів або нейронні мережі, є потужним інструментом для прогнозування та моделювання складних залежностей у даних. Ці методи зазвичай використовуються для розв'язання складних задач прогнозування, де інші методи можуть бути менш ефективними.

Застосування прогресійних моделей для визначення потреб поливу рослин є ключовим етапом в оптимізації агрокультурного процесу. Ці моделі дозволяють аналізувати та прогнозувати зміни вологості ґрунту в часі, що в свою чергу допомагає визначити оптимальні час та обсяг поливу для забезпечення здоров'я та росту рослин.

На початковому етапі, прогресійні моделі використовуються для аналізу історичних даних про вологості ґрунту, погодних умов, типу ґрунту та інших факторів, які впливають на потреби рослин у воді. За допомогою лінійних чи нелінійних регресійних моделей, можна встановити залежність між цими факторами та вологості ґрунту у минулому, що дозволяє прогнозувати майбутні зміни вологості. Після цього, отримані моделі можуть бути використані для розробки алгоритмів оптимального поливу. Ці алгоритми

будуть враховувати не лише погодні умови та характеристики ґрунту, а й інші фактори, такі як тип рослин, їхні вікові та фізіологічні особливості. Наприклад, за допомогою прогресійних моделей можна визначити оптимальну кількість води, необхідної для різних видів культур у різні періоди їхнього життєвого циклу.

Такий підхід дозволяє створити систему автоматичного поливу, яка буде адаптуватися до змінних умов та потреб рослин. Це забезпечує оптимальні умови для росту та розвитку рослин, зменшуючи водні витрати та підвищуючи ефективність агрокультурного процесу в цілому [23].

1. Розробка алгоритмів оптимального поливу на основі прогресійних моделей - це комплексний процес, що передбачає аналіз та використання даних для визначення оптимальних параметрів поливу для рослин. Основний крок цього процесу включає наступні етапи:

2. Збір та підготовка даних: Перший крок – це збір необхідних даних, таких як вологість ґрунту, погодні умови, тип ґрунту та інші фактори, що можуть впливати на потреби рослин у воді. Потім ці дані піддаються попередній обробці, включаючи очищення від викидів та форматування для подальшого аналізу.

3. Аналіз даних та вибір моделі: Наступний крок - це використання прогресійних моделей, таких як лінійна або нелінійна регресія, для моделювання залежностей між вологості ґрунту та різними факторами. Це дозволяє нам отримати розуміння тенденцій у даних та їх впливу на потреби рослин у воді.

4. Визначення оптимальних параметрів поливу: На основі результатів моделювання визначаються оптимальні параметри поливу для різних умов. Це включає визначення оптимальних часів поливу, тривалості поливу та обсягу води, необхідного для задоволення потреб рослин у воді.

5. Розробка алгоритму поливу: На цьому етапі розробляються алгоритми поливу, які враховують отримані з прогресійних моделей дані та оптимальні параметри поливу. Ці алгоритми можуть бути реалізовані у

вигляді програмного забезпечення для автоматичного керування системами поливу або вбудовані у спеціалізовані пристрої.

6. Тестування та налагодження: Останній крок - це тестування розроблених алгоритмів на реальних даних та налагодження їх для досягнення оптимальної продуктивності. Під час цього етапу може здійснюватися коригування параметрів алгоритму для покращення його ефективності та точності.

Результатом цього процесу є розроблений алгоритм оптимального поливу, який дозволяє автоматично керувати поливом рослин на основі змінних умов та потреб культур. Такий підхід допомагає забезпечити оптимальні умови для росту та розвитку рослин, зменшуючи водні витрати та підвищуючи ефективність сільськогосподарського виробництва [24].

2.3 Модель на основі ШІ

Створення моделі на основі ШІ для потреб агросектору вимагає поєднання кількох ключових компонентів, таких як збирання та обробка даних, вибір відповідних алгоритмів машинного навчання, використання потужних обчислювальних ресурсів та тестування моделі на реальних даних. Розглянемо основні етапи та можливості створення такої моделі.

Першим етапом є збирання даних з різних джерел, таких як датчики вологості ґрунту, погодні станції, супутникові знімки, а також історичні дані про врожайність. Ці дані є основою для навчання моделі. Зібрані дані потрібно очистити та нормалізувати для подальшої обробки. Важливо забезпечити достатній обсяг даних для досягнення високої точності моделі.

Для створення моделі можна використовувати різні алгоритми машинного навчання, такі як регресія, класифікація або кластеризація. Наприклад, для прогнозування врожайності можна використовувати алгоритми лінійної регресії або дерева рішень. Вибір алгоритму залежить від

конкретного завдання та типу даних. Нейронні мережі та методи глибокого навчання можуть бути ефективними для обробки великих обсягів даних та складних завдань.

Першим етапом початку роботи в середовищі є завантаження необхідних бібліотек (рис. 2.1). Спочатку були завантажені необхідні бібліотеки для обробки та аналізу даних, включаючи Numpy, Pandas, TensorFlow.Keras, Matplotlib, і Google.Colab.

```
# Робота з масивами
import numpy as np

# Робота з таблицями
import pandas as pd

# Кодування категоріальних даних в форматі OHE
from tensorflow.keras.utils import to_categorical

# Класи-конструктори моделей нейронних мереж
from tensorflow.keras.models import Sequential, Model

# Основні шари
from tensorflow.keras.layers import concatenate, Input, Dense, Dropout, BatchNormalization, Flatten, Conv1D, Conv2D, LSTM, GlobalMaxPooling1D, MaxPooling1D, RepeatVector

# Оптимізатори
from tensorflow.keras.optimizers import Adam

# Нормувальники
from sklearn.preprocessing import StandardScaler, MinMaxScaler

# Матриця помилок класифікатора
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# Відображення графіків
import matplotlib.pyplot as plt

# Відображення графіки в комірці colab
%matplotlib inline

# Відключення попереджень
import warnings
warnings.filterwarnings('ignore')

# Прив'язка Google Drive
from google.colab import drive
drive.mount('/content/gdrive')
```

Рисунок 2.1 – Завантажені бібліотеки

Після завантаження бібліотек було встановлено стилі та розмір графіків за замовчуванням. Нижче на рис. 2.2 вказаний код з налаштуваннями.

```
# Призначення розміру та стилю графіків за умовчанням
from pylab import rcParams
plt.style.use('ggplot')
rcParams['figure.figsize'] = (14, 7)
```

Рисунок 2.2 – Налаштування відображення

Далі, дані були завантажені з файлу у форматі .txt за допомогою бібліотеки Pandas. При цьому були призначені відповідні імена стовпцям

даних. На рис. 2.3 зображено код для введення даних з локальних файлів через Google Drive.

Завантаження даних

```
# Читання даних у таблицю Pandas та призначення імен стовпців
data = pd.read_csv('/content/gdrive/MyDrive/ai/Content_base_MIN60copy.txt', sep="\t", names=['Datetime', 'Open', 'High', 'Low', 'Close'])

# Налаштування імен стовпців
COL_DATE = 'Datetime'
COL_CLOSE = 'Close'

# Налаштування часового інтервалу для обмеження даних
TIME_RANGE = ('2023-09-01 10:00:00.000', '2024-03-01 23:59:00.000')

# Перетворення стовпця дати на індекс таблиці та видалення непотрібного стовпця дати
data.index = pd.to_datetime(data[COL_DATE])
data.drop(columns=COL_DATE, inplace=True)

# Відбір даних по заданому часовому інтервалу
mask = (data.index >= TIME_RANGE[0]) & (data.index <= TIME_RANGE[1])
data = data[mask]
data
```

Рисунок 2.3 – Завантаження даних з текстового документу

Були налаштовані імена стовпців для зручного доступу до даних, а також встановлено часовий діапазон для обмеження даних відповідно до заданих критеріїв. Стовпець дати був перетворений у індекс таблиці для подальшого зручного використання в аналізі. При цьому стовпець дати був видалений з основної таблиці. Були відібрані дані, які відповідають заданому часовому інтервалу, за допомогою маски. Після введення було виконано перевірку результату працездатності та коректного імпорту даних. На рис. 2.4 показано таблицю, яка є результатом імпорту даних.

Datetime	Open	High	Low	Close
2023-09-01 10:00:00	74.38	74.92	73.60	73.66
2023-09-01 11:00:00	73.65	73.95	73.15	73.34
2023-09-01 12:00:00	73.38	73.87	73.26	73.68
2023-09-01 13:00:00	73.66	73.77	73.30	73.68
2023-09-01 14:00:00	73.65	73.90	73.31	73.50
...	...	...	...	...
2024-03-01 14:00:00	107.17	107.22	106.50	106.80
2024-03-01 15:00:00	106.79	107.40	106.50	107.29
2024-03-01 16:00:00	107.29	108.30	107.15	108.10
2024-03-01 17:00:00	108.12	108.20	107.60	108.03
2024-03-01 18:00:00	108.08	108.26	107.85	108.26

1125 rows × 4 columns

Рисунок 2.4 – Результати завантаження представлений у вигляді таблиці

На завершення, був побудований графік часового ряду вологості ґрунту за допомогою бібліотеки Matplotlib, що дозволило візуалізувати дані та зробити їх аналіз. На рис. 2.5 вказаний код для побудови часового ряду в графічному вигляді.

Ілюстрація даних у графічному вигляді

```
# Відображення часового ряду у графічному вигляді
plt.figure(figsize=(20, 8))
plt.plot(data.index, data[COL_CLOSE])
plt.title('Графік вологості ґрунту (1h)')
plt.show()
```

Рисунок 2.5 – Налаштування відображення часового ряду

Результатом кодування є графік, який показує погодинний ріст або падіння графіку за кожну годин в межах встановленої дати. На рис. 2.6 зображено часовий ряд на основі даних про вологість ґрунту.

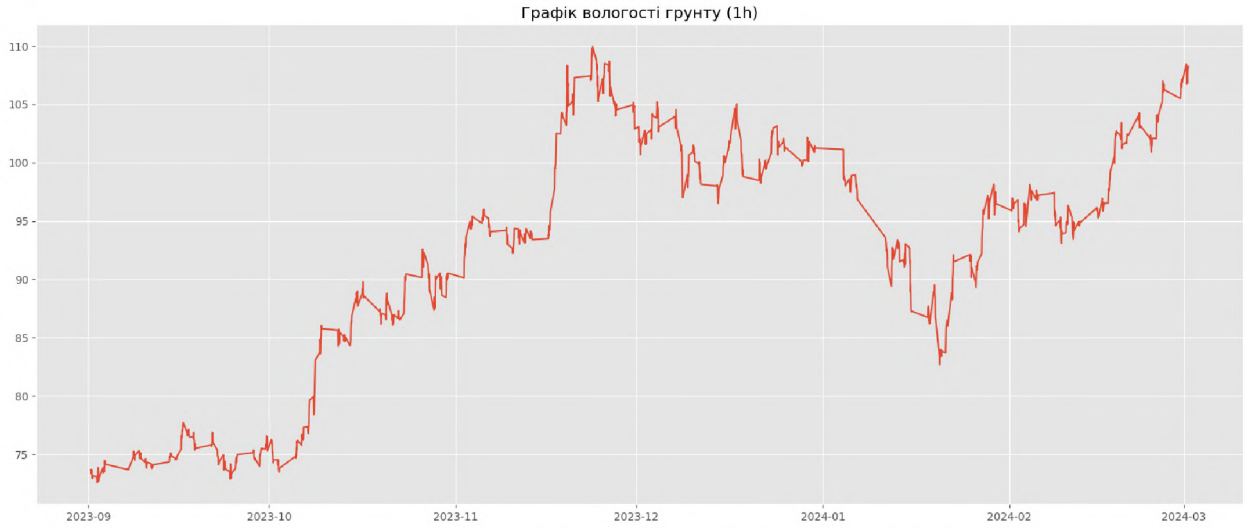

Рисунок 2.6 – Часовий ряд на основі введених даних

Такий часовий ряд допомагає розуміти межі, в яких була підвищена, нормальна або низька вологість ґрунту. На основі цього графіку можна додати дані та інформацію про вологість і закономірність росту вологості

протягом часу до нейронної мережі для подальшого прогнозування, виходячи з різниці сезону та зовнішніх чинників.

Першим кроком у підготовці вибірок даних для нейромережі було визначення ознак, які використовуватимуться для навчання. Одна з ознак була виділена як основна – це значення вологості ґрунту. Крім того, було створено додаткові ознаки, які відображають зміни вологості ґрунту за певний період часу. На рис. 2.7 зображено код для налаштування гіперпараметрів нейронної мережі.

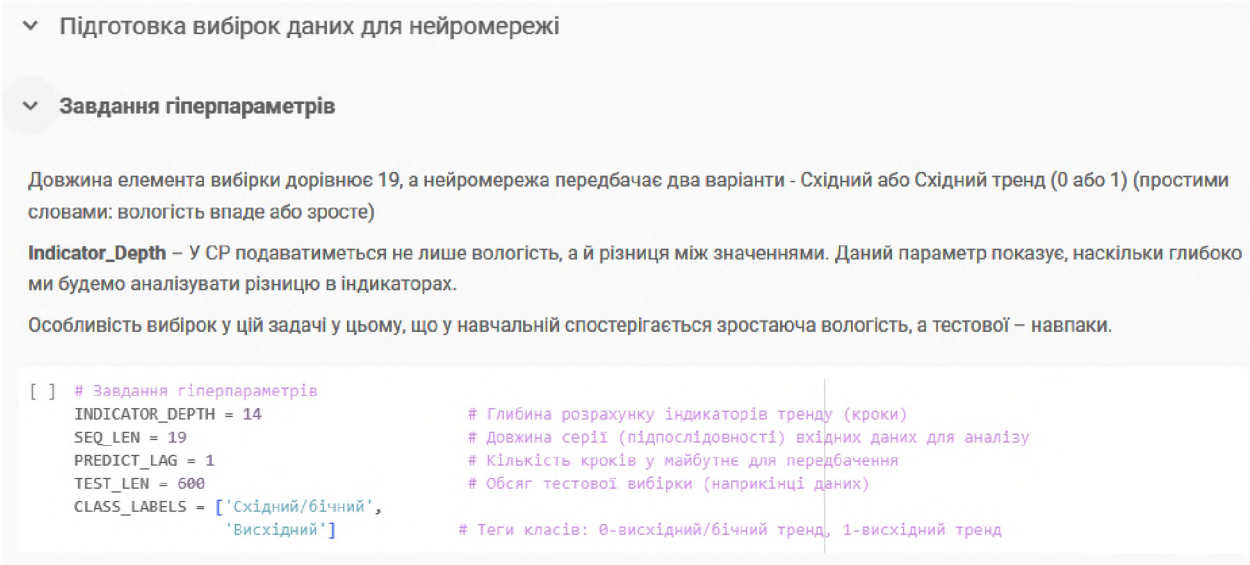

Рисунок 2.7 – Налаштування параметрів для навчання нейромережі

На цьому етапі ми встановлюємо гіперпараметри для підготовки вибірок даних для нейромережі.

Довжина елемента вибірки (SEQ_LEN) визначає кількість зразків, які будуть використані для аналізу одного елемента вхідних даних. У нашому випадку, ця величина дорівнює 19.

Indicator_Depth вказує глибину розрахунку індикаторів тренду. Це означає, що ми аналізуємо різницю між значеннями на певну кількість попередніх кроків, яку встановлюємо на 14.

PREDICT_LAG показує кількість кроків у майбутнє для передбачення. У нашому випадку, ми прогнозуємо значення лише на один крок вперед.

TEST_LEN визначає обсяг тестової вибірки, яка буде використана для оцінки працездатності моделі. Вона складається з останніх 600 зразків у даних.

CLASS_LABELS містить мітки для класів, які передбачаємо моделлю. У нашому випадку, це «Східний/бічний» та «Висхідний».

Наступним етапом йде розрахунок ознак тренду. На рис. 2.8 вказаний код для розрахунку індикаторів відносної необхідності поливу.

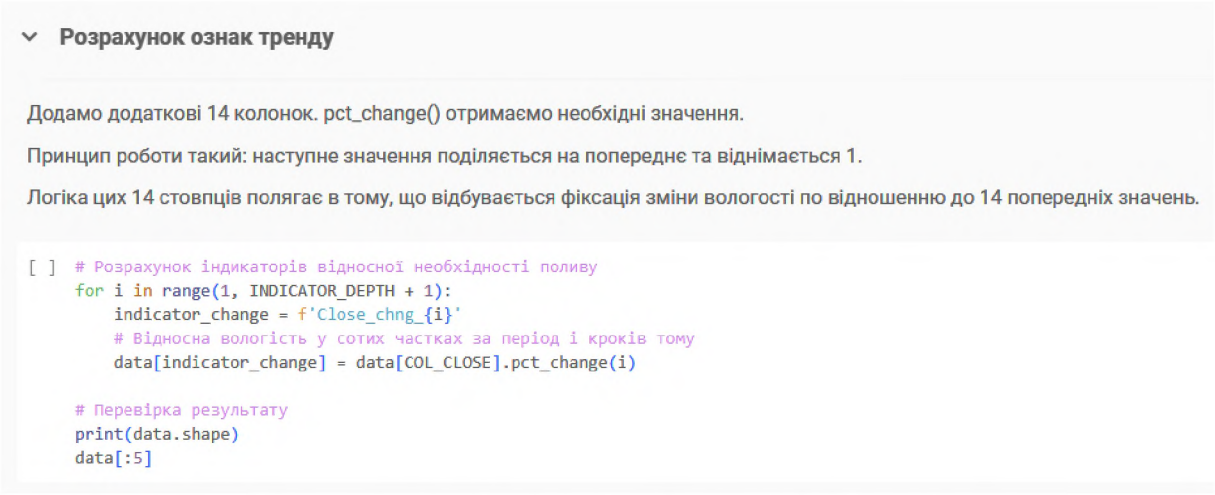

Рисунок 2.8 – Розрахунок індикаторів поливу

На цьому етапі ми обчислюємо ознаки тренду, які допоможуть нам у подальшому аналізі та передбаченні. Для цього додаємо 14 додаткових стовпців до наших даних.

Ми використовуємо метод pct_change(), щоб обчислити різницю вологості між поточним та попереднім значеннями для кожного з 14 попередніх кроків. Це дає нам інформацію про те, як вологість змінюється відносно попередніх значень на різних відстанях у часі. Наприклад, якщо вибрати i = 1, ми отримаємо різницю вологості між поточним та попереднім кроком. Якщо вибрати i = 2, отримаємо різницю вологості між поточним та значенням, яке було два кроки тому, і так далі.

Це дозволяє нам виявити закономірності у зміні вологості та використовувати їх для передбачення майбутніх значень. Після обчислення

ми перевіряємо форму даних та виводимо перші п'ять рядків, щоб переконатися, що обчислення пройшло успішно. На рис. 2.9, зображено результати розрахунків у вигляді таблиці з даними.

(1125, 18)

Datetime	Open	High	Low	Close	Close_chng_1	Close_chng_2	Close_chng_3	Close_chng_4	Close_chng_5	Close_chng_6	Close_chng_7	Close_chng_8	Close_chng_9	Close_chng_10	Close_chng_11	Close_chng_12	Close_chng_13	Close_chng_14
2023-09-01 10:00:00	74.38	74.92	73.60	73.66	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2023-09-01 11:00:00	73.65	73.95	73.15	73.34	-0.004344	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2023-09-01 12:00:00	73.38	73.87	73.26	73.68	0.004636	0.000272	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2023-09-01 13:00:00	73.66	73.77	73.30	73.68	0.000000	0.004636	0.000272	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2023-09-01 14:00:00	73.65	73.90	73.31	73.50	-0.002443	-0.002443	0.002182	-0.002172	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Рисунок 2.9 – Виведення результату у вигляді таблиці

Після обчислення ознак тренду можуть виникнути рядки з пропущеними значеннями (NaN). У даному випадку, перші 13 рядків містять такі пропущені значення.

Ми використовуємо метод dropna() для видалення рядків з неповними даними (рис. 2.10). Це допомагає зберегти лише ті рядки, де всі значення заповнені.

Видалимо рядки, в яких є **NaN** (це будуть перші 13 рядків):

```
[ ]  # Видалення рядків із неповними даними
     data.dropna(inplace=True)

     # Перевірка результату
     print(data.shape)
     data[:5]
```

Рисунок 2.10 – Редагування комірок з неповними даними в таблиці

Після видалення ми перевіряємо форму даних та виводимо перші п'ять рядків, щоб переконатися, що пропущені значення були успішно видалені. На рис. 2.11 зображено перевірку результату видалення.

(1111, 18)

Datetime	Open	High	Low	Close	Close_chng_1	Close_chng_2	Close_chng_3	Close_chng_4	Close_chng_5	Close_chng_6	Close_chng_7	Close_chng_8	Close_chng_9	Close_chng_10	Close_chng_11	Close_chng_12	Close_chng_13	Close_chng_14
2023-09-02 15:00:00	73.27	73.98	73.01	73.86	0.008328	0.016935	0.015816	0.017636	0.010812	0.009844	0.012474	0.010950	0.012891	0.004898	0.002443	0.002443	0.007090	0.002715
2023-09-02 16:00:00	73.88	74.18	73.15	73.40	-0.006228	0.002048	0.010602	0.009490	0.011296	0.004516	0.003555	0.006169	0.004654	0.006583	-0.001361	-0.003600	-0.003800	0.000818
2023-09-02 17:00:00	73.36	73.70	72.65	72.72	-0.009264	-0.015435	-0.007235	0.001239	0.000138	0.001929	-0.004790	-0.005742	-0.003153	-0.004654	-0.002743	-0.010612	-0.013029	-0.013029
2023-09-02 18:00:00	72.72	73.11	72.69	72.90	0.002475	-0.006812	-0.012998	-0.004778	0.003717	0.002613	0.004409	-0.002327	-0.003281	-0.000685	-0.002190	-0.000274	-0.008163	-0.010586
2023-09-03 10:00:00	73.50	73.97	73.41	73.65	0.013032	0.015539	0.006131	-0.000135	0.008191	0.016797	0.015679	0.017498	0.010675	0.009707	0.012337	0.010613	0.012754	0.004762

Рисунок 2.11 – Результати після введення коректування

Створення моделі ШІ вимагає значних обчислювальних ресурсів. Використання хмарних платформ, таких як Google Colab, дозволяє отримати доступ до потужних графічних процесорів (GPU) для прискорення процесу навчання моделі. Це особливо важливо при роботі з великими даними та складними моделями, що вимагають великих обчислювальних потужностей.

Після вибору алгоритму та збору даних модель потрібно навчити. Процес навчання включає подачу даних на вхід моделі та коригування її параметрів для мінімізації похибки прогнозів. Після початкового навчання модель можна оптимізувати за допомогою методів перехресної перевірки та підбору гіперпараметрів. Важливо перевіряти модель на валідаційних наборах даних для запобігання перенавчанню.

Останнім етапом є тестування моделі на реальних даних, які не використовувалися під час навчання. Це дозволяє оцінити точність та надійність моделі у реальних умовах. Модель можна оцінювати за різними метриками, такими як середньоквадратична похибка (MSE), середня абсолютна похибка (MAE) або коефіцієнт детермінації (R^2).

Після успішного тестування модель можна впроваджувати у практичні умови сільського господарства. Наприклад, модель може бути інтегрована в систему автоматизованого поливу для оптимального використання водних ресурсів або використовуватися для прогнозування врожайності та планування сільськогосподарських робіт.

РОЗДІЛ 3
ПРАКТИЧНА РЕАЛІЗАЦІЯ СИСТЕМИ ІНТЕЛЕКТУАЛЬНОГО ПОЛИВУ

3.1. Розробка та навчання нейронної мережі

Після обчислення ознак тренду можемо сформувати вибірку для навчання моделі у два можливих способи.

1. «Подавати дані як …» – цей підхід передбачає подання ознак тренду в їхньому поточному числовому вигляді. Кожен стовпчик з ознаками тренду буде мати числові значення, які відображають різницю вологості на певному віддаленому кроці часу.

2. Заміна значень. У такому випадку, можемо замінити числові значення ознак тренду на 0, якщо різниця вологості була від'ємною, і на 1, якщо вона була позитивною. Це дозволяє перетворити числові ознаки тренду на категоріальні, які можуть бути легше інтерпретовані та використані моделлю.

У даному коді використовуємо другий підхід. Для кожного періоду тренду ми створюємо стовпець індикатора тренду (indicator_trend), який заповнюється значеннями 0 для низхідного/бічного тренду та 1 для висхідного тренду (рис. 3.1).

Можливі два варіанти формування вибірки: подавати дані як є, або замінити значення на 0 де значення негативне і на 1 де позитивне.

```
# Обчислення індикаторів напряму тренду
for i in range(1, INDICATOR_DEPTH + 1):
    indicator_change = f'Close_chng_{i}'
    indicator_trend = f'Y_{i}'
    # Заповнення стовпця індикатора тренда нулями (низхідний/бічний тренд)
    data[indicator_trend] = 0.
    # Якщо індикатор зміни періоду на і кроків позитивний, то висхідний тренд
    data.loc[data[indicator_change] > 0. , indicator_trend] = 1.

# Перевірка результату
print(data.shape)
data[:5]
```

Рисунок 3.1 – Обчислення індикаторів напряму тренду

Для цього ми перевіряємо, чи є значення ознаки зміни позитивним, і призначаємо відповідне значення індикатору тренду. Після цього ми перевіряємо форму даних та виводимо перші п'ять рядків, щоб переконатися, що дані були успішно сформовані. На рис. 3.2 зображено результат сформованих даних.

(1111, 32)

Datetime	Open	High	Low	Close	Close_chng_1	Close_chng_2	Close_chng_3	Close_chng_4	Close_chng_5	Close_chng_6	...	Y_5	Y_6	Y_7	Y_8	Y_9	Y_10	Y_11	Y_12	Y_13	Y_14
2023-09-02 15:00:00	73.27	73.98	73.01	73.86	0.008328	0.016935	0.015816	0.017636	0.010812	0.009844	...	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
2023-09-02 16:00:00	73.88	74.18	73.15	73.40	-0.006228	0.002048	0.010602	0.009490	0.011298	0.004516	...	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
2023-09-02 17:00:00	73.36	73.70	72.65	72.72	-0.009264	-0.015435	-0.007235	0.001239	0.000138	0.001929	...	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2023-09-02 18:00:00	72.72	73.11	72.69	72.90	0.002475	-0.006812	-0.012998	-0.004778	0.003717	0.002613	...	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2023-09-03 10:00:00	73.50	73.97	73.41	73.85	0.013032	0.015539	0.006131	-0.000135	0.008191	0.016797	...	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

5 rows x 32 columns

Рисунок 3.2 – Результати формування вибірок

Наступним етапом ми формуємо вхідні дані для нашої моделі. Вхідні дані складаються зі значень закриття (COL_CLOSE) та індикаторів тренду (Y_i), де i від 1 до INDICATOR_DEPTH. Формуємо список колонок x_col, який містить назви стовпців для закриття та індикаторів тренду.

Потім ми створюємо матрицю вхідних даних x_data, витягуючи значення цих стовпців з нашого DataFrame. Використовуємо .values, щоб отримати числове представлення даних.

Після формування даних перевіряємо їх розмірність та тип, щоб впевнитися, що вони відповідають нашим очікуванням. На рис. 3.3 зображено код та результат перевірки.

Формування вхідних даних

```
[ ]  x_col = [COL_CLOSE] + [f'Y_{i}' for i in range(1, INDICATOR_DEPTH + 1)]
     x_data = data[x_col].values

     # Перевірка результату
     print(x_data.shape, type(x_data))
```

```
(1111, 15) <class 'numpy.ndarray'>
```

Рисунок 3.3 – Формування вхідних даних

Далі формуємо вихідні дані для нашої моделі. Вихідні дані представляють собою класифікаційну ознаку, яка вказує на тенденцію тренду, тобто чи буде вологість зростати (висхідний тренд) або зменшуватися (низхідний або бічний тренд).

Після цього ми перевіряємо перші п'ять значень вихідних даних, щоб побачити, чи відповідають вони нашим очікуванням. На рис. 3.4 зображено результати формування вихідних даних.

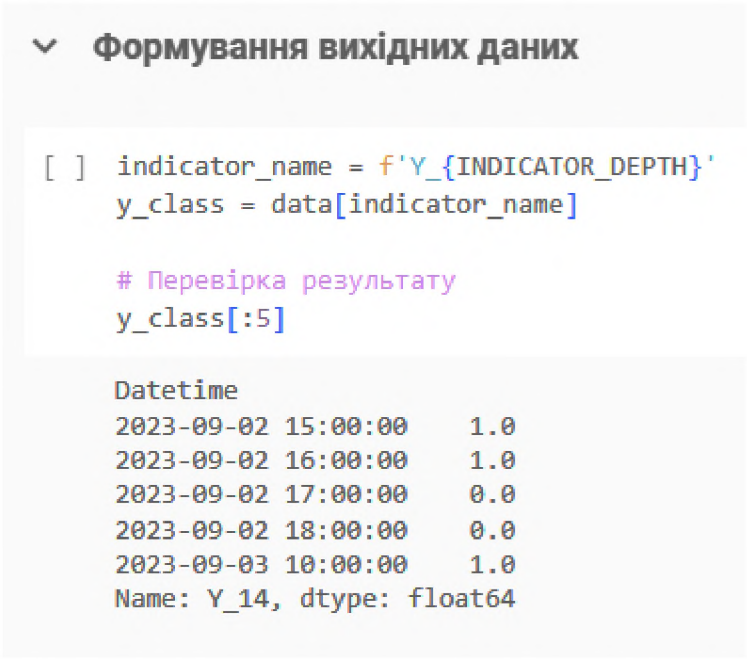

Рисунок 3.4 – Формування вихідних даних

Ми використовуємо назву стовпця indicator_name, щоб визначити стовпець, який містить класифікаційну ознаку. Потім отримуємо відповідний стовпець з даних та присвоюємо його змінній y_class.

Далі було використано метод one-hot encoding для перетворення категоріальної ознаки «Напрямок зміни вологості ґрунту на числовий вектор, що складається з двох елементів». Перетворюємо категоріальну ознаку (клас тренду) в формат one-hot encoding. Це необхідно для того, щоб модель могла правильно інтерпретувати ці дані.

Метод to_categorical з бібліотеки Keras виконує цю операцію, конвертуючи цільовий вектор у бінарну матрицю, де кожна колонка

відповідає одному класу, а значення 1 в певній колонці позначає належність елемента до цього класу.

Після кодування ми перевіряємо розмір отриманої матриці та виводимо перші п'ять рядків для перевірки правильності операції. На рис. 3.5 зображено код та результати кодування.

```
[ ]  # Кодування ознаки класу в one hot encoding
     y_data = to_categorical(y_class)

     # Перевірка результату
     print(y_data.shape)
     y_data[:5]
```

```
(1111, 2)
array([[0., 1.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [0., 1.]], dtype=float32)
```

Рисунок 3.5 - Кодування класу в one-hot encoding

Перший елемент відповідає класу «Східний/бічний», а другий – «Висхідний». Цей підхід дозволяє моделі ефективно працювати з цільовою ознакою, враховуючи її категоріальний характер.

Після цього дані було розділено на навчальну та перевірочну вибірки для оцінки продуктивності моделі на нових даних. Навчальна вибірка використовувалася для навчання моделі, тоді як перевірочна – для перевірки її продуктивності.

Ми використовуємо останні TEST_LEN точок даних для тестування, і всі інші - для навчання. При цьому ми також враховуємо SEQ_LEN, оскільки для кожної точки даних у вибірці нам необхідно мати попередні SEQ_LEN точок для аналізу. Після поділу даних ми перевіряємо розмір отриманих вибірок та загальну довжину, щоб переконатись, що поділ виконано коректно. На рис. 3.6 код та результати кодування.

Поділ даних на навчальну та перевірочну вибірки

```
[ ]  x_train_data, y_train_data = x_data[:-TEST_LEN-SEQ_LEN], y_data[:-TEST_LEN-SEQ_LEN]
     x_test_data, y_test_data = x_data[-TEST_LEN:], y_data[-TEST_LEN:]

     # Перевірка результату
     print('train_data: ', x_train_data.shape, y_train_data.shape,
           '\ntest_data:  ', x_test_data.shape, y_test_data.shape,
           '\nЗагальна довжина:', x_train_data.shape[0] + x_test_data.shape[0])
```

```
train_data:  (492, 15) (492, 2)
test_data:   (600, 15) (600, 2)
Загальна довжина: 1092
```

Рисунок 3.6 – Поділ даних на навчальну та перевірну вибірки

Для нормалізації вхідних даних було застосовано метод масштабування. Це дозволяє привести всі ознаки до одного масштабу і сприяє кращому навчанню моделі.

Тут ми застосовуємо масштабування до вхідних даних за допомогою методу StandardScaler з бібліотеки scikit-learn. Це стандартизує кожен ознаку так, що їх середнє значення буде 0, а стандартне відхилення – 1.

Ми обчислюємо параметри масштабування (середнє значення та стандартне відхилення) лише на навчальних даних, а потім застосовуємо ці параметри до навчальних та тестових даних. Це допомагає уникнути витоку інформації з тестової вибірки в процесі навчання моделі.

Після масштабування ми перевіряємо розмір та тип отриманих даних для перевірки коректності операції. На рис. 3.7 зображено код та результат виконання масштабування вхідних даних.

Масштабування вхідних даних

```
[ ]  x_scaler = StandardScaler()
     x_scaler.fit(x_train_data)
     x_train_data = x_scaler.transform(x_train_data)
     x_test_data = x_scaler.transform(x_test_data)

     # Перевірка результату
     print(x_train_data.shape, x_test_data.shape, type(x_train_data))
     x_test_data[-1:]
```

```
(492, 15) (600, 15) <class 'numpy.ndarray'>
array([[ 2.66857276,  0.95624102,  0.91798509,  0.9067647 ,  0.87737345,
         0.87375103,  0.84515425,  0.84515425, -1.23413003,  0.78978629,
         0.76288503,  0.77627692,  0.77627692,  0.76622241,  0.74629888]])
```

Рисунок 3.7 – Масштабування вхідних даних

Далі необхідно поділити набори даних на вибірки для навчання нейронної мережі. На рис. 3.8 зображено код формування вибірок для нейронної мережі.

Формування вибірок для нейромережі

```
[ ]  # Функція поділу набору даних на вибірки для навчання нейромережі
     # x_data - набір вхідних даних
     # y_data - набір вихідних даних
     # seq_len – довжина серії (підпослідовності) вхідних даних для аналізу
     # predict_lag - кількість кроків у майбутнє для передбачення
     def split_sequence(x_data, y_data, seq_len, predict_lag):
         # Визначення максимального індексу початку підпослідовності
         x_len = x_data.shape[0] - seq_len - (predict_lag - 1)
         # Формування підпослідовностей вхідних даних
         x = [x_data[i:i + seq_len] for i in range(x_len)]
         # Формування міток вихідних даних,
         # віддалених на predict_lag кроків після кінця підпослідовності
         y = [y_data[i + seq_len + predict_lag - 1] for i in range(x_len)]
         # Повернення результатів у вигляді масивів numpy
         return np.array(x), np.array(y)
```

Рисунок 3.8 – Формування вибірок для нейронної мережі

Тут ми визначаємо функцію split_sequence, яка допомагає розділити набір даних на підпослідовності для навчання нейромережі. Функція приймає на вхід набір вхідних даних x_data, набір вихідних даних y_data, довжину серії seq_len та кількість кроків у майбутнє для передбачення predict_lag.

Вона обчислює максимальний індекс, по якому можна сформувати підпослідовності з вхідних даних. Потім вона формує підпослідовності для вхідних даних та відповідні мітки для вихідних даних. Функція повертає результати у вигляді масивів numpy.

Далі ми використовуємо раніше визначену функцію split_sequence для розділення навчальних та тестових даних на підпослідовності, які будуть використовуватися для навчання та перевірки нейромережі. На рис. 3.9 зображено код. Результати перевіряються, і нам надаються форми цих масивів.

```
[ ]  # Формування вибірок для навчання нейромережі
     x_train, y_train = split_sequence(x_train_data, y_train_data, SEQ_LEN, PREDICT_LAG)
     x_test, y_test = split_sequence(x_test_data, y_test_data, SEQ_LEN, PREDICT_LAG)

     # Перевірка результату
     print('Масиви x_train та y_train:\t', x_train.shape, y_train.shape)
     print('Масиви x_test та y_test:\t', x_test.shape, y_test.shape)

     Масиви x_train та y_train:      (473, 19, 15) (473, 2)
     Масиви x_test та y_test:        (581, 19, 15) (581, 2)
```

Рисунок 3.9 – Формування вибірок для навчання та результати перевірки

Ми подаємо вхідні дані x_train_data та x_test_data, вихідні дані y_train_data та y_test_data, а також значення SEQ_LEN (довжина серії) та PREDICT_LAG (кількість кроків у майбутнє для передбачення) для розділення даних.

Після цього ми отримуємо дві окремі вибірки для навчання (x_train та y_train) та перевірки (x_test та y_test) нейромережі.

Наступним кроком було створення матриці помилок для подальшого аналізу нейромережі. На рис. 3.10 зображено код для відображення матриці помилок.

Ця функція eval_model відображує матрицю помилок для заданої моделі та вхідних даних. Матриця помилок допомагає визначити ефективність моделі, показуючи співвідношення між фактичними та передбаченими класами.

Параметри функції включають:

– model: навчена модель для оцінки;

– x: вхідні дані для оцінки;

– y_true: фактичні вихідні дані;

– class_labels: список міток класів (простір імен класів);

– cm_round: кількість знаків після коми для округлення значень у матриці помилок;

– title: заголовок для візуалізації;

– figsize: розмір фігури для відображення матриці помилок.

```
[ ]  # Функція виведення результатів оцінки моделі на задані дані
     def eval_model(model, x, y_true,
                    class_labels=[],
                    cm_round=3,
                    title='',
                    figsize=(8, 8)):
         # Обчислення передбачення мережі
         y_pred = model.predict(x)
         # Побудова матриці помилок
         cm = confusion_matrix(np.argmax(y_true, axis=1),
                               np.argmax(y_pred, axis=1),
                               normalize='true')
         # Округлення значень матриці помилок
         cm = np.around(cm, cm_round)

         # Завдання розміру шрифту за промовчанням для тексту у клітинках матриці
         old_font_size = rcParams['font.size']
         rcParams['font.size'] = 30

         # Відображення матриці помилок
         fig, ax = plt.subplots(figsize=figsize)
         ax.set_title(f'Нейромережа {title}: матриця помилок нормалізована', fontsize=18)
         disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=class_labels)
         disp.plot(ax=ax, cmap='cividis')
         ax.images[-1].colorbar.remove()       # Стирання непотрібної колірної шкали
         plt.xticks(fontsize=12)
         plt.xlabel('Передбачені класи', fontsize=16)
         plt.yticks(fontsize=12)
         plt.ylabel('Вірні класи', fontsize=16)
         plt.show()

         # Відновлення розміру шрифту за замовчуванням
         rcParams['font.size'] = old_font_size
```

Рисунок 3.10 – Формування матриці помилок

Функція обчислює передбачення моделі для вхідних даних x, будує матрицю помилок, нормалізовану до 1, і відображує її графічно.

Наступний блок коду, зображеного на рис. 3.11, призначений для навчання моделі та відображення прогресу навчання та оцінки результатів.

Ця функція train_eval_model призначена для навчання моделі та відображення прогресу навчання та оцінки результатів.

Параметри функції включають:

- model: модель, яку потрібно навчити та оцінити;
- x_train, y_train: навчальні дані;
- x_test, y_test: дані для перевірки моделі;

– epoch_list: список кортежів, де кожний кортеж містить кількість епох та оптимізатор для кожної фази навчання;

– title: заголовок для візуалізації результатів.

```
# Функція навчання моделі та малювання прогресу та оцінки результатів
# На вході - модель, генератори навчальної та перевірочної вибірок
# і перелік епох як ((epochs1, opt1), (epochs2, opt2), ...].
# Модель послідовно навчається протягом epochs1 епох із оптимізатором opt1 і т.д.
def train_eval_model(model, x_train, y_train, epoch_list,
                     x_test, y_test, title=''):
    # Відображення зведення моделі
    model.summary()

    # Навчання моделі у декілька фаз відповідно до списку epoch_list
    for epochs, opt in epoch_list:
        # Компіляція моделі
        model.compile(loss='binary_crossentropy',
                      metrics=['accuracy'],
                      optimizer=opt)

        # Фаза навчання моделі
        print(f'Навчання {epochs} епох')
        history = model.fit(x_train, y_train,
                            epochs=epochs,
                            validation_data=(x_test, y_test),
                            verbose=1)
```

Рисунок 3.11 – Навчання та формування графіків нейронної мережі

Функція послідовно навчає модель протягом кількох фаз, використовуючи різні набори параметрів епох та оптимізаторів. Після кожної фази навчання вона відображає графіки прогресу навчання (рис. 3.12) (помилка та точність на навчальному та перевірочному наборах) і матрицю помилок для оцінки результату роботи моделі.

Цей блок коду відповідає за малювання графіків прогресу навчання після кожної фази навчання моделі.

Перший графік показує зміну помилки на навчальному та перевірочному наборах даних протягом кожної епохи. Помилка відображається на вісі Y, а номер епохи – на вісі X.

Другий графік показує точність класифікації моделі на навчальному та перевірочному наборах даних протягом кожної епохи. Точність також відображається на вісі Y, а номер епохи – на вісі X.

```
# Малювання графіків минулої фази навчання
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 6), sharex=True)
ax1.set_title(f'{title}: Графік прогресу навчання')
ax1.plot(history.history['loss'], label='Помилка на навчальному наборі')
ax1.plot(history.history['val_loss'], label='Помилка на наборі перевірок')
ax1.set_ylabel('Середня помилка')
ax1.legend()

ax2.plot(history.history['accuracy'], label='Точність на навчальному наборі')
ax2.plot(history.history['val_accuracy'], label='Точність на перевірочному наборі')
ax2.set_ylabel('Точність класифікації')
ax2.legend()

# Вказівка показувати тільки цілі позначки шкали осі x
fig.gca().xaxis.get_major_locator().set_params(integer=True)
plt.xlabel('Епоха навчання')
plt.show()

# Малювання графіків оцінки результату роботи моделі після фази навчання
eval_model(model, x_test, y_test, title=title)
```

Рисунок 3.12 – Задання функції для графіків оцінки результату роботи

Графіки розділені за допомогою параметра sharex=True, щоб обидва вони мали спільну вісь X, що спрощує порівняння значень помилки та точності. Назва кожного графіка містить заголовок, який включається у параметр title. На рис.3.13 представлена модель нейронної мережі. Цей блок коду представляє створення моделі нейронної мережі. Основні кроки описані мають наступний зміст.

1. Вхідний шар: Визначаємо вхідний шар, який має форму, що відповідає формі вхідних даних x_train.

2. Перетворення на одномірний вектор: Використовується шар Flatten, який перетворює вхідні дані у одномірний вектор.

3. Повторення вектору кілька разів: Використовується шар RepeatVector, який копіює вхідний вектор декілька разів. Це допомагає розширити представлення вхідних даних.

```
# Вхідний шар
input = Input(shape=x_train.shape[1:])
# Перетворення на одномірний вектор
x = Flatten () (input)
# Повторення вектора кілька разів
x = RepeatVector(4)(x)
# Одновимірна згортка
x = Conv1D(SEQ_LEN * 2, 5, padding='same', activation='relu')(x)
# Макспулінг та зниження розмірності
x = MaxPooling1D(pool_size=2)(x)
# Перетворення на одномірний вектор
x = Flatten()(x)
# Повторення вектора кілька разів
x = RepeatVector(4)(x)
# Одновимірна згортка
x = Conv1D(SEQ_LEN, 5, padding='same', activation='relu')(x)
# Макспулінг та зниження розмірності
x = MaxPooling1D(pool_size=2)(x)
# Перетворення на одномірний вектор
x = Flatten()(x)
# Повнозв'язковий шар
x = Dense (SEQ_LEN * 100, activation = 'relu') (x)
# Шар регуляризації
x = Dropout(0.4)(x)
# Фінальний шар класифікатора
x = Dense(y_train.shape[1], activation='sigmoid')(x)

# Складання моделі з входу та виходу
model = Model(input, x)
```

Рисунок 3.13 – Модель нейронної мережі

4. Одновимірна згортка: Застосовується шар Conv1D, який використовує згортку одновимірного ядра для виконання операцій згортки над вхідними даними.

5. Макспулінг та зниження розмірності: Використовується шар MaxPooling1D для виконання операції макспулінгу та зменшення розмірності вихідних даних.

6. Повторення вектора кілька разів: Ще один шар RepeatVector для розширення представлення даних.

7. Одновимірна згортка: Ще один шар Conv1D для подальшого виконання операцій згортки над вхідними даними.

8. Макспулінг та зниження розмірності: Знову використовується шар MaxPooling1D для зменшення розмірності даних.

9. Повно-зв'язковий шар: Використовується шар Dense з функцією активації ReLU для створення повно-зв'язкового шару.

10. Шар регуляризації: Застосовується шар Dropout з коефіцієнтом викидання 0.4 для регуляризації моделі та запобігання перенавчанню.

11. Фінальний шар класифікатора: Використовується повно-зв'язковий шар з функцією активації sigmoid для виведення остаточного результату.

12. Складання моделі: Модель складається з вхідного та вихідного шарів за допомогою класу Model.

Наступний блок коду, зображеного на рис. 3.14, навчання нейронної мережі.

```
# Навчання та виведення результатів роботи моделі
train_eval_model(model, x_train, y_train, [(6, Adam(learning_rate=1e-04))],
                 x_test, y_test, title='Передбачення тренду')
```

Рисунок 3.14 – Навчання нейронної мережі

Цей код викликає функцію train_eval_model, яка навчає модель за допомогою переданих навчальних даних x_train та y_train, використовуючи оптимізатор Adam з швидкістю навчання 1e-4. Модель навчається протягом 6 епох.

3.2. Імплементація системи мультимодального прогнозування

Мультимодальне прогнозування є підходом у сфері аналітики даних, що використовує комбінацію кількох типів даних для побудови точних та комплексних прогнозних моделей. Цей метод стає особливо важливим у галузі сільського господарства, де різноманітні фактори, такі як погодні умови, характеристики ґрунту, історичні врожаї, та супутникові знімки, впливають на вирощування культур. Мультимодальне прогнозування дозволяє інтегрувати ці різнорідні дані, забезпечуючи більш повне розуміння складних взаємозв'язків між ними.

Основна ідея мультимодального прогнозування полягає в тому, що кожен тип даних може надати унікальну інформацію, яка доповнює інші джерела. Наприклад, погодні дані можуть дати інформацію про температуру та опади, тоді як супутникові знімки можуть забезпечити візуальне уявлення про стан посівів та індекси вегетації. Об'єднання цих даних дозволяє створювати моделі, які можуть більш точно прогнозувати врожайність, потребу в поливі, ризики захворювань рослин та інші ключові показники.

Одним з ключових аспектів мультимодального прогнозування є необхідність уніфікації даних з різних джерел. Це може включати нормалізацію даних, щоб вони були сумісні один з одним, а також використання методів машинного навчання для обробки великих обсягів даних. Алгоритми машинного навчання, такі як глибокі нейронні мережі, можуть бути особливо ефективними в аналізі складних мультимодальних даних, оскільки вони здатні виявляти приховані патерни та взаємозв'язки.

Мультимодальне прогнозування також вимагає використання потужних обчислювальних ресурсів, особливо при роботі з великими наборами даних та складними моделями. Хмарні платформи, такі як Google Colab, Amazon Web Services (AWS) та Microsoft Azure, надають необхідні обчислювальні потужності та інструменти для розробки та навчання моделей. Використання графічних процесорів (GPU) може значно прискорити процес навчання, що

особливо важливо для глибокого навчання. Практичні застосування мультимодального прогнозування в агросекторі включають прогнозування врожайності, оптимізацію використання водних ресурсів, виявлення та попередження захворювань рослин, а також планування агротехнічних заходів. Наприклад, прогнозування врожайності може допомогти фермерам планувати майбутні посівні кампанії та розподіляти ресурси більш ефективно. Оптимізація поливу на основі мультимодальних прогнозів може знизити витрати води та підвищити врожайність, що має важливе значення в умовах кліматичних змін.

Крім того, мультимодальне прогнозування дозволяє створювати системи підтримки прийняття рішень, які допомагають фермерам оперативно реагувати на зміни умов та робити більш обґрунтовані рішення. Це підвищує стійкість та ефективність аграрного виробництва, знижує ризики та покращує загальну продуктивність.

Загалом, мультимодальне прогнозування є потужним інструментом, який значно покращує можливості аграріїв у управлінні виробничими процесами. Інтеграція різних типів даних та використання передових алгоритмів машинного навчання дозволяє досягти нових висот у точності та ефективності прогнозування, що є ключовим фактором для сталого розвитку сільського господарства в умовах сучасних викликів [25].

За допомогою даної нейронної мережі можна реалізувати систему мультимодального прогнозування, яка здатна передбачати не лише напрям зміни вологості ґрунту, але й інші параметри погоди або клімату. Для цього можна розширити вхідні дані моделі, додавши інші параметри, такі як температура, атмосферний тиск, опади тощо.

Далі, з урахуванням доданих параметрів, модель може бути адаптована до мультимодального прогнозування, де вона буде здатна одночасно передбачати кілька параметрів погоди або клімату. Це дозволить створити більш повну та точну систему прогнозування, яка буде корисною для різних сфер, включаючи сільське господарство, енергетику, транспорт та інші.

Додатково, з урахуванням можливостей нейромережі, систему можна постійно покращувати та адаптувати до змін у вхідних даних та умовах навколишнього середовища. Це дозволить забезпечити більш точні та надійні прогнози для користувачів системи.

3.3. Рекомендації щодо практичної реалізації системи інтелектуального поливу

Після завершення кожної фази навчання, функція виводить графіки, які показують прогрес навчання моделі, включаючи середню помилку на навчальному та перевірочному наборах даних, а також точність класифікації на цих наборах. Крім того, вона викликає функцію eval_model, яка відображає матрицю помилок нормалізовану для оцінки результатів роботи моделі після кожної фази навчання. Тривалість навчання може варіюватися в залежності від обсягу та складності даних, а також параметрів навчання, таких як кількість епох. На рис. 3.15 - 3.18 наведені результати перевірки навчання нейронної мережі.

Model: "model_4"

Layer (type)	Output Shape	Param #
input_5 (InputLayer)	[(None, 19, 15)]	0
flatten_12 (Flatten)	(None, 285)	0
repeat_vector_8 (RepeatVector)	(None, 4, 285)	0
conv1d_8 (Conv1D)	(None, 4, 38)	54188
max_pooling1d_8 (MaxPooling1D)	(None, 2, 38)	0
flatten_13 (Flatten)	(None, 76)	0
repeat_vector_9 (RepeatVector)	(None, 4, 76)	0
conv1d_9 (Conv1D)	(None, 4, 19)	7239
max_pooling1d_9 (MaxPooling1D)	(None, 2, 19)	0
flatten_14 (Flatten)	(None, 38)	0
dense_8 (Dense)	(None, 1900)	74100
dropout_4 (Dropout)	(None, 1900)	0
dense_9 (Dense)	(None, 2)	3802

Рисунок 3.15 – Перевірка результату навчання

```
=================================================================
Total params: 139329 (544.25 KB)
Trainable params: 139329 (544.25 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Навчання 6 епох
Epoch 1/6
15/15 [==============================] - 2s 33ms/step - loss: 0.6603 - accuracy: 0.6575 - val_loss: 0.6307 - val_accuracy: 0.7143
Epoch 2/6
15/15 [==============================] - 0s 9ms/step - loss: 0.5270 - accuracy: 0.7992 - val_loss: 0.5027 - val_accuracy: 0.8038
Epoch 3/6
15/15 [==============================] - 0s 11ms/step - loss: 0.3924 - accuracy: 0.8436 - val_loss: 0.3902 - val_accuracy: 0.8348
Epoch 4/6
15/15 [==============================] - 0s 10ms/step - loss: 0.3010 - accuracy: 0.8647 - val_loss: 0.3499 - val_accuracy: 0.8554
Epoch 5/6
15/15 [==============================] - 0s 10ms/step - loss: 0.2562 - accuracy: 0.8837 - val_loss: 0.3312 - val_accuracy: 0.8640
Epoch 6/6
15/15 [==============================] - 0s 9ms/step - loss: 0.2286 - accuracy: 0.9027 - val_loss: 0.3199 - val_accuracy: 0.8744
```

Рисунок 3.16 – Перевірка результату навчання

Після завершення навчання моделі проводиться оцінка її працездатності. Це включає в себе аналіз зміни значень метрик якості (наприклад, середньої помилки та точності) під час навчання на навчальних та перевірочних даних. Графічне представлення цих метрик дозволяє візуально оцінити ефективність навчання моделі та виявити ознаки перенавчання або недостатньої навчання.

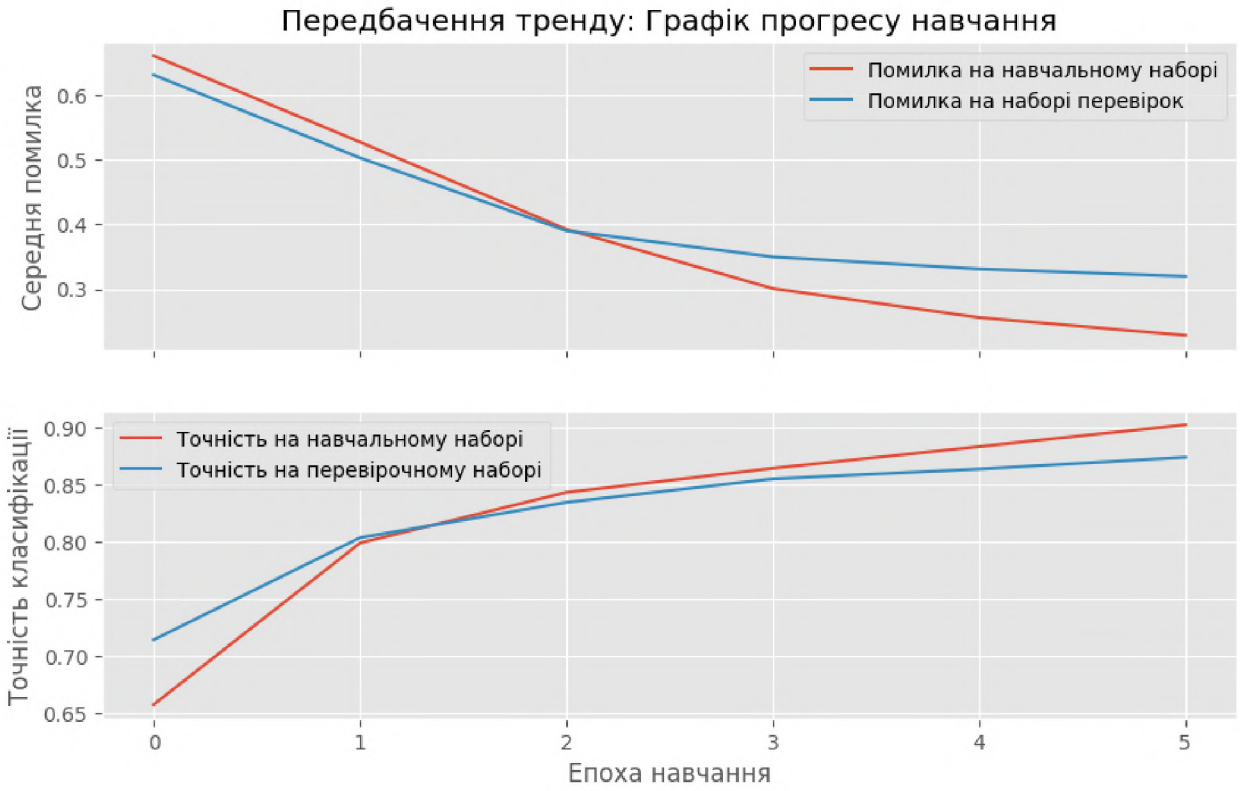

Рисунок 3.17 – Графіки прогресу навчання нейронної мережі

Додатково, для оцінки якості роботи моделі може бути використана матриця помилок. Ця матриця дозволяє визначити, наскільки ефективно модель класифікує дані на основі даних про реальні та передбачені значення.

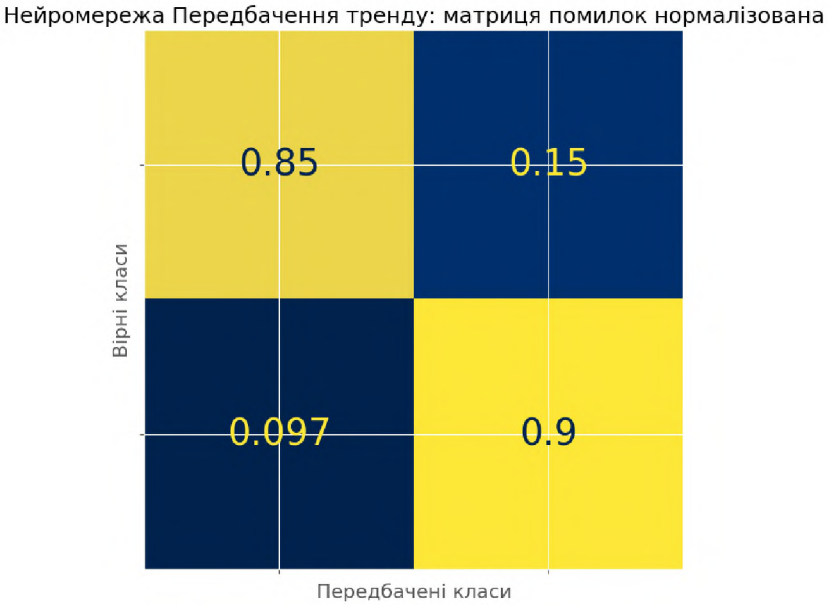

Рисунок 3.18 – Матриця помилок

Модель має загальну кількість параметрів, рівну 139,329, і всі вони навчаються. Після навчання протягом 6 епох ми отримали наступні результати:

– помилка на навчальному наборі: ≈ 0.2286;

– точність на навчальному наборі: ≈ 0.9027;

– помилка на наборі перевірок: ≈ 0.3199;

– точність на наборі перевірок: ≈ 0.8744;

З цих результатів видно, що точність моделі на наборі перевірок становить близько 87,44 %, що свідчить про її досить хорошу працездатність. Графіки також вказують на те, що модель навчається ефективно, оскільки помилка на навчальному та перевірочному наборах даних поступово зменшується, а точність збільшується протягом епох навчання.

3.4 Економічне обґрунтування прийнятих рішень

Для створення нейронної мережі, що дозволить реалізувати систему інтелектуального поливу рослин, необхідно врахувати кілька ключових аспектів, що впливають на загальні витрати. Це включає витрати на розробку коду, оплату праці спеціалістів, вартість використання хмарних сервісів для навчання моделі та інші супутні витрати.

Вартість розробки нейронної мережі.

1. Заробітна плата спеціалістів. За даними Glassdoor, середня заробітна плата спеціаліста з Python у США становить близько $58,000 на рік. В Україні ця цифра може бути нижчою і становити приблизно $10,800 на рік залежно від рівня спеціаліста та компанії.

2. Вартість одного рядка коду на Python. Розрахунок вартості одного рядка коду залежить від складності завдання. В середньому, розробка одного рядка коду може коштувати від $0.5 до $1. Якщо взяти середнє значення $0.75, то можна оцінити загальні витрати на написання коду.

3. Кількість рядків коду в мережі. Припустимо, що середня нейронна мережа для подібних завдань має близько 10,000 рядків коду. Це оцінка для невеликого проекту, який може бути масштабованим.

4. Вартість написання коду. Вартість написання 10,000 рядків коду при середній ціні $0.75 за рядок:

$$10,000 * 0.75 = \$7,500.$$

Абонплата за Google Collaboratory. Google Collaboratory пропонує різні плани для використання хмарних ресурсів, включаючи GPU та TPU:

1. Безкоштовний план. Обмежені ресурси, які можуть бути недостатніми для великих проектів.

2. Google Colab Pro. Вартість: $9.99 на місяць. Переваги: доступ до швидших GPU, збільшений час виконання, пріоритетний доступ до ресурсів.

3. Google Colab Pro+. Вартість: $49.99 на місяць. Переваги: ще швидші GPU, більше пам'яті, ще довший час виконання.

Для тривалих проектів може бути доцільно використовувати Google Colab Pro або Pro+, залежно від потреб у ресурсах. Припустимо, проект триватиме 6 місяців:

Google Colab Pro на 6 місяців:

$$6 * 9.99 = \$59.94.$$

Google Colab Pro+ на 6 місяців:

$$6 * 49.99 = \$299.94.$$

Загальні витрати. Підсумовуючи всі витрати:

1. Заробітна плата спеціалістів (наприклад, 1 рік для одного розробника в Україні): $10,800.
2. Вартість написання коду: $7,500.
3. Абонплата за Google Colab Pro на 6 місяців: $59.94 (або $299.94 для Pro+).

Загальні витрати при використанні Google Colab Pro:

$$10,800 + 7,500 + 59.94 = \$18,359.94.$$

Загальні витрати при використанні Google Colab Pro+:

$$10,800 + 7,500 + 299.94 = \$18,599.94.$$

Таким чином, загальні витрати на створення та підтримку інтелектуальної мережі для поливу рослин можуть коливатися в межах $18,359.94 до $18,599.94, залежно від обраного плану Google Colab. З урахуванням поточного курсу валют НБУ – максимальна сума складає:

$$39.85 \times 18,599.94 \approx 741210 \text{ грн.}$$

ВИСНОВКИ

У багатьох регіонах світу проблема нестачі води стає все більш нагальною. Інтелектуальні системи поливу дозволяють значно зменшити витрати води шляхом оптимізації режимів поливу, що сприяє збереженню цього цінного ресурсу.

Правильне зволоження ґрунту є ключовим фактором для забезпечення високої врожайності. Системи, які можуть автоматично налаштовуватися в залежності від погодних умов, типу ґрунту та потреб рослин, допомагають досягти оптимальних умов для їх росту.

Використання інтелектуальних систем дозволяє знизити енергетичні витрати, пов'язані з поливом, завдяки більш точному та своєчасному керуванню процесом.

Розробка таких систем включає використання сучасних технологій, таких як Інтернет речей (IoT), штучний інтелект та машинне навчання. Це сприяє розвитку технологічного сектору та впровадженню інновацій у повсякденне життя.

Зменшення надлишкового поливу допомагає уникнути негативних екологічних наслідків, таких як ерозія ґрунтів, засолення та змивання поживних речовин.

Автоматизація процесів поливу дозволяє знизити витрати на обслуговування та експлуатацію систем, що робить їх економічно вигідними для фермерів та садівників.

З огляду на вищезазначені фактори, розробка інтелектуальної системи поливу є важливим напрямком досліджень та впроваджень, що відповідає сучасним вимогам сталого розвитку та ефективного управління ресурсами. На основі бази Google Colaboratory було побудовано та навчено нейронну мережу, яка здатна аналізувати та передбачати необхідність поливу на основі даних, отриманих від вологості ґрунту та кліматичних умов. Спроби зі створення автоматизованої системи прогнозування та оптимізації поливу на

основі штучного інтелекту мають потенціал для підвищення ефективності агрокультури та зменшення витрат на зрошення. Проте, для досягнення повного потенціалу цих систем потрібно проводити подальші дослідження та вдосконалення методів та моделей.

Таким чином, результатами роботи є система автоматичного поливу на основі штучного інтелекту. Вони можуть бути використані для подальших досліджень за даною тематикою та при проектуванні розумної ферми.