НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
ім. Ігоря Сікорського

Навчально-науковий комплекс «Інститут прикладного системного аналізу»
(повна назва інституту/факультету)

Кафедра Системного проектування
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
“___”_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки 6.050101 Комп’ютерні науки
(код і назва)

на тему: Система моніторингу та прогнозування стану середовища на основі технології Інтернет речей _____

Виконав (-ла): студент (-ка) _4_ курсу, групи __ДА-32_____
(шифр групи)

Бондаренко Наталія Сергіївна _____
(прізвище, ім’я, по батькові) (підпис)

Керівник старший викладач Британ О.А. _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант економічний розділ _____ _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Нормоконтроль старший викладач Британ О.А. _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2017 року

Національний технічний університет України
«Київський політехнічний інститут»
ім. Ігоря Сікорського

Інститут (факультет) ННК «Інститут прикладного системного аналізу
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050101 Комп'ютерні науки
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
«___»_____20__ р.

ЗАВДАННЯ
на дипломну роботу студенту
Бондаренко Наталії Сергіївні
(прізвище, ім'я, по батькові)

1. Тема роботи Система моніторингу та прогнозування стану середовища на основі технології Інтернет речей _____ _____ ,
керівник роботи Бритов О.А., ст. викладач _____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом роботи 10.06.2017

3. Вихідні дані до роботи _____

Система моніторингу та прогнозування стану середовища, яка:

- Виконує збір даних з навколишнього середовища;
- Виконує збереження даних;
- Забезпечує зручний доступ до даних за запитом користувача;
- Виконує прогнозування показника врожайності та інших супутніх параметрів;

Платформа Raspberry Pi 2 Model B, бібліотеки для роботи з GPIO, сенсор DHT22, сенсор DS18B20, технологія Internet of Things, база даних PostgreSQL,

фреймворк Ruby on Rails, мова програмування Python 2.7, операційна система Raspbian Jessie, операційна система Mac OS 10.11.6, прогнозуючі алгоритми.

4. Зміст роботи

- Розглянути та проаналізувати предметну область: дослідити технологію Internet of Things, зробити огляд прогнозуючих моделей та методів;
- Вибрати прогнозуючу модель: виділити предмет прогнозування, обрати вхідні дані, визначити тип прогнозу, обрати методи, визначити зв'язки між показниками, побудувати модель;
- Виконати проектування та реалізацію системи: виділити функціональні та нефункціональні вимоги до системи, виконати проектування складових системи – пристрою та серверу, виконати реалізацію.
- Виконати функціонально-вартісний аналіз програмного продукту, визначити основні функції продукту, виділити варіанти реалізації функцій, обрати систему параметрів продукту, зробити їх оцінювання, зробити економічний аналіз варіантів розробки продукту.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

1. Діаграма потоків даних – плакат.
2. Діаграма розгортання – плакат.
3. Приклади роботи системи – плакат.
4. Фрагменти архітектури системи – плакат.
5. Презентація до захисту роботи.

6. Консультанти розділів роботи*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Рощина Н.В., доц.		

7. Дата видачі завдання 1.02.2017р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	01.02.2017	
2	Збір інформації	15.02.2017	
3	Ознайомлення з літературою і підготовка теоретичної частини роботи	28.02.2017	
4	Аналіз вимог завдання, вибір методів і засобів розв'язання поставленої задачі.	15.03.2017	
5	Розробка архітектури системи	25.03.2017	

* Консультантом не може бути зазначено керівника дипломної роботи.

6	Проектування макету пристрою. Тестування пристрою	10.04.2017	
7	Вибір прогнозуючої моделі. Пошук даних для моделювання.	25.04.2017	
8	Розробка програмної реалізації моделі та взаємодії з користувачем	07.05.2017	
9	Оформлення дипломної роботи	31.05.2017	
10	Отримання допуску до захисту та подача роботи в ДЕК	10.06.2017	

Студент _____ Бондаренко Н.С.
(підпис) (ініціали, прізвище)

Керівник роботи _____ Бритов О.А.
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

бакалаврської дипломної роботи Бондаренко Наталії Сергіївни на тему: “Система моніторингу та прогнозування стану середовища на основі технології Інтернет речей”

Дипломна робота присвячена розробці системи для моніторингу стану середовища та надання прогнозів спеціалістам з агрономії щодо стану агрономічної системи в майбутньому. Метою роботи є аналіз існуючих підходів до створення прогнозуючих моделей, та реалізація додатку із прогнозуючими властивостями.

У роботі проведено дослідження найсучасніших методів прогнозування та виділено їх переваги та недоліки у контексті вимог до системи. Клієнтська частина системи була розроблена на базі платформи Raspberry Pi. Серверна частина системи була реалізована у вигляді веб-сервісу.

Загальний обсяг роботи: 93 сторінки, 30 ілюстрацій, 9 таблиць, 17 посилань , 1 додаток на 8 сторінках.

Ключові слова: Інтернет речей, веб-додаток, прогнозування, причинно-наслідкове прогнозування, навколишнє середовище, лінійна регресія, модель авторегресії та ковзного середнього.

АННОТАЦИЯ

бакалаврской дипломной работы Бондаренко Наталии Сергеевны на тему: «Система мониторинга и прогнозирования состояния среды на основе технологии Интернет вещей»

Дипломная работа посвящена разработке системы для мониторинга состояния среды и предоставления прогнозов специалистам по агрономии о состоянии агрономической системы в будущем. Целью работы является анализ существующих подходов к созданию прогнозирующих моделей, и реализация приложения с прогнозирущими свойствами.

В работе было проведено исследование современных методов прогнозирования и определены их преимущества и недостатки в контексте требований к системе. Клиентская часть системы была разработана на базе платформы Raspberry Pi. Серверная часть системы была реализована в виде веб-сервиса.

Общий объем работы 93 страницы, 30 иллюстраций, 9 таблиц, 17 ссылок, 1 приложение на 8 страницах.

Ключевые слова: Интернет вещей, веб-приложение, прогнозирования, причинно-следственное прогнозирования, окружающая среда, линейная регрессия, модель авторегрессии и скользящего среднего.

ABSTRACT

of the barchelor`s thesis of Bondarenko Nataliia Sergyivna

"System for monitoring and forecasting the state of the environment based on Internet of things"

Thesis is devoted to developing a system for monitoring the environment and providing forecasts for agronomy experts on the state of agronomic systems in the future. The aim is to analyze existing approaches of forecasting models creating, application and implementation of forecasting properties.

The paper studies the advanced methods of forecasting and highlighted their strengths and weaknesses in the context of system requirements. The client part of the system was developed based on the platform Raspberry Pi. The server component of the system was implemented as a web service.

The thesis contains 93 pages, 30 pictures, 9 tables, 17 references and 1 appendix (8 pages).

Keywords: Internet of Things, web application, forecasting, causal forecasting, environment, linear regression model and autoregressive integrated moving average model.

ЗМІСТ

ВСТУП

Комп'ютерні та інформаційні технології вже давно перестали обслуговувати лише проблеми, пов'язані з військовою галуззю, як це було на початкових етапах їх розвитку більш ніж півстоліття тому. На сьогоднішній день вони стають все більш інтегрованими в прикладні галузі, пропонуючи свій інструментарій для вирішення системних проблем.

Аграрна промисловість стала однією з тих областей, в який вплив інформаційних технологій є досить слабким, особливо в Україні. Різноманітні автоматизовані ферми, «розумні» теплиці все ще обмежуються поодинокими випадками. Подальший розвиток сільського господарства вимагає використання сучасних технологій і обладнання, які економлять ресурси і підвищують ефективність праці. У всьому світі зростає попит на послуги зі збору та обробки точних даних в інтересах точного землеробства. Такі дані можуть бути використані для планування і контролю етапів сільськогосподарського виробництва, економії посівного матеріалу і добрив, що дозволить ефективно використовувати час і грошові ресурси, а також значно поліпшити стан агротериторій.

Одним з інноваційних рішень, що активно захоплюють ринок в аграрному середовищі (принаймні в США), є безпілотні технології. Використання дронів дозволяє аграріям моніторити стан і якість оранки і посіву, прогнозувати урожай, захищати долі від пожеж і розкрадань і навіть локально покращувати стан ґрунту, виступаючи в якості носія добрив. Але така система має ряд істотних недоліків: труднощі в закладці маршруту для дрона, залежність від погодних умов, висока вартість, а також складність підтримки робочого стану.

Для розв’язання поставленої задачі в рамках даної роботи було прийнято рішення розробити систему, яка дозволила б виконувати моніторинг та контроль стану посівних угідь без активного втручання людини.

Система на базі технології Інтернет речей припускає наявність пристрою (базового компоненту в рамках технології) та сервера, який виконує обробку даних, їх представлення, також формує прогнози обраних показників на основі отриманих даних.

Досягнення поставленої мети вимагає вирішення ряду задач:

- Вибір технічної бази
- Аналіз технологій та методів прогнозування
- Проектування та розробка моделі прогнозування

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

1.1 Дослідження технології Internet of Things

1.1.1 Огляд

Інтернет речей (IoT) – це мережева парадигма, що передбачає наявність взаємопов'язаних, “розумних” об'єктів, які безперервно генерують дані і передають їх через Інтернет. Це мережа, що складається із взаємозв'язаних фізичних об'єктів (речей) або пристроїв, які мають вбудовані давачі, а також програмне забезпечення, що дозволяє здійснювати передачу і обмін даними між фізичним світом і комп'ютерними системами, за допомогою використання стандартних протоколів зв'язку. Крім давачів, мережа може мати виконавчі пристрої, вбудовані у фізичні об'єкти і пов'язані між собою через дротові і бездротові мережі. Ці взаємопов'язані об'єкти (речі) мають можливість зчитування та приведення в дію, функцію програмування та ідентифікації, а також дозволяють виключити необхідність участі людини, за рахунок використання інтелектуальних інтерфейсів.[1] Стратегія роботи з даними для IoT має декілька етапів: збір, обробку, розповсюдження та використання даних.

Концепція IoT не є новою. Програмований логічний контролер (ПЛК) з 1970-их років є мікро-моделлю системи IoT, яка широко використовувалась для управління верстатами і процесами на заводах. ПЛК є системою, що складається з входів (датчиків, приводів, вимикачів), виходу (дані у цифровому або аналоговому вигляді), процесора, і включає зв’язок між ними. Такі системи знаходилися в межах заводу і не були підключеними до мережі Інтернет. Наразі існує цілий ряд технологій, які роблять втілення концепції Інтернету Речей у життя можливим (табл. 1.1).

Таблиця 1.1 – Технології, що становлять підґрунтя IoT систем (відповідно до дослідження, проведеного компанією Deloitte)[1]

Технологія	**Визначення**
Сенсор	Пристрій, який генерує електричний сигнал від фізичного стану або події.
Мережа	Механізм для передачі електронного сигналу.
Механічно доповнений інтелект	Аналітичні інструменти, які покращують здатність описувати, передбачати і використовувати відносини між явищами.
Аргументована поведінка	Технології та методи, які дозволяють виконувати дії, базуючись на типовій поведінці людини у даній ситуації.

Технології, зазначені у таблиці, стають з часом все більш доступними. Нова версія Інтернет-протоколу (IPv6), що підтримує адреси розміром 128-біт, що відповідає 340 трильйонам адрес (3,4x1038), а це 5x1028 адрес на кожну людину, дозволяє практично необмежену кількість пристроїв, підключених до мереж. Ціни датчиків суттєво знизилися за останні десятиліття. Завдяки закону Мура розмір та ціна інтегрованих процесорів знизилися, а їх можливості достатньо зросли, щоб задовольнити вимірювання з великою точністю та навіть первинну обробку вимірювань. Ціни модулів підтримки бездротових мереж (Bluetooth, Wi-Fi, 3G) також знижуються в геометричній прогресії. Така доступність технологій дозволяє розробити достатньо потужну IoT мережу з мінімальними витратами.

Система, побудована за допомогою технології Internet of things може мати різну архітектуру в залежності від ролей, відведених її компонентам.

На рисунку 1.1 зображено чотири основні компоненти, зв'язки між ними та функції, які вони можуть виконувати у рамках загальної мережі. [1]

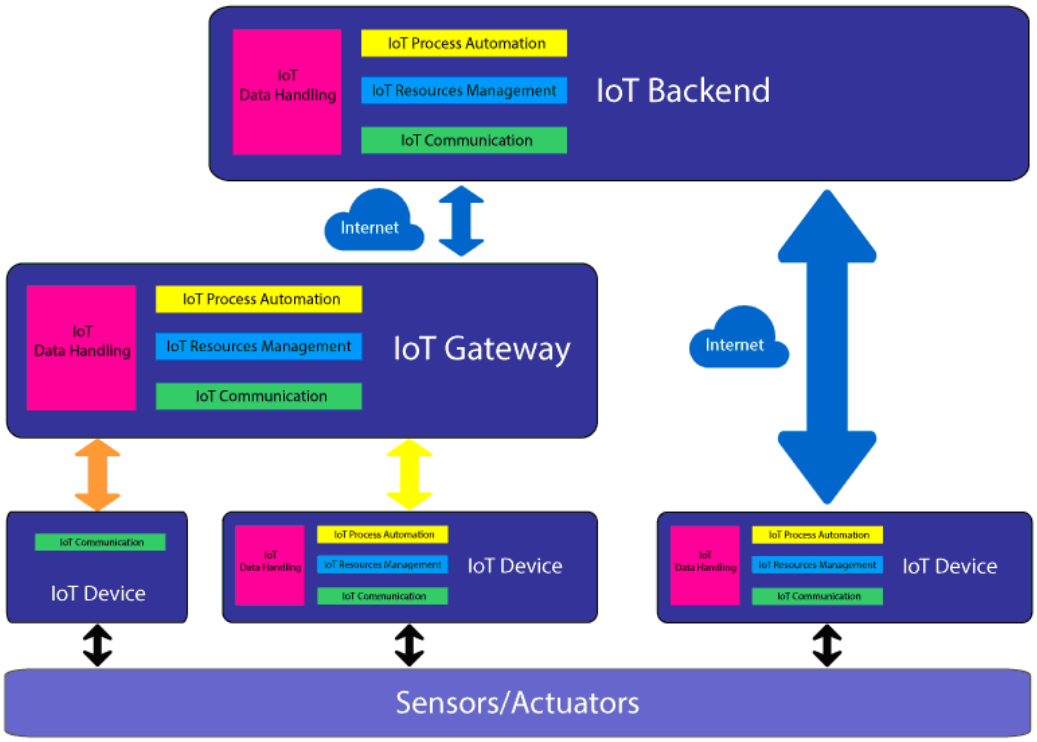

Рис. 1.1 – Схема системи побудованої відповідно до технології Internet of Things. [1]

1.1.2 Складові системи IoT

- Сенсори. В залежності від цілі та призначення мережі сенсори можуть відрізнятися за часом життя, чутливістю, часом відповіді, рівнем енергоспоживання, вартістю, часом життя та призначенням.
- Пристрій (device, thing). Електронний пристрій: мікрокомп'ютер, платформа з мікроконтролером, телефон, смартфон, натільний пристрій, який часто безпосередньо поєднується з сенсорами і отримує від них дані.
- Пристрій-шлюз (gateway). Використовується в деяких IoT системах для проміжного збору та обробки даних. Прикладом може слугувати система "розумного будинку", в якому всі пристрої (телефони, холодильники, мікрохвильовки тощо) посилають дані в локальній мережі на проміжні

пристрої, наприклад смартфони, які, в свою чергу, посилають оброблені дані з певною систематичністю на сервер через мережу Інтернет.

- Сервер, хмара, сховище даних (backend). В залежності від призначення та об’ємів IoT мережі дані можуть оброблятися та зберігатися в різних середовищах. Це можуть бути, наприклад, високопотужні сервери з встановленими системами управління нереляційними базами даних у випадку, якщо обсяги даних дійсно є дуже великими та потребують особливих методів зберігання.

1.1.3 Різновиди мереж в залежності від завдань, що виконуються складовими мережі

- Пристрій-Сервер. Система, в якій пристрій, що збирає дані, має достатню обчислювальну здатність, щоб виконувати первинну обробку отриманих від сенсорів показань. Після обробки пристрій відправляє дані на сервер або тримає їх у тимчасовій пам’яті до запиту сервера.
- Передавач-Шлюз-Сервер. Система, у якій пристрій, що збирає дані з датчиків, не виконує ніяких функцій, крім передачі даних на шлюзовий пристрій, який вже потім виконує попереднє очищення, обробку та впорядкування даних.
- Пристрій-Шлюз-Сервер. Кожна складова такої системи може виконувати обробку даних на різному рівні та контролювати процес комунікації з іншими складовими.

Гібридна система. Така мережа будується з використанням декількох видів зв’язків між пристроями, але з дотриманням ієрархії, що є для них загальною.

1.1.4 Проблема безпеки в IoT

Забезпечення безпеки продуктів і послуг IoT має бути основним пріоритетом в цій галузі. Через постійне збільшення числа пристроїв, підключених до Інтернету, виникають нові потенційні уразливі місця. Недостатньо захищені пристрої можуть служити точками доступу для

кібератак, дозволяючи зловмисникам перепрограмувати пристрій або викликати його несправність. Пристрої недосконалої конструкції з недостатнім рівнем захисту потоків даних можуть спричиняти уразливість даних користувачів. Для пристроїв, які зазвичай використовуються для передачі даних з місць на сервер і також є підключеними до глобальної мережі, питання безпеки є навіть серйознішим, ніж для традиційних комп'ютерів. Наприклад, незахищений холодильник в США, заражений шкідливим програмним забезпеченням, може відправляти тисячі шкідливих повідомлень електронної пошти одержувачам у всьому світі за допомогою домашнього підключення Wi-Fi.

Саме тому безпека пристроїв і послуг IoT є основою темою обговорень у спільноті і повинна бути визнана критично важливою проблемою. Міра залежності світу від глобалізованих пристроїв поступово й неухильно зростає і тому потрібно всіма способами знижувати ризики, пов'язані з непередбачуваністю їх роботи.

1.1.5 Проблема конфіденційності даних в IoT

Досить часто Інтернет речей є глобальною мережею сенсорних пристроїв, які збирають дані про оточення і нерідко - про людей. Звичайно, ці дані можуть бути корисними для власників пристроїв, але дуже часто вони представляють інтерес і для виробників і постачальників пристроїв.

Комбінації потоків IoT-даних, що на перший погляд здаються нешкідливими, також можуть становити загрозу персональній конфіденційності. При об'єднанні або зіставленні кількох потоків даних іноді можна отримати більш точний цифровий портрет людини, ніж при використанні одного потоку IoT-даних. Наприклад, підключена до Інтернету зубна щітка може записувати і передавати нешкідливі дані про те, як її власник чистить зуби. Але якщо його холодильник передає дані про те, що він їсть, а фітнес-трекер передає дані про його фізичну активність, то комбінація цих потоків дозволяє отримати детальніший і точніший опис загального стану

здоров'я цієї людини. Ефект групування даних може бути особливо справедливим по відношенню до IoT-пристроїв, так як багато пристроїв генерують додаткові метадані (наприклад, час і місце розташування), які дозволяють отримати конкретнішу інформацію про людину.

Такого роду функції можуть не тільки приносити користь обізнаним користувачам, але і створювати проблеми конфіденційності тим, хто не підозрює про присутність цих пристроїв і не може контролювати використання зібраних даних.

1.1.6 Проблема стандартизації в IoT

У традиційному Інтернеті інтероперабельність пристроїв представляє ключову цінність. Найважливіша вимога до Інтернет-підключення полягає в тому, щоб «пов'язані» системи були здатні «говорити однією мовою» протоколів і кодів. Так звані «закриті платформи», де користувачі мають можливість взаємодіяти лише в рамках обмеженого набору веб-сайтів і сервісів, можуть значно знизити соціальні, політичні і економічні переваги від доступу до необмеженому Інтернет-простору.

В умовах повної інтероперабельності будь-який IoT-пристрій міг би встановлювати зв'язок з будь-яким іншим пристроєм або системою і виконувати бажаний обмін інформацією. Але на практиці інтероперабельність є складнішим явищем. Взаємодія між IoT-пристроями і системами відбувається на різних рівнях і в різних шарах в рамках стека комунікаційних протоколів між пристроями. Крім того, повна інтероперабельність по всьому діапазону технічної продукції не завжди здійсненна, необхідна або бажана, особливо якщо нав'язувати її штучно, що може послужити бар'єром для інвестицій і інновацій.

Стандартизація та прийняття протоколів, що визначають принципи зв'язку (в тому числі реальну потребу в наявності стандартів), є основною темою дискусій, що стосуються Інтернету речей.

1.2 Огляд прогнозуючих моделей та методів

1.2.1 Прогнозування

Прогнозування — це процес передбачення майбутнього стану предмета чи явища на основі аналізу його минулого і сучасного, систематично оцінювана інформація про якісні й кількісні характеристики розвитку обраного предмета чи явища в перспективі. [2]

Основною задачею прогнозування не є надання чіткої інформації про стан предмета, радше зниження рівня невизначеності щодо нього. Прогнози в подальшому розшифровуються людьми, які мають додаткові відомості у тій чи іншій галузі і вже після обробки можуть становити цінність для користувача. Хто може стати “користувачем” прогнозу? Будь-яка людина – одні займаються прогнозуванням ринку валют, інші – економіко-політичної ситуації в країні, треті – щодня переглядають прогноз погоди.

Прогнозування має бути невід'ємною частиною діяльності по прийняттю управлінських рішень, так як може надавати критично важливу інформацію. Відповідно до призначення, цілей та масштабів прогнозування виділяють різні види прогнозів за тривалістю прогнозованого періоду.

1.2.2 Види прогнозів

В залежності від тривалості прогнозу їх поділяють на такі:

- **Короткострокові прогнози.** Прогнози тривалістю від 1 дня до 3 місяців. Необхідні для планування персоналу, виробництва і транспортування. В рамках процесу планування іноді бувають потрібні також прогнози попиту.
- **Прогнози середньої тривалості.** Прогнози тривалістю від 3 місяців до 2 років. Необхідні для визначення потреб ресурсів у майбутньому для того, щоб закуповувати сировину, наймати персонал, або купувати машини і обладнання.

- **Довгострокові прогнози.** Прогнози тривалість від 2 років. Використовуються в стратегічному плануванні. Такі рішення повинні враховувати можливості ринку, фактори навколишнього середовища та внутрішніх ресурсів.

1.2.3 Методи прогнозування

Методи прогнозування різняться між собою за великою кількістю характеристик: точністю, складністю призначенням та ін. Але найважливішим параметром вибору методу прогнозування є наявні дані, їх кількість та якість.

Методи прогнозування поділяються на три основні типи: якісні (qualitative) методи, прогнозування на основі аналізу часових рядів (time series analysis), а також причинно-наслідкові моделі (causal models). [2]

Перший підхід використовує якісні дані (експертний висновок, наприклад) і інформацію про подібні системи чи ситуації, яка може напряму не відноситися до прогнозу, і може взяти або не взяти до уваги історичні дані (яких, як правило, немає або вони наявні в недостатній кількості).

Другий підхід повністю фокусується на шаблонах і на їх зміні в часі і, отже, повністю залежить від історичних даних. Існує широкий спектр кількісних методів прогнозування, які часто розробляються в рамках конкретних дисциплін для конкретних цілей. Кожен з них має власні властивості, які необхідно враховувати при виборі конкретного методу. Більшість методів прогнозування на основі аналізу часових рядів використовують або серії часових рядів (зібраних через регулярні проміжки часу), або зрізові дані (зібрані в один момент часу). [3]

Дані у вигляді часових рядів є корисними, якщо залежність прогнозованої величини від часу (наприклад, ціни на акції, обсяги продажів, прибути і т.д.) дає змогу побудувати певні шаблони. Приклади часових рядів даних включають в себе:

- Щоденні ціни на акції IBM
- Місячна кількість опадів

- Поквартальні результати продажів на Amazon
- Річний прибуток Google

При прогнозування на основі аналізу часових рядів, мета полягає в тому, щоб оцінити, як часова послідовність буде поводити себе у майбутньому. На рисунку 1.2 показаний поквартальний графік виробництва пива в Австралії з 1992 до третього кварталу 2008 року.

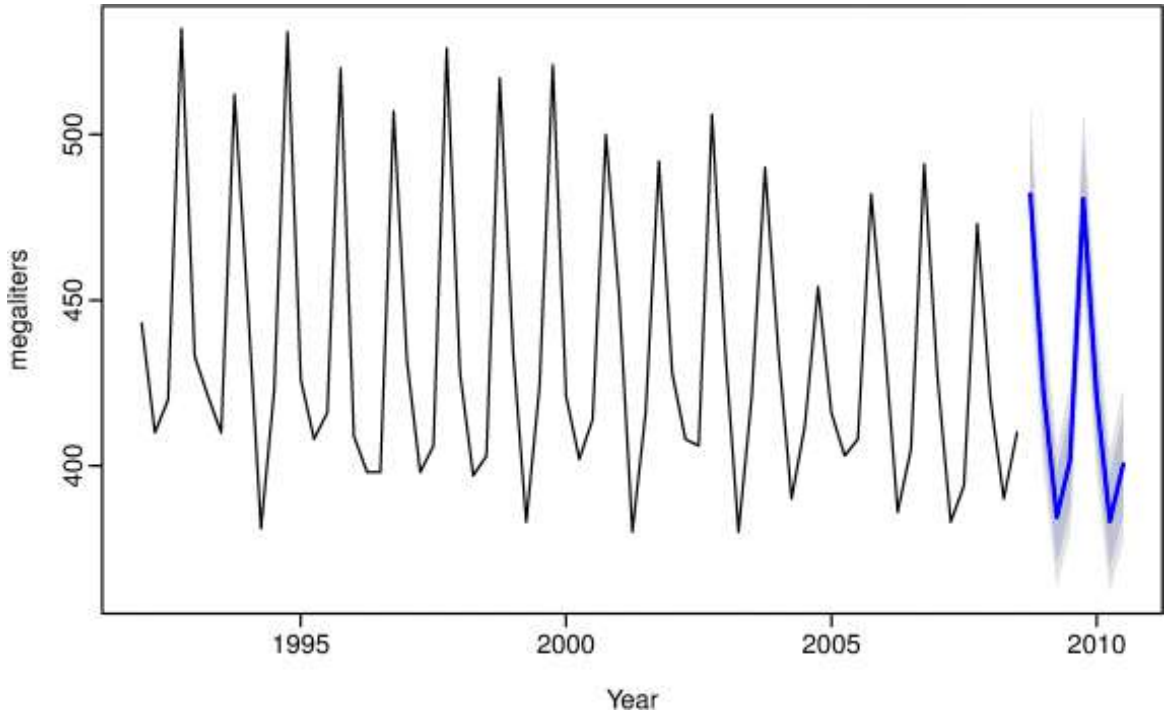

Рисунок 1.2 – Поквартальний графік виробництва пива в Австралії з першого кварталу 1992 року до третього кварталу 2008 року із прогнозом на два роки.[6]

Синя ламана лінія показує прогноз на найближчі два роки. Прогнозуюча модель відслідкувала сезонні повторення в історичних даних і застосували цей патерн до прогнозу на наступні два роки. Темно-сіре штрихування показує прогнозуючий інтервал із імовірністю 80%. Це означає, що прогнозована величина потрапить у цю область із імовірністю 80%. Світле штрихування показує прогнозуючий інтервал із імовірністю потрапляння 95%.

Такі інтервали є дуже корисним способом відображення ступеню невизначеності в прогнозах. У випадку, представленому на рисунку, очікується, що прогноз буде дуже точними, тому інтервали передбачення досить вузькі.

Цей метод використовує тільки інформацію про змінну, значення якої потрібно прогнозувати, і не робить жодних спроб виявити фактори, які впливають на її поведінку. Тому він буде екстраполювати тенденції і сезонні моделі та ігнорувати всі інші потенційно важливі дані, такі як маркетингові ініціативи, просування технічного прогресу, нестабільність економічних процесів, тощо.

Моделі цього класу включають інтегровану модель Бокса-Дженкінса (ARIMA), методи на основі експоненційного згладжування і структурні моделі.

Третій використовує ретельно підготовану інформацію про залежності між елементами системи, і є досить потужним, щоб враховувати унікальні та нетипові ситуації. Як і для методів прогнозування на основі аналізу часових рядів, історична інформація є важливою.

Такі моделі використовуються, коли прогнозована змінна залежить від одної або декількох інших змінних, які називаються предикторами. Мета причинно-наслідкових моделей полягає у тому, щоб описати характер взаємозв'язку і використовувати його для обчислення значень прогнозованої змінної. Відповідно до цієї моделі, будь-яка зміна предикторів цілком передбачувано впливатиме на значення прогнозованої змінної, якщо припустити, що характер зв’язків між змінними не змінився. Методи цього класу включають в себе модель регресії, адитивні моделі, а також деякі види нейронних мереж.

Розглянемо деякі з методів, що належать до описаних класів.

1.2.4 Прогнозування на основі часових рядів

Аналіз часових рядів враховує той факт, що дані вимірювані протягом довгого часу можуть мати внутрішню структуру (наприклад, автокореляція, тренд або сезонні коливання), які повинні бути враховані. [3]

Існує декілька найпростіших способів прогнозування на основі часових рядів, які в деяких випадках виявляються найефективнішими. Зазвичай вони використовуються у якості еталонних тестів для інших методів. Розглянемо деякі з них.

1.2.4.1 Метод середнього

Метод використовує середнє значення всіх наявних історичних даних по змінній у якості прогнозу цієї змінної. Метод може використовуватись при причинно-наслідковому прогнозуванні (коли ми передбачаємо значення, яке не входить в набір наявних даних) . В такому випадку прогнозом змінної вважається середнє значень всіх інших наявних змінних. [4]

1.2.4.2 Метод наївного прогнозування

Відповідно до цього методу прогноз змінної дорівнює значенню, отриманому під час останнього вимірювання цієї змінної. Цей метод працює на диво добре для багатьох економічних і фінансових часових рядів. [4]

1.2.4.3 Метод сезонного індексування

Метод, що є дуже схожим на попередній, але використовує при прогнозуванні знайдені у історичних даних сезонні патерни. В такому випадку прогноз для часу $T + h$ може бути записаний як

$$y_{T+h-km}, \text{де } m - \text{тривалість сезону}, k = \left\lfloor {(h-1)}/{m} \right\rfloor + 1 \qquad (1.1)$$

1.2.4.4 Метод ковзного (рухомого) середнього

Просте ковзне середнє, або арифметичне ковзне середнє чисельно дорівнює середньому арифметичному значень вихідної функції за встановлений період. Таким чином, прогноз для часу $T + h$ може бути знайдений з рівняння:

$$y_T + \frac{h}{T-1}\sum_{t=2}^{T}(y_t - y_{t-1}) = y_T + h\left(\frac{y_T - y_1}{T-1}\right) \quad (1.2)$$

Еквівалентом можна вважати лінію, проведену від першого до останнього значення і екстрапольовану на майбутнє. [5]

Приклади застосування наведених вище методів наведені на рисунках 1.3 та 1.4.

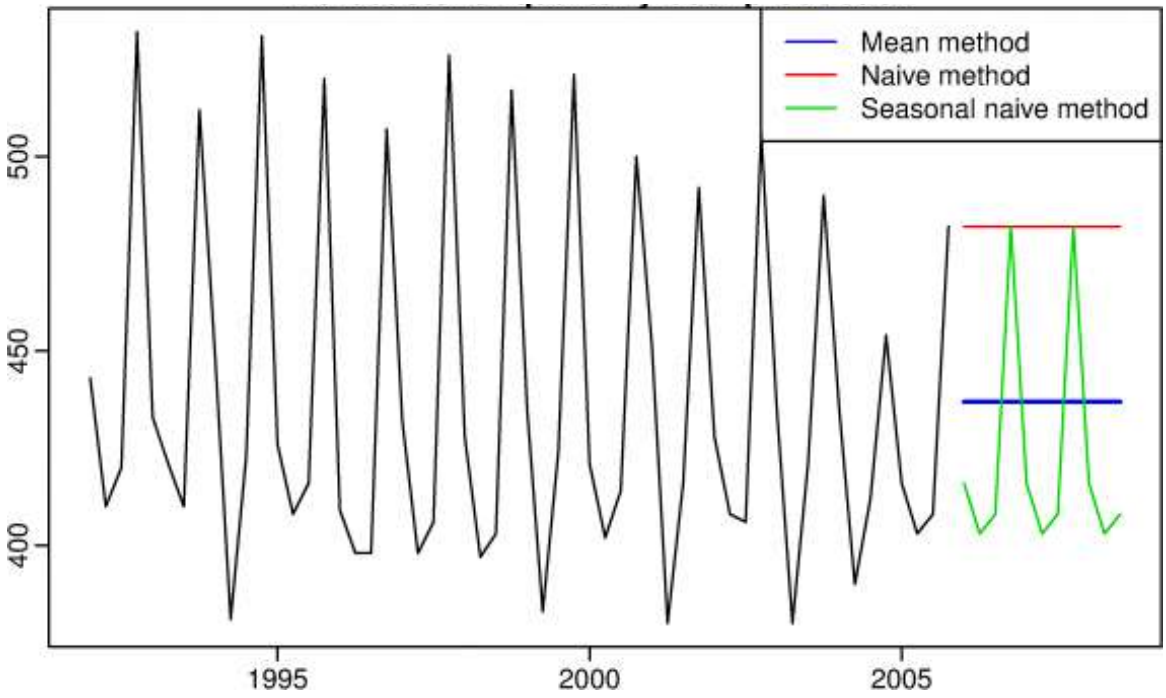

Рисунок 1.3 – Прогноз виробництва пива в Австралії.[6]

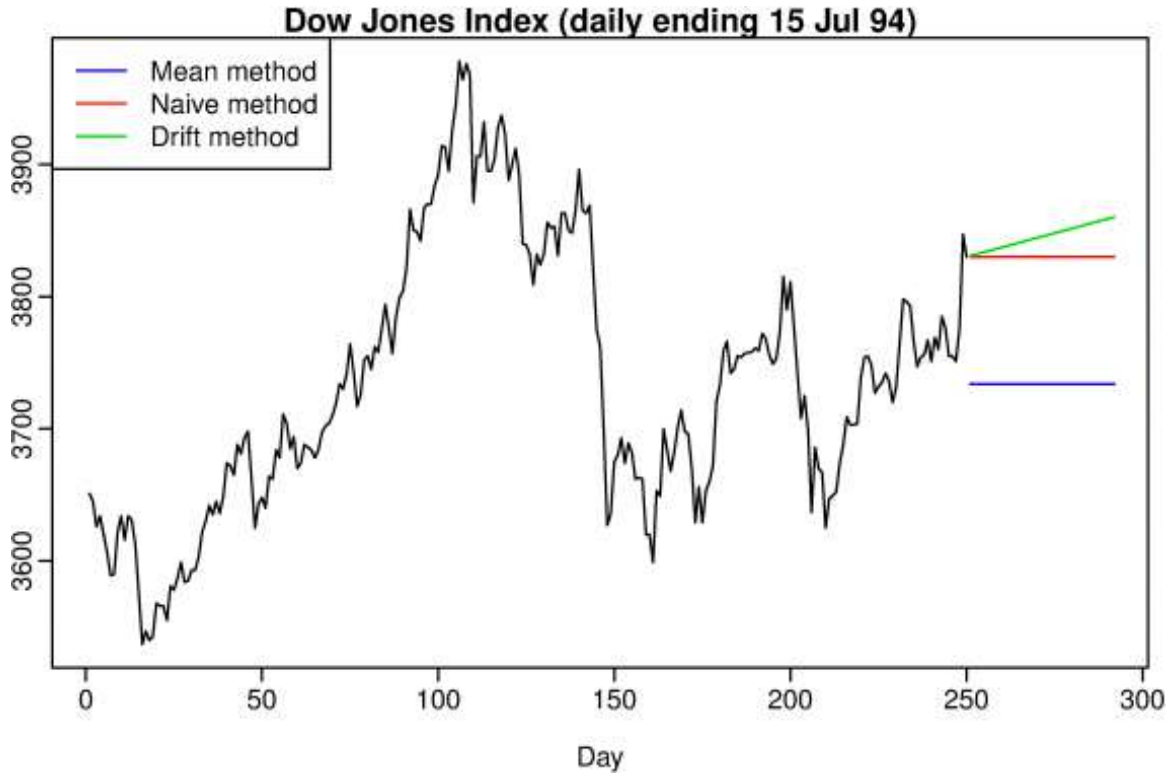

Рисунок 1.4 – Прогноз індексу Доу-Джонса, що базується на інформації за останні 250 днів.[6]

1.2.4.5 Експоненційне згладжування

Ідея експоненціального згладжування була запропонована в кінці 1950-х років і стала основою для деяких з найпопулярніших методів прогнозування. Результатом прогнозу, отриманого з використанням експоненційних методів згладжування є середньозважене значення минулих спостережень, але ступінь врахування зменшується з часом за експоненційним законом. Іншими словами, пізніші спостереження мають більшу вагу при обчисленні прогнозу.

Головне достоїнство прогнозної моделі, заснованої на експоненційних середніх, полягає в тому, що вона здатна послідовно адаптуватися до нового рівня процесу без значного реагування на випадкові відхилення. Ця система генерує надійні прогнози швидко і для широкого спектра часових рядів, що є великою перевагою. Це спонукало високу популярність застосування цієї ідеї у промисловості. [6]

Історично метод незалежно був розроблений Брауном і Холтом. Холт також розробив моделі експоненціального згладжування для процесів з постійним рівнем, процесів з лінійним зростанням і процесів з сезонними ефектами.

Процедура простого експоненціального згладжування здійснюється за такими формулами:

$$S_1 = X_0; \tag{1.3}$$

$$S_t = \alpha X_{t-1} + (1-\alpha) * S_{t-1}; \tag{1.4}$$

де X_{t-1} - фактичне спостереження в момент $t-1$;

S_t – значення експоненціального середнього в момент t;

α – параметр згладжування – $\alpha = const, \alpha \in (0; 1]$

Експоненціальне середнє в момент t тут виражено як зважена сума поточного спостереження і експоненціального середнього минулого спостереження з вагами α і $(1-\alpha)$ відповідно. Якщо послідовно використовувати дане рекурентне співвідношення, то значення S_t можна виразити через значення часового ряду X:

$$S_t = \alpha \sum_{i=0}^{\infty} (1-\alpha)^i * X_{t-i}; \tag{1.5}$$

Таким чином, величина S_t виявляється зваженою сумою всіх членів ряду. Причому значення ваг зменшуються експоненціально залежно від віддаленості спостереження щодо моменту t. Це і пояснює назву «експоненціальне середнє».

Експоненціальне згладжування можна уявити як фільтр, на вхід якого у вигляді потоку послідовно надходять члени вихідного ряду, а на виході

формуються значення експоненційних середніх. Причому, згладжений ряд S_t має теж математичне очікування, що і ряд X, але меншу дисперсію.

При високому значенні α дисперсія згладженого ряду не значно відрізняється від дисперсії ряду X. Чим менше α, тим більшою мірою скорочується дисперсія згладженого ряду (тобто придушуються коливання вихідного ряду).

Далі експоненціальне середнє можна використовувати для побудови короткострокових прогнозів. У цьому випадку передбачається, що вихідний ряд описується моделлю:

$$X_t = a_t + err_t; \tag{1.6}$$

де a_t - середній рівень ряду, що змінюється в часі;

err_t - випадкові відхилення з нульовим математичним сподіванням.

Прогнозна модель має вигляд:

$$\tilde{X}_{T+\tau} = \tilde{a}_T; \tag{1.7}$$

де $\tilde{X}_{T+\tau}$ -- прогноз, зроблений в момент Т на τ одиниць вперед,

$\tilde{a}_T$ -- оцінка a_T

Оцінкою параметра моделі a_T служить експоненціальне середнє ряду S_t. Таким чином, всі властивості експоненціального середнього поширюються на прогнозну модель. Зокрема, якщо привести рекурентну формулу до наступного вигляду:

$$S_t = S_{t-1} + \alpha * (X_{t-i} - S_{t-1}); \tag{1.8}$$

і розглядати S_{t-1} як прогноз на один крок вперед, то величина $(X_{t-i} - S_{t-1})$ буде похибкою цього прогнозу, а новий прогноз S_t виходить в результаті коригування попереднього прогнозу з урахуванням його помилки. В цьому і полягає суть адаптації.

На основі простого експоненціального згладжування були розроблені більш складні моделі згладжування часових рядів, що містять періодичні сезонні коливання і / або володіють тенденцією зростання.

Дана система дозволяє будувати поряд з простим експоненціальним згладжуванням моделі, що відображають ефекти зростання (лінійного, експоненціального або затухаючого) і сезонності (адитивного або мультиплікативного), якими володіє вихідний ряд.

Під сезонністю розуміють вплив зовнішніх факторів, що діють циклічно із заздалегідь відомою періодичністю. Часовий ряд і періодичні сезонні коливання, що він містить, можна уявити, як моделі двох основних типів: з адитивними та мультиплікативними сезонними компонентами.

Якщо у вихідному ряді спостерігаються досить постійні періодичні відхилення в абсолютному вираженні від змінюваного в часі середнього рівня ряду з заздалегідь відомим періодом, то сезонна компонента має **адитивну** природу, і в модель експоненціального згладжування для відображення цієї особливості додається додатковий параметр, який зазвичай позначається, як δ. У більшості випадків, коли сезонність присутня в початковому ряді, оптимальне значення параметра δ розташовується між нулем і одиницею. [6]

Особливістю **мультиплікативної** моделі є те, що сезонні відхилення від змінюваного в часі середнього рівня вихідного ряду носять характер досить стійкої відносного зміни (тобто для кожної точки сезонного циклу визначено зміна щодо даного рівня).

1.2.4.6 Модель ARIMA

Модель AR(I)MA - одна з найбільш популярних моделей для побудови короткострокових прогнозів. AR(I)MA (autoregressive (integrated) moving-

average model) – є комбінацією двох більш простих моделей аналізу часових рядів – модель авторегресії (AR) та модель ковзного середнього (MA). [7]

Модель авторегресії

Така модель може інтерпретуватися як лінійна модель множинної регресії, в якій в якості предикторів виступають минулі значення самої залежної змінної. Термін *авторегресія* наголошує, що це регресія змінної на саму себе. Авторегресійна модель порядку p може бути записана як:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + e_t; \quad (1.9)$$

де $c -$ константа,

e_t - білий шум,

y_k- відомі значення змінної.

Авторегресійні моделі є надиво гнучкими при обробці широкого спектру різних часових рядів. На рисунку 1.5 показано графіки, що відповідають AR(1) – моделі першого порядку та AR(2) – моделі другого порядку відповідно. Дисперсія змінної e_t, що відповідає за похибку, впливає лише на масштаб послідовності. Рівняння, що відповідають моделям:

$$\text{AR(1): } y_t = 18 - 0.8 y_{t-1} + e_t \quad (1.10)$$

$$\text{AR(2): } y_t = 8 + 1.3 y_{t-1} - 0.7 y_{t-2} + e_t \quad (1.11)$$

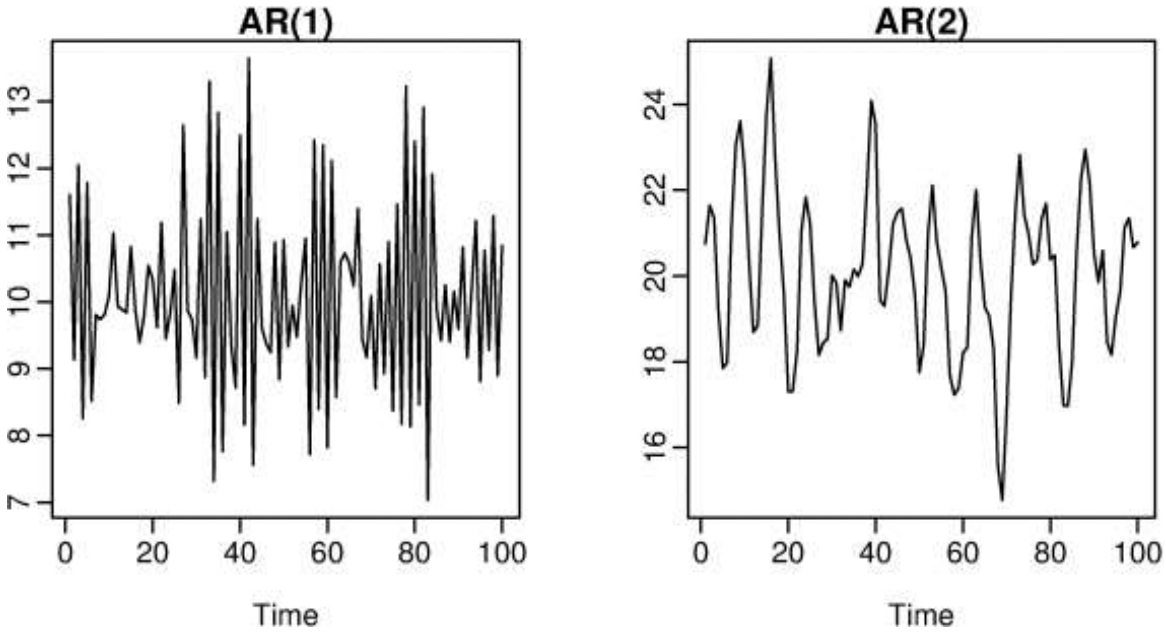

Рисунок 1.5 – AR(1) та AR(2)[6]

Моделі Бокса-Дженкінса є моделями стаціонарних часових рядів. Якщо часовий ряд є нестаціонарним і відноситься до класу DS-рядів, то перш за все його потрібно перетворити так, щоб він став стаціонарним.

Так як процес має бути стаціонарним, на коефіцієнти ϕ накладаються певні обмеження. Наприклад, для моделі першого порядку AR(1):

$$-1 < \phi_1 < 1 \tag{1.12}$$

Для моделі другого порядку AR(2):

$$-1 < \phi_1 < 1;\ \phi_1 + \phi_2 < 1;\ \phi_2 - \phi_1 < 1 \tag{1.13}$$

Згідно з методологією Бокса-Дженкінса побудова моделі AR(p) включає в себе наступні етапи:

1. Ідентифікація моделі;
2. Оцінювання параметрів моделі;
    a. Метод найменших квадратів;

b. Метод максимальної правдоподібності;
c. Метод моментів;

3. Аналіз адекватності моделі;
   a. Тестування параметрів моделі на статистичну значимість;
   b. Тестування залишків моделі на некорельованість та стаціонарність, нормальний розподіл;
   c. Економність моделі;
4. Прогнозування по моделі.

Модель ковзного середнього

Замість того щоб використовувати минулі значення прогнозованої змінної в регресії, модель ковзного середнього використовує минулі помилки прогнозу в моделі регресійного типу.

$$y_t = c + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \cdots + \theta_q e_{t-q} + e_t; \quad (1.14)$$

де e_t - білий шум

Звичайно, ми не вимірюємо значення e_t, так що це не можна називати регресією в звичному сенсі цього слова. При цьому кожне значення y_t можна розглядати як зважене ковзне середнє останніх кількох помилок прогнозування. На рисунку 1.6 показано графіки, що відповідають MA(1) – моделі першого порядку та MA(2) – моделі другого порядку відповідно. Рівняння, що відповідають моделям:

$$\text{MA(1)}: y_t = 2 + -0.8e_{t-1} + e_t \quad (1.15)$$

$$\text{MA(2)}: y_t = -e_{t-1} + 0.8e_{t-2} + e_t \quad (1.16)$$

Як і у випадку з AR моделями, дисперсія змінної e_t, впливає лише на масштаб послідовності.

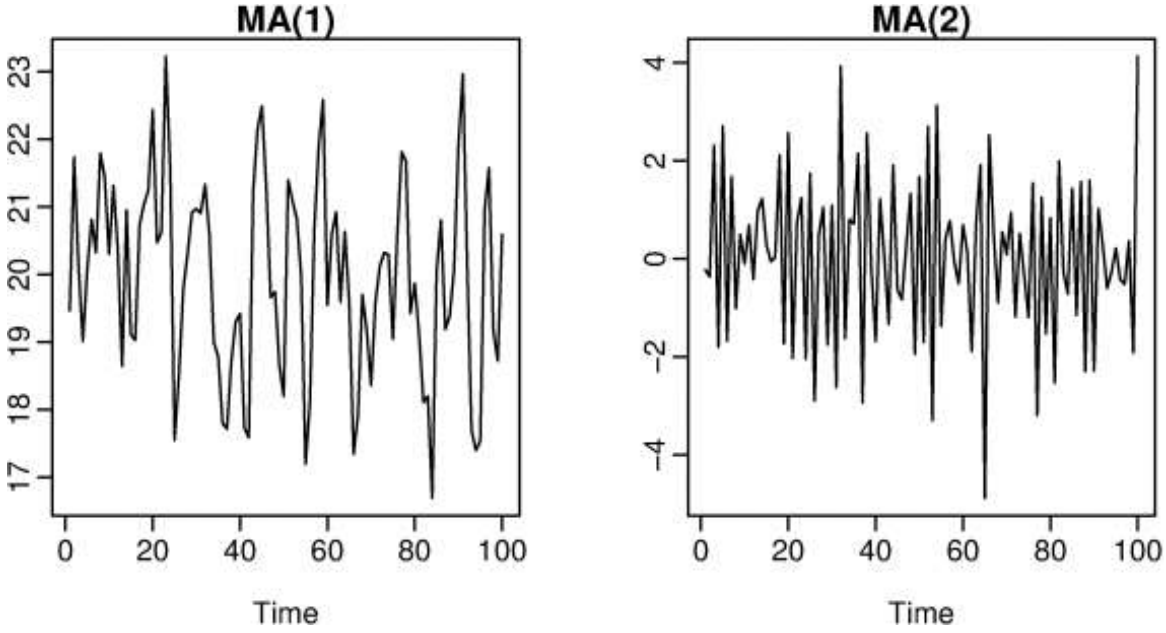

Рисунок 1.6 – MA(1) та MA(2)[6]

Всякий стаціонарний процес може бути представлений як процес ковзного середнього MA (∞) - процес з деякими коефіцієнтами (сума їх модулів повинна бути кінцевою). Тобто значення будь-якого стаціонарного часового ряду можна як завгодно точно наблизити, використовуючи деякий MA(q) - процес кінцевого порядку:

$$\begin{aligned} y_t &= \phi_1 y_{t-1} + e_t \\ &= \phi_1(\phi_1 y_{t-2} + e_{t-1}) + e_t \\ &= {\phi_1}^2 y_{t-2} + \phi_1 e_{t-1} + e_t \\ &= {\phi_1}^3 y_{t-3} + {\phi_1}^2 e_{t-2} + \phi_1 e_{t-1} + e_t \\ &= \cdots \end{aligned} \tag{1.17}$$

Протилежне перетворення можна виконати, якщо накласти деякі обмеження на параметри MA. Тоді модель MA називається «оборотною».

Тобто, що ми можемо написати будь-який оборотний MA(q) процес у вигляді AR(∞) процесу. У разі процесу MA(q) оборотність потрібна для більшої точності висновків і прогнозів, в разі процесу AR(p) - для стаціонарності. Звідси, перевірка оборотності процесу аналогічна перевірці стаціонарності.

Для скорочення кількості параметрів моделі, тобто пониження порядку q моделі MA(q) доповнюють авторегресійною частиною. Такі моделі прийнято називати ARMA-моделями.

1.2.5 Причинно-наслідкове прогнозування

Коли наявні історичні дані і було проведено достатньо досліджень, щоб виявити співвідношення між прогнозованою змінною та іншими чинниками (наприклад, соціально-економічні фактори, технічні фактори, погодні умови), прогнозисти часто віддають перевагу причинно-наслідковим моделям.

Причинно-наслідкова модель є найвитонченішим видом серед прогнозуючих інструментів. Окрім застосування співвідношень між предикторами, вона також може включати в себе результати аналізу часових рядів.

1.2.5.1 Регресійні моделі

Розглянемо дві неперервні змінні:

$$x = (x_1, x_2, x_3, \dots, x_n),\ y = (y_1, y_2, y_3, \dots, y_n).$$

Розмістимо точки на двовимірному графіку розсіювання і скажемо, що ми маємо лінійне співвідношення, якщо дані апроксимуються прямою лінією.

Якщо ми вважаємо, що у залежить від x, причому зміни в y викликаються саме змінами в x, ми можемо визначити лінію регресії (регресія y на x), яка найкраще описує прямолінійне співвідношення між цими двома змінними.

Математичне рівняння, яке оцінює лінію простої (парної) лінійної регресії:

$$y = b_0 + b_1 x + \varepsilon; \tag{1.18}$$

Вільний член b_0 дорівнює прогнозованому значенню y, коли $x = 0$. Кутовий коефіцієнт b_1 представляє передбачуване збільшення y в результаті збільшення x на одну одиницю. На рисунку 1.7 показано, що кожне спостережене значення містить дві складові – систематизовану частину моделі, представлену рівнянням, та довільну похибку ε_t. Похибка в такому разі не означає помилку моделі, а лише показує відхилення від базової моделі прямої лінії. Вона відображає всі чинники, які можуть впливати на y_i крім x_i. [8]

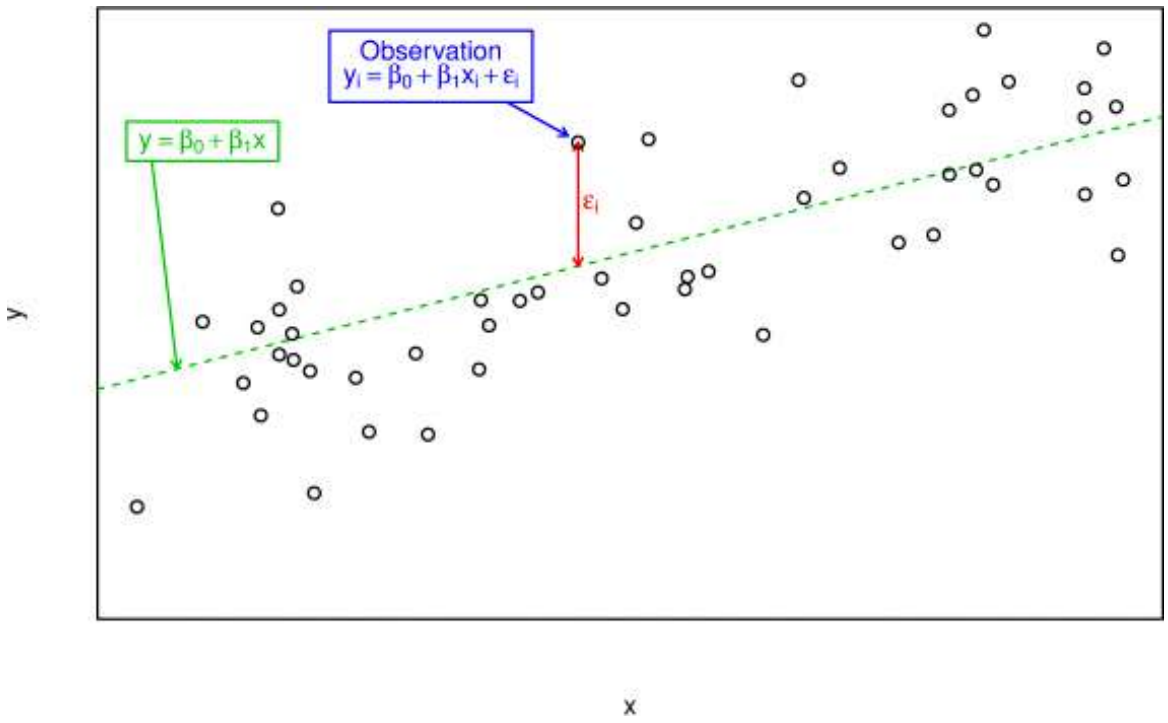

Рисунок 1.7 – Приклад даних з моделі лінійної регресії. [6]

Можна використовувати залишки для перевірки наступних припущень, що лежать в основі лінійної регресії:

- Між y_i і x_i існує лінійне співвідношення: для будь-яких пар дані повинні апроксимувати пряму лінію. Якщо нанести на двовимірний графік залишки, то повинно спостерігатись випадкове розсіювання точок, а не яка-небудь систематична картина;
- Залишки нормально розподілені з нульовим середнім значенням;

- Залишки мають одну і ту ж варіабельність (постійну дисперсію) для всіх передбачених величин y.

Якщо припущення лінійності, нормальності і / або постійної дисперсії сумнівні, модель має бути перебудована за допомогу будь-якого іншого методу.

Можна застосовувати лінію регресії для прогнозування значення за значенням в межі спостережуваного діапазону.

Ми робимо прогноз залежної величини y шляхом підстановки в рівняння регресії спостереженого значення x. Використовуємо цю передбачену величину і її стандартну похибку, щоб оцінити довірчий інтервал для істинної середньої величини в популяції. Наприклад, межі довірчого інтервалу, що відповідає 95% можна оцінити за такою формулою:

$$\hat{y} \pm 1.96 s_e \sqrt{1 + \frac{1}{N} + \frac{x - \bar{x}^2}{(N-1)s_x^2}}; \tag{1.19}$$

де N - кількість спостережень, s_e - стандартна похибка, s_x – стандартне відхилення спостережуваних значень x. Прогноз разом із довірчим інтервалом зображений на рисунку 1.8.

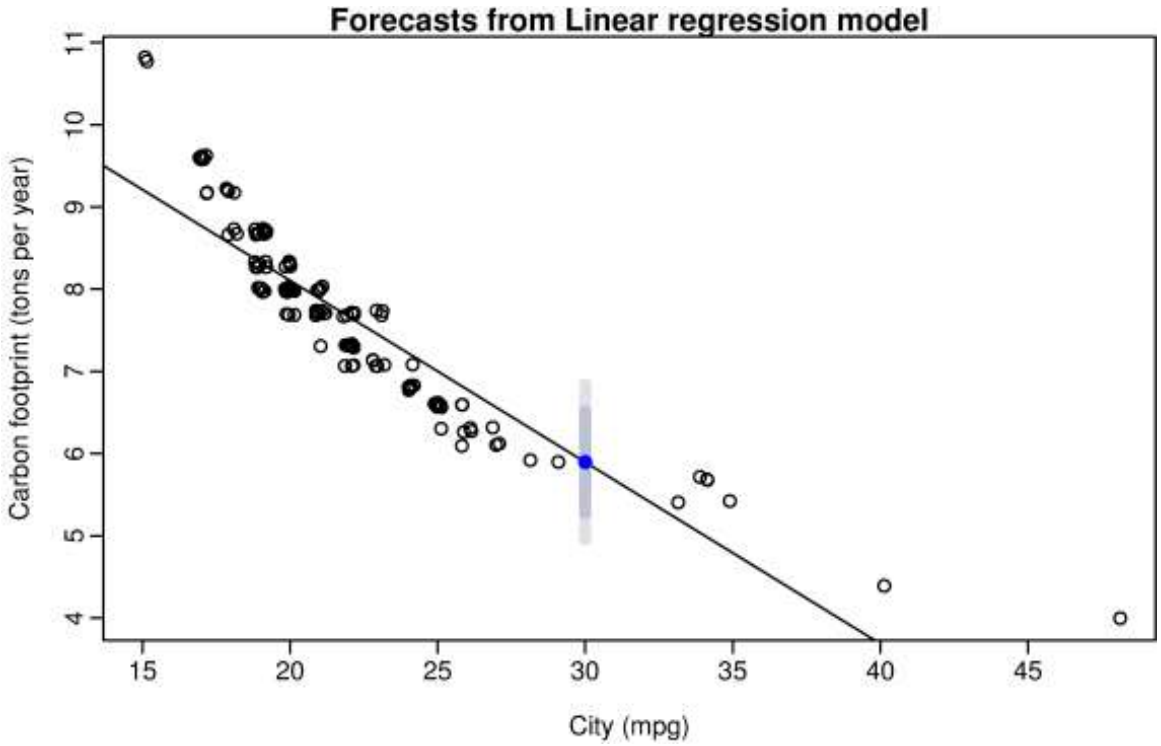

Рисунок 1.8 – Прогноз із 95-ти та 80-ти відсотковими довірчими інтервалами для залежності викидів карбону від економії автомобілями палива. [6]

Повторення цієї процедури для різних величин x дозволяє побудувати довірчі кордони для цієї лінії. Це смуга або область, яка містить справжню лінію-прогноз, наприклад, з 95% довірчою ймовірністю.

Подібним чином можна розрахувати ширшу область, усередині якої, як ми очікуємо, лежить найбільша кількість (зазвичай 95%) спостережень.

Побудова регресійної моделі

Найбільш простим методом визначення коефіцієнтів рівняння лінійної регресії є метод найменших квадратів (МНК). Метод найменших квадратів виконує вибір коефіцієнтів шляхом мінімізації суми квадратів залишків (вертикальна відстань від кожної точки до лінії – див. рис. 1.9).

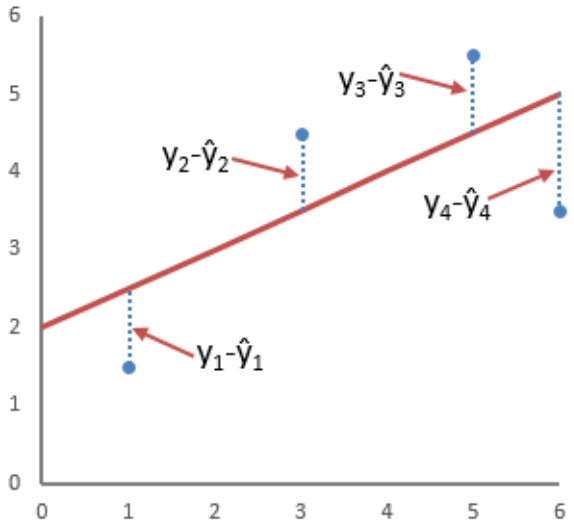

Рисунок 1.9 – Визначення найменших відстаней від точок до лінії. [8]

1.2.6 Порівняння моделей прогнозування

Прогнозуючі моделі мають безліч характеристик, які можна використовувати для їх порівняння між собою з метою вибору найбільш зручної для використання для вирішення конкретної задачі. У таблиці 1.2 представлені основні моделі для 2 методів прогнозування з їх порівнянням за точністю прогнозів на різні терміни, приблизною вартістю проектування та побудови прогнозу на основі моделі, строком, потрібним для побудови моделі та прогнозу.

Таблиця 1.2 – Порівняння деяких прогнозуючих моделей [9]

Техніка		Аналіз часових рядів				Причинно-наслідкові моделі	
		Ковзне середнє	Експоненційне згладжування	Box-Jenkins (AR(I)MA)	X-11	Регресійні моделі	Економетричні моделі
Точність	Короткий строк	Погано-добре	Точно-дуже добре	Дуже точно-чудово	Дуже точно-чудово	Добре-дуже добре	Добре-дуже добре
	Середній строк	Погано	Погано-добре	Погано-добре	Добре	Добре-дуже добре	Дуже добре-погано
	Довгий строк	Дуже погано	Погано	Погано	Погано	Погано	Добре
Вартість (у доларах США)	З використанням ЕОМ	0.005	0.005	10	10	100	5000
	Можливість ручного розрахунку	так	так	так	ні	так	так
Час		1 день	1 день	1-2 дні	1 день	Залежить від можливості визначити зв’язки	2 місяці +

1.3 Висновок

У розділі було проведено огляд основних методів та технологій, які підлягають дослідженню в роботі: Інтернет речей та моделей прогнозування.

Було визначено основні складові систем , що базуються на технології Інтернет речей, також типи таких систем в залежності від складових елементів. Було розглянуто основні проблеми, які виникають у таких системах.

У ході опрацювання літератури, яка стосується методів та способів прогнозування було визначено, що таке прогноз та прогнозування, в чому полягає задача прогнозування. Також визначено основні типи прогнозів та чинники, які впливають на вибір методів та моделей для прогнозування. Серед них такі:

- Кількість даних, які доступні для тренування моделі;
- Якість даних;
- Тип прогнозу за часовими характеристиками;
- Наявність чітко означених предикторів (змінних, від яких залежить показник, який має бути прогнозований);

Також було визначено основні типи прогнозуючих моделей:

- Моделі, що базуються на часовому аналізі;
- Причинно-наслідкові моделі;
- Кількісні методи;

Було розглянуто переваги та недоліки кожної з моделей і визначено типові ситуації, в яких краще застосовувати ту чи іншу модель. Результати порівняння були зведені в таблицю 1.2.

2 ВИБІР ПРОГНОЗУЮЧОЇ МОДЕЛІ

2.1 Предмет прогнозування

Часто визначення предмету, поведінка якого має бути прогнозованою є найбільш важкою частиною прогнозування. Визначення проблеми ретельно вимагає розуміння того, як будуть використовуватися прогнози, для яких користувачів призначені прогнози, і яким є загальний вплив прогнозування на стан прогнозованого предмету (системи, явища тощо). Прогнозист повинен виявити вимоги всіх без винятку осіб, які мають безпосередній інтерес у постановці прогнозу та побудові коректної моделі.

В нашому випадку глобальним предметом прогнозу є виявлення урожайності певної культури на поточний або планований посівний сезон. Існує декілька показників урожайності, які використовуються в агрономічному плануванні, обліку та економічному аналізі.

- Потенційна врожайність – максимальна кількість продукції, яку можна отримати з 1 га земельних угідь при повній реалізації продуктивних можливостей сільськогосподарської культури (або сорту). Обчислюється (стосовно до ідеальних і звичайних умов) головним чином сільськогосподарськими науково-дослідними і дослідницькими установами. Показник використовують для визначення раціональної структури землеробських галузей, набору сортів і сільськогосподарських культур в господарстві, області, зоні.
- Планова врожайність - кількість продукції, яку можна отримати з 1 га земельних угідь в конкретних господарських умовах. Визначається до посіву з урахуванням потенційних можливостей сорту, досягнутого рівня урожайності в минулому, родючості ґрунту, забезпеченості господарства технікою, мінеральними добривами і т.д. Планова врожайність - показник виробничо-фінансового плану сільськогосподарського підприємства, що використовується в управлінні сільськогосподарським виробництвом.

- Очікувана врожайність (сподівання на врожай) - передбачуваний збір продукції. Визначається в центнерах з 1 га земельних угідь або умовно (висока, середня, низька, на рівні минулого року) в окремі періоди зростання і розвитку сільськогосподарських культур (по густині стеблостою і загальному стану рослин). Показник використовують для планування агротехнічних заходів.
- Врожайність на корені (біологічна врожайність) - кількість вирощеної продукції. Встановлюється вибірково наступними методами: оцінка на око, методом взяття проб (до збирання врожаю) або розрахунково-балансовим (після збирання - за даними про фактичний намолот і втрати в процесі збирання). Показник використовують в економічному аналізі для пошуку резервів зниження втрат врожаю при зборі.
- Фактичний збір з 1 га земельних угідь - зібрана і врахована продукція. Визначається різними способами: у початково оприбуткованій або чистій (після обробки) вазі в розрахунку на 1 га посівної, весняної продуктивної або фактично зібраної площі (в залежності від сільськогосподарської культури). Враховується сільськогосподарськими підприємствами в два терміни: попередньо - за оперативними даними про хід збирання, і остаточно - за даними бухгалтерського обліку (показник відбивається в статистичних довідниках та інших джерелах і характеризує розвиток землеробських галузей). [10]

Таким чином предметом оцінки прогнозуючої моделі можна вважати показник очікуваної урожайності у одиницях центнер/га без врахування таких показників, як родючість ґрунту та інших показників, які є індивідуальними для сільськогосподарських господарств.

2.2 Вхідні дані

Завжди є принаймні два види інформації, необхідної для проектування прогнозуючої моделі:

1. статистичні дані;
2. накопичений досвід людей, які збирають дані і використовують прогнози.

Часто отримати достатню кількість історичних даних, щоб побудувати достатньо гарну статистичну модель, дуже складно. Хоча в деяких випадках занадто застарілі дані є менш корисними через те, що вони не відображають зміни, що стались в прогнозованій моделі. Тож потрібно визначити, який вплив має минуле на прогнозоване майбутнє.

У визначенні показника врожайності, а саме така задача перед нами стоїть, можна знехтувати даними, давність яких складає 10 або більше років. Це зумовлено тими показниками, які ми не враховуємо безпосередньо в моделі, такими як розвиток сільськогосподарської техніки чи зниження чи підвищення урожайності ґрунту, так як їхній вплив на систему стрімко падає з плином часу. Наприклад, збиральна техніка протягом останніх 2 десятиліть здатна підтримувати досить незначний рівень втрат урожаю при збиранні. А родючість ґрунтів досить давно і успішно контролюється шляхом грамотної політики внесення добрив та календарного планування. Саме на такі техніки слід звернути увагу.

Найбільш впливовим фактором, що відображається на стані врожаю, є погодний стан. Світло — джерело електромагнітних хвиль, які збуджують хлорофіл і забезпечують фотосинтез. Світловий режим відіграє вирішальну роль у процесах росту, розвитку і формування врожаю та його якості. Світло впливає на процеси росту. При недостатньому освітленні збільшується швидкість лінійного росту, зменшується механічна міцність стебла і трав’янисті рослини вилягають. Прискорює процеси розтягування клітин і спричиняє збільшення лінійних розмірів стебла інфрачервоне випромінювання, яке переважає в хмарну погоду. Світло значно впливає на процес розвитку рослин. Строки цвітіння і плодоношення багатьох рослин можна змінювати, регулюючи тривалість дня.

Різна вимогливість рослин до тепла виявляється вже при проростанні насіння і спостерігається протягом їх життя. Для кожної фази розвитку і росту

рослини існують мінімальні, оптимальні і максимальні температури. За вимогливістю до тепла польові культури умовно можна поділити на холодостійкі (культури ранніх строків посіву), середньохолодостійкі (культури середніх строків посіву) і теплолюбні (культури пізніх строків посіву).

Ріст і розвиток рослин залежать від температурного режиму ґрунту. Зниження температури ґрунту нижче +10°С негативно впливає на надходження мінеральних елементів живлення в корені (насамперед азоту, потім фосфору і кальцію і меншою мірою калію). Температура впливає на мікробіологічну діяльність у ґрунті, від якої в значній мірі залежить його родючість. Оптимальна температура для життєдіяльності ґрунтової мікрофлори +15—+20°С.

У більшості зелених і свіжозібраних рослин міститься 75—90% води. Рослинна клітина повинна бути постійно насичена водою. З нею в рослину надходять і переміщуються в ній поживні речовини. Вода бере участь в утворенні поживних речовин, фотосинтезі, завдяки їй підтримується постійна температура в рослині, попереджається перегрів її сонцем.

Крім вуглецю, кисню і водню, до складу рослин входить близько 70 хімічних елементів. Більшість із них рослини вбирають з ґрунту.

Правильний обробіток ґрунту в сівозмінах, боротьба з бур’янами значно поліпшують поживний режим ґрунту. Найбільш ефективним заходом регулювання поживного режиму ґрунту є біологічно обґрунтоване внесення органічних і мінеральних добрив. [11]

Таким чином можна визначити, що найголовнішими факторами, які впливають на показники врожайності, є кількість світла, температура повітря, температура ґрунту, кількість води та мінеральних речовин у ґрунті.

Деякі з цих показників не можуть контролюватися людиною, якщо мова йде про відкриті площі: температура та кількість світла. Деякі показники можуть контролюватися, але зазвичай контроль зводиться лише до мінімального задоволення потреб посівів, це – кількість води, яку отримують рослини. До того ж цей показник великою мірою залежить від вище зазначеної

температури. А такий показник, як хімічний склад ґрунту, залежить більшою мірою від людини і може цілком контролюватися нею. **Тому доцільно виділити в якості параметрів моделі температуру та вологість.** Так як дані про хімічний склад ґрунту за довгий період часу знайти доволі тяжко, то можна обмежитись **даними про терміни внесення добрив.**

2.3 Визначення типу прогнозу

Так як предметом прогнозу є показник очікуваної врожайності, то нас цікавить прогноз однієї сівозміни від її початку, або з будь-якого моменту її тривалості, до кінця, тобто, збору. **Такий прогноз можна віднести до прогнозів середнього терміну.**

2.4 Вибір прогнозуючих методів

2.4.1 Прогнозування температури та вологості.

Для прогнозування цих показників нам достатньо мати історичні дані про стан цих показників. Таким чином для прогнозування на період до 1 року ми можемо застосувати будь-який з методів прогнозування на основі часових рядів, які дають потрібну точність для такого періоду часу у враховують фактори сезонності, так як погодні умови містять у собі елементи сезонності. Модель має бути також достатньо точною, щоб відокремлювати аномальні випадки, як от сніг у травні.

Алгоритм Бокса – Дженкінса дозволяє виконувати достатньо точний короткочасний прогноз. Але необхідно відмітити, що не існує простого способу корекції параметрів ARIMA моделі для нових даних. Модель потрібно періодично повністю перебудовувати або вибирати зовсім нову модель.

2.4.2 Прогнозування плану внесення добрив.

Зробивши прогноз температурних показників, можемо зробити прогноз щодо планування внесення добрив. План внесення добрив не може бути остаточно визначений лише на основі історичних даних. Погодні чинники

також мають стійких вплив на терміни польових робіт. Прогнозування плану внесення добрив направлене на зменшення щодо нього невизначеності, що дасть агроному можливість вчасно оцінити ресурси і підготуватися до проведення робіт.

Для прогнозування в цьому випадку доцільно обрати модель, що враховує причинно-наслідкові зв’язки між елементами системи. Стан системи в минулий момент часу може також бути визначений як один із факторів впливу. В такому випадку для прогнозування оберемо модель множинної регресії, яка дає гарні результати як на короткострокових прогнозах, так і на прогнозах середньої тривалості.

В якості предикторів моделі оберемо значення температури та вологості.

Для даного типу прогнозування, прогнозування абсолютних значень – так або ні (0 чи 1), треба застосовувати до прогнозу нормалізацію значень, так як в реальних умовах отримати значення 0 та 1 майже неможливо.

Нормалізація складатиметься з двох етапів:

- Масштабування раду отриманих значень щодо найбільшого
- Дискретизація значень із рівнем 0.5

2.4.3 Прогнозування показника врожайності.

Так як показник врожайності чітко можна поставити в залежність від температурних даних впродовж сівозміни і плану внесення добрив (якість добрив та правильність їх хімічного складу покладемо на плечі агронома), то для прогнозування також можна побудувати множинну регресійну модель. В якості предикторів будемо використовувати значення температури, вологості та відомості про план внесення добрив. Значення показника вимірюється і обчислюється у форматі центнер/га.

2.5 Визначення зв’язків між показниками

У попередньому розділі було визначено методи прогнозування кожного та дані, які потрібні для побудови кожного рівня моделі. Також визначено, що

показники є деякою мірою взаємозалежними. Так прогнозування плану внесення добрив потребує даних про показники температури та вологості. А прогнозування показника врожайності потребує поряд з температурними показниками також прогноз плану внесення добрив. Взаємозв'язки між вищезазначеними факторами схематично відображено на рис. 2.1.

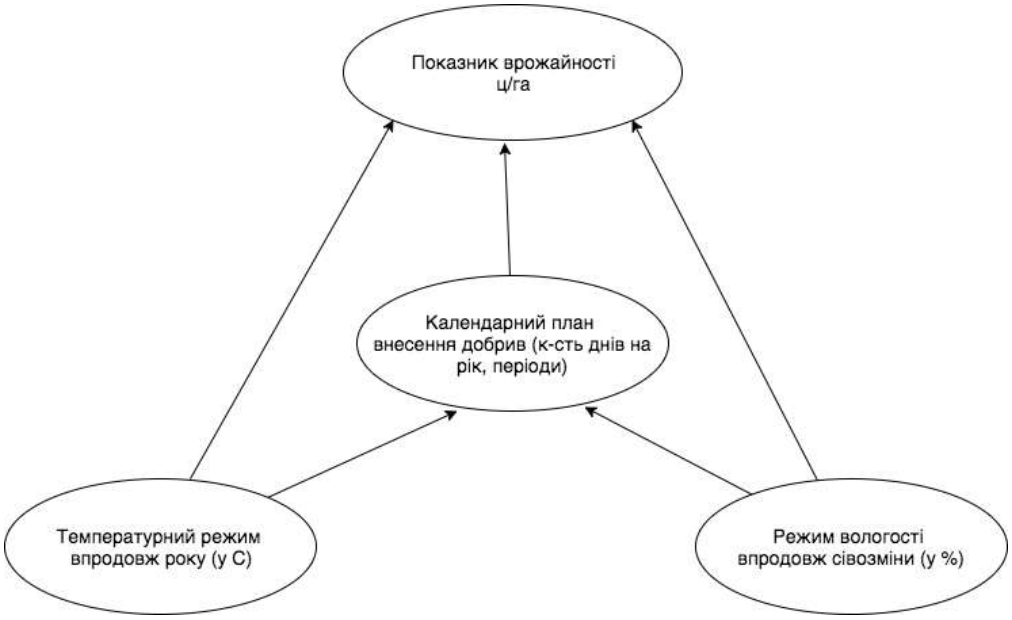

Рисунок 2.1 – Взаємозв'язки між вхідними даними та прогнозованим показником

З рисунка видно, що фактори не лише спільно впливають на основний прогнозуючий показник, а ще й взаємопов'язані між собою. Причому характер такий, що календарний план внесення добрив не може бути витіснений температурними показниками, чи показниками вологості, так як крім цих показників на нього впливає людський фактор.

2.6 Вибір моделі

Наявність взаємозв'язків між етапами прогнозування дає нам можливість використати каскадну модель, в якій результат кожного з етапів подається на вхід наступного етапу. Враховуючи моделі, обрані для кожного етапу у розділі 2.4 та зв'язки між показниками, встановлені у розділі 2.5, можемо обрати

прогнозуючу модель. Її характер та ступені обробки даних зображені на діаграмі потоків даних (рис 2.2).

Такий ітеративний характер моделі дає нам змогу відстежувати помилки прогнозування на кожному з етапів окремо. Модель дозволяє використовувати кожен з етапів прогнозування окремо для інших цілей, наприклад, аналітичного прийняття рішень сторонньою особою чи механізмом. Також з'являється можливість редагувати дані за необхідності для кожного з етапів. Це означає більшу гнучкість у використанні, а отже й оптимальність.

2.7 Висновок

У розділі було виділено основний предмет прогнозування – очікувана врожайність на поточний посівний період у вигляді формалізованого значення – центнер/га.

Також було виконано аналіз основних чинників, які потенційно впливають на предмет прогнозування з метою виділення декількох основних, значення яких можна використати для побудови причинно-наслідкової моделі для прогнозування показника врожайності.

Були визначені способи та методи для прогнозування основних незалежних відносно предмета прогнозування показників, визначено обмеження для побудови моделей прогнозування.

Було обрано модель для прогнозування показника врожайності. Результатом є ітеративний процес, кожен наступний крок якого залежить від попередніх і водночас може бути використаний самостійно без подальшого застосування в моделі.

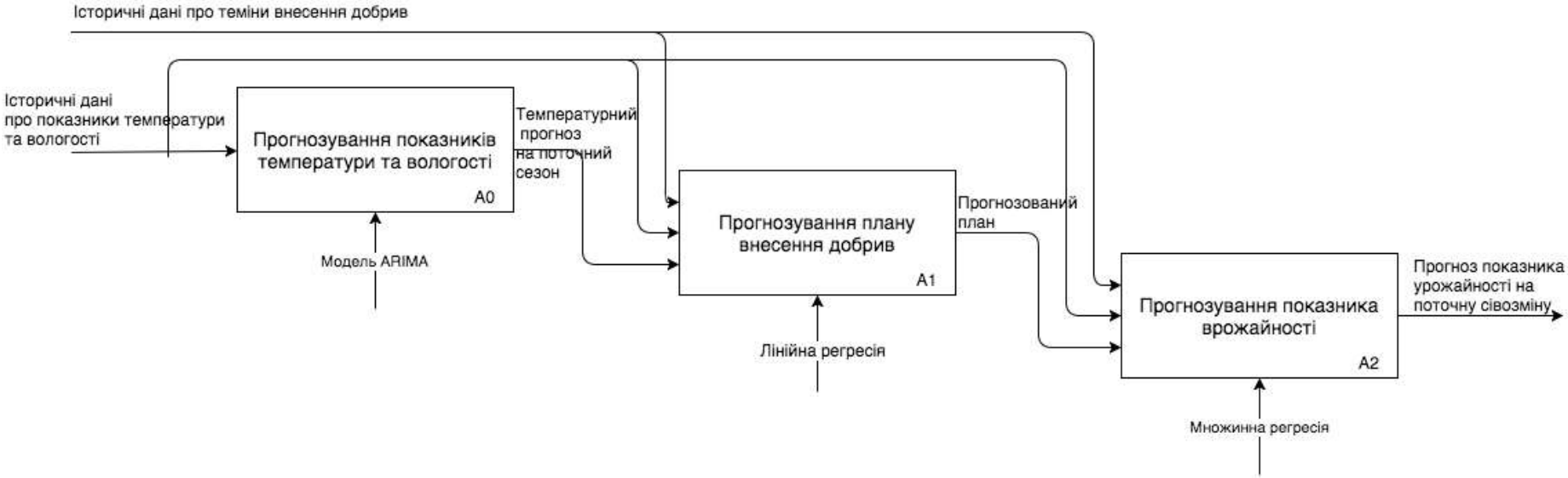

Рисунок 2.2 – Діаграма потоків даних і процесів прогнозуючої моделі

3 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ СИСТЕМИ. АСПЕКТИ РЕАЛІЗАЦІЇ

3.1 Вимоги до системи

У даній роботі маємо побудувати макет системи, яка виконуватиме такі завдання:

- Збір певних типів інформації з навколишнього середовища з певною періодичністю, а саме – температурні показники, показники вологості повітря та ґрунту.
- Попередня обробка даних у пристрої: приведення показань до загальноприйнятих шкал (градуси Цельсія, Фаренгейта, вологість у відсотках).
- Передача даних на сервер через мережу Інтернет – передача у зручному форматі через HTTP з’єднання.
- Прийом даних на сервері – виділений канал для прийому повідомлень від пристроїв.
- Інструменти керування станом пристроїв для користувача, зручне відображення даних від пристроїв.
- Можливість підключення до сторонніх API для отримання історичних даних про показники стану середовища.
- Можливість завантаження користувачем даних про показники врожайності та календарні плани внесення добрив.
- Аналіз отриманих даних із можливістю отримання результату прогнозування за умови наявності потрібних для системи даних.
- Можливість отримати результат роботи на всіх рівнях аналізу за умови наявності потрібних даних.
- Зручний користувацький інтерфейс для управління даними користувача, конфіденційність інформації.

Маємо набір функціональних вимог до макету системи. Серед нефункціональних вимог можна виділити такі: відмовостійкість, масштабованість та продуктивність.

3.2 Проектування системи

Типову архітектуру системи зображено на рис. 3.1.

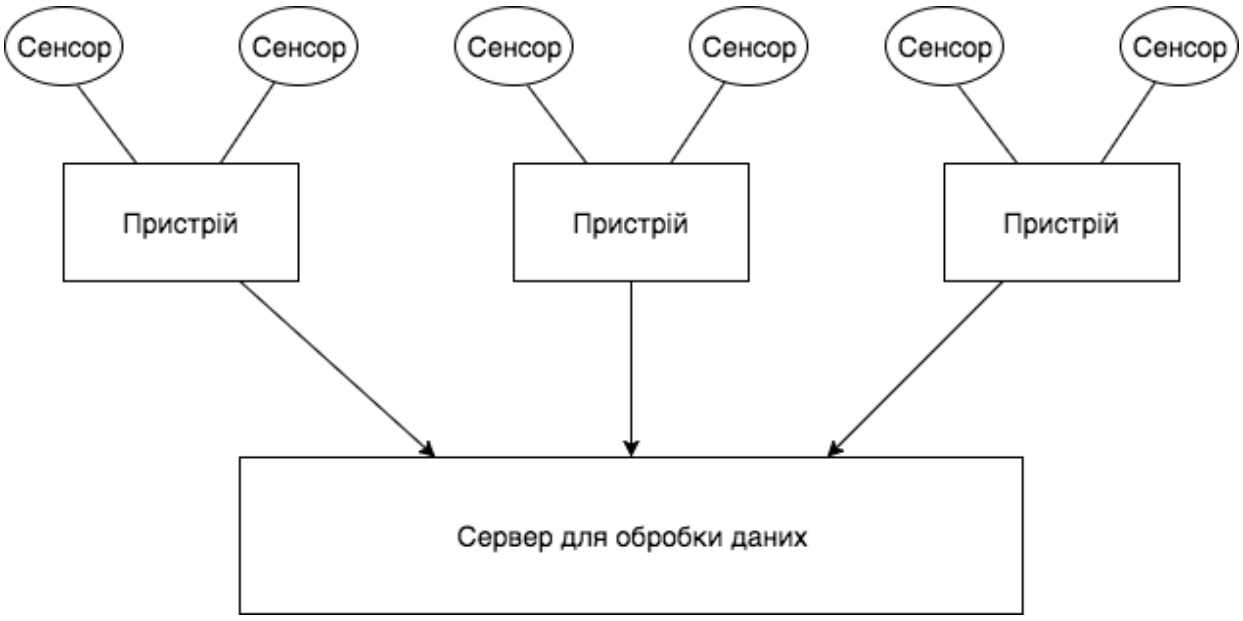

Рисунок 3.1 – Архітектура систем на базі технології IoT

Певна кількість пристроїв різної конфігурації збирає інформацію з середовища з допомогою сенсорів, з якими вони пов’язані різноманітними інтерфейсами, та передає їх на платформу для здійснення подальших маніпуляцій.

Таким чином проектування та розробку системи можна розділити на два етапи: проектування та розробку пристрою із врахуванням його інтерфейсів взаємодії з сенсорами і проектування та розробку сервера з врахуванням розробки обраної моделі прогнозування та всіх вимог, поставлених до нього.

3.2.1 Проектування пристрою

Зробимо огляд основних компонентів пристрою, можливості їх використання у контексті системи із задоволенням встановлених у підрозділі 3.1 вимог.

3.2.1.1 Raspberry Pi

Raspberry Pi -- мікрокомп'ютер, який містить мінімально необхідну кількість компонентів, які забезпечують його функціонування. Ініціатором розробки мікрокомп’ютера був британський благодійний фонд Raspberry Pi який мав на меті надання недорогих комп'ютерів і безкоштовного програмного забезпечення для студентів та учнів. Їх кінцева мета полягає у сприянні комп'ютерної природничо-наукової освіти, і вони сподіваються, що цей маленький доступний комп'ютер буде інструментом, який дозволить проводити різноманітні наукові експерименти.

Основою мікрокомп’ютера є друкована плата (так звана система на чипі (SoC) Broadcom BCM2835). Система включає в себе 32-бітний ARM1176JZFS процесор, що працює на 13 частоті 700 МГц. Вона має 256 МБ оперативної пам'яті в пакеті POP. Raspberry Pi отримує енергію від зарядного пристрою Micro USB 5V AC або принаймні 4-х батарейок типу AA. У той час як ARM процесор забезпечує реальну продуктивність, аналогічну з 300MHz Pentium 2, Broadcom GPU є потужним графічним ядром, здатним до апаратного декодування декількох форматів відео високої чіткості.

В залежності від специфікації (A чи B) наявна різна кількість інтерфейсів підключення зовнішніх пристроїв, а також різна кількість портів GPIO. Модель A - без інтерфейсу підключення Ethernet і має один порт USB. Модель B підтримує всі наявні інтерфейси.

В даній роботі була використана модель B - вона оснащена HDMI входом і композитним відеовиходом, має два порти USB 2.0, порт 10/100 Ethernet, слот

для SD-карти, GPIO (General Purpose I / O) роз'єм і аналоговий аудіо вихід (3,5 мм роз'єм для навушників). Дешевша Модель А не має порту Ethernet і один лише один USB вхід, але має такі ж конфігурації процесора.

Однією з особливостей Raspberry Pi є ряд GPIO пінів (пінів вводу / виводу загального призначення) - шпильки уздовж краю плати. Схема та призначення пінів зазначено на рис. 3.2.

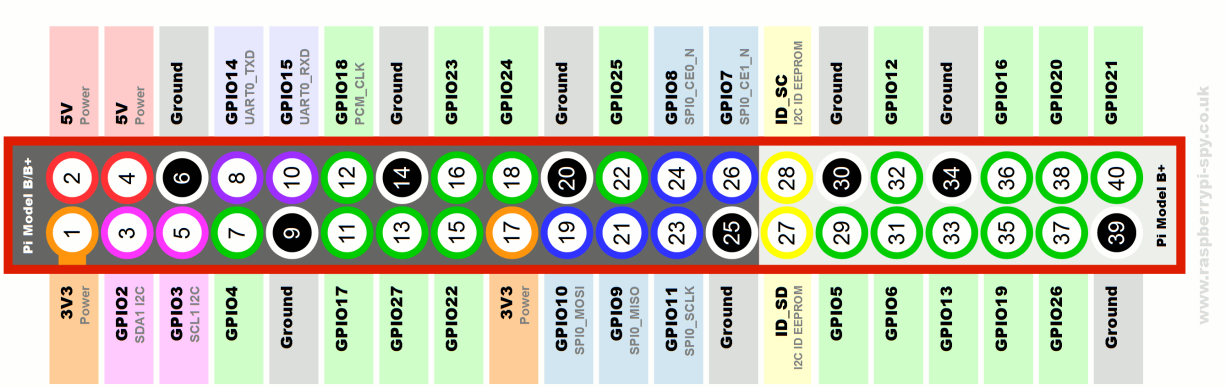

Рисунок 3.2 – Карта GPIO-пінів Raspberry Pi Model B. [12]

3.2.1.2 Сенсор DHT22

Цей сенсор температури повітря дозволяє отримувати дані підвищеної точності у цифровому вигляді.

Сенсор має такі характеристики:

- Напруга живлення від 3 до 5 В;
- Максимально можливе використання струму на рівні 2.5мА при перетвореннях (запитах даних);
- Розрахований на вимірювання вологості у діапазоні від 0% до 100%. При цьому точність коливається у діапазоні 2% - 5%;
- Вимірює температуру у діапазоні від -40 до 125 градусів за Цельсієм із точністю $\pm$0.5 градусів;
- Частота вимірювань до 0.5 Гц (1 вимірювання за 2 секунди);
- Розмір корпуса: 15.1 мм x 25 мм x 7.7 мм;

- Чотири конектори.

Електричне коло для підключення сенсору до GPIO-піну включає в себе такі обов’язкові компоненти: живлення 3-5 В, резистор із опором на рівні 5-10КОм та сенсор. Схема підключення зображена на рис. 3.3.

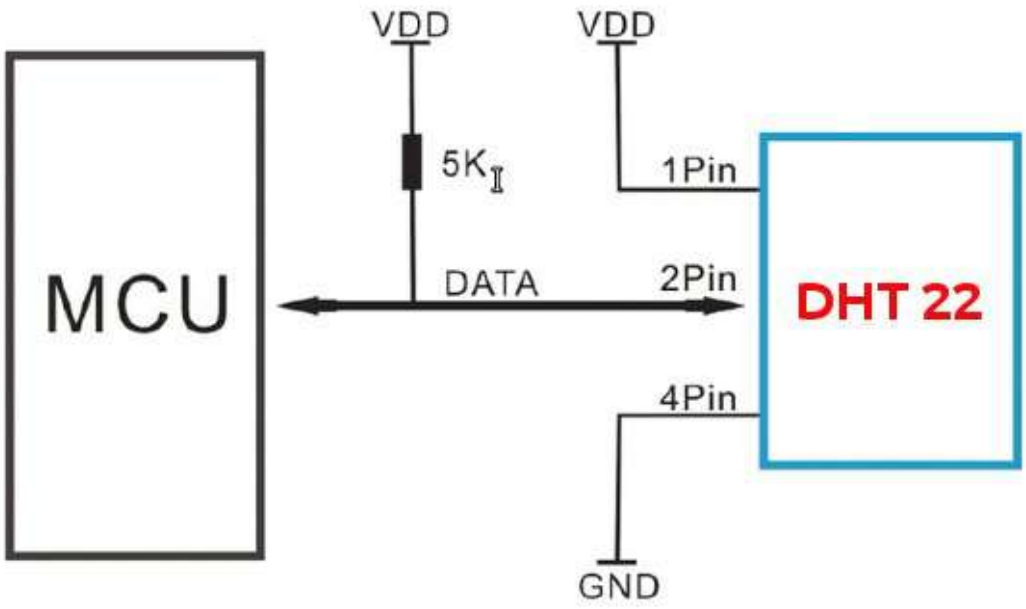

Рисунок 3.3 – Схема підключення DHT22 до мікрокомп'ютера. [12]

Для використання сенсору у програмах, написаних на Python [13], використаємо Python версію бібліотеки pigpio [14], яка призначена для управління GPIO-пінами. Встановлення бібліотеки потребує таких кроків:

```
wget abyz.co.uk/rpi/pigpio/pigpio.zip
unzip pigpio.zip
cd PIGPIO
make
sudo make install
```

Виконуємо завантаження архіву з бібліотекою, після розархівування виконуємо встановлення. Далі маємо встановити модуль, призначений саме для DHT22 сенсора:

```
wget abyz.co.uk/rpi/pigpio/code/DHT22_py.zip
```

Перед запуском програми, яка використовує цю бібліотеку, потрібно запускати спеціальний процес командою `sudo pigpiod`, який надає права доступу до пінів, які за замовчуванням вимагають `sudo` прав.

Розробимо функцію, яка виконуватиме зчитування показників температури та вологості:

```
import pigpio
import DHT22
pi = pigpio.pi()
s = DHT22.sensor(pi, 17)
s.trigger()
sleep(.01) # Necessary on faster Raspberry Pi's

def readDHT22():
        s.trigger()
        hum = '%.2f' % (s.humidity())
        temp = '%.2f' % (s.temperature())
        return (hum, temp)
```

Скрипт із даною функцією може бути включений в програму вищого рівня, яка виконуватиме збір показників з більшої кількості сенсорів та відправку даних, в якості модуля.

3.2.1.3 Сенсор DS18B20

Цей датчик температури базується на популярній мікросхемі DS18B20. Він дозволяє визначити температуру навколишнього середовища в діапазоні від -55 ° C до + 125 ° C і отримувати дані в вигляді цифрового сигналу з 12-бітовою розрядністю по 1-Wire протоколу. Цей протокол дозволяє підключити величезну кількість таких датчиків, використовуючи всього 1 цифровий порт контролера, і всього 2 дроти для всіх датчиків: землі і живлення. У цьому випадку застосовується так зване паразитне живлення, при якому датчик отримує енергію прямо з лінії сигналу, якщо на ній гарантовано протягом довгого часу наявний сигнал високого рівня. Пристрій, що отримує таким чином живлення, може деякий час заряджатись від лінії, а потім передавати чи отримувати інформацію по цій же лінії. За таким принципом також живиться наприклад комп’ютерна мишка.

Кожен датчик має унікальний прошитий на виробництві 64-бітний код, який може використовуватися мікроконтролером для спілкування з конкретним сенсором на загальній шині. Код окремого сенсора може бути отриманий за допомогою окремої команди.

В постійній пам'яті DS18B20 можна зберегти граничні значення температури, при виході з яких сенсор буде переходити в режим тривоги. На загальній шині з багатьох сенсорів мікроконтролер може за раз дізнатися, які з них перейшли в цей режим. Таким чином стає легко визначити проблемну ділянку в контрольованому середовищі.

Розрядність показань налаштовується і може становити від 9 до 12 біт. Схема підключення датчика до мікрокомп'ютера Raspberry Pi показана на рис. 3.4. Підключення потребує також наявність у схемі резистора опором 5-10 КОм.

За замовчуванням підтримка інтерфейсу 1-Wire в Raspberry вимкнена. Щоб дозволити використання інтерфейсу треба додати в /boot/config.txt строчку `dtoverlay=w1-gpio.` Після перезавантаження мікрокомп'ютера інтерфейс має працювати.

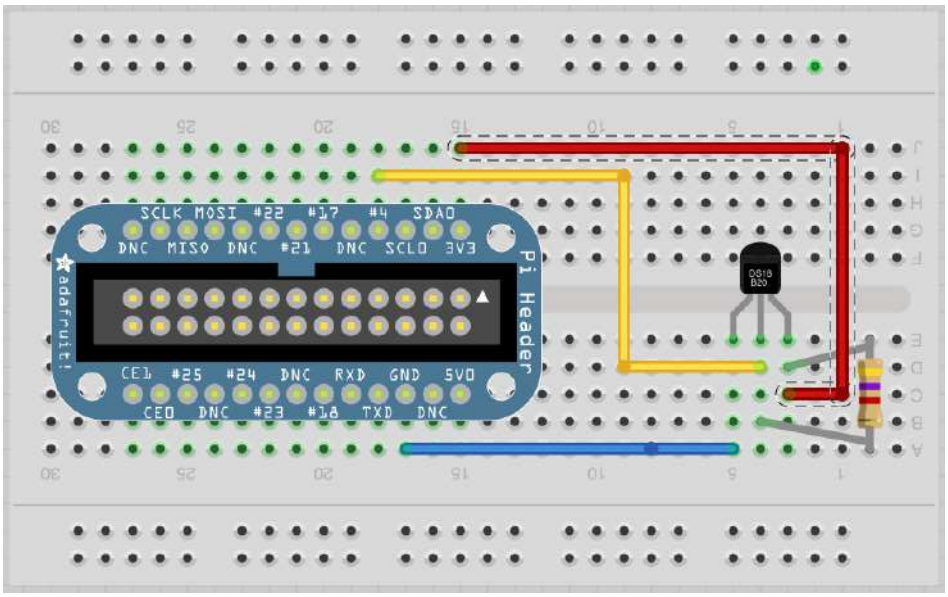

Рисунок 3.4 – Схема підключення сенсора DS18B20 до Raspberry Pi через GPIO піни. [15]

Одноразово зчитати дані з сенсора можна, використавши такий набір команд:

```
sudo modprobe w1-gpio
sudo modprobe w1-therm
cd /sys/bus/w1/devices
ls
cd 28-xxxx (change this to match what serial number pops up)
cat w1_slave
```

На місці хрестиків у командах, наведених вище буде ідентифікаційний номер датчика. Приклад зчитування даних наведений на рис. 3.5.

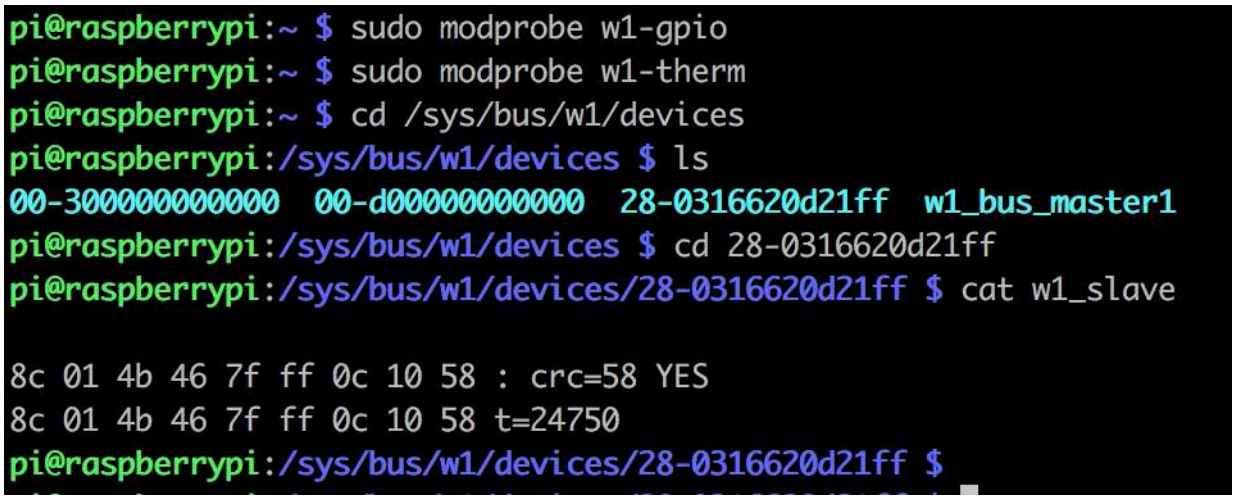

Рисунок 3.5 – Приклад зчитування даних з датчика DS18B20

Ідентифікаційний номер мого сенсора 0316620d21ff. Символ YES показує, що температура може бути зчитана коректно, а числове значення в наступній строчці показує значення температури разом з трьома знаками після коми, тобто помножене на 10^3. Дізнаємося, що температура в момент часу становить 24.750 градусів за Цельсієм.

Розробимо модуль, призначений для роботи з сенсором DS18B20:

```
import os
import glob

os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')
```

```
base_dir = '/sys/bus/w1/devices/'
device_folder = glob.glob(base_dir + '28*')[0]
device_file = device_folder + '/w1_slave'

def read_temp_raw():
    f = open(device_file, 'r')
    lines = f.readlines()
    f.close()
    return lines

def read_temp():
    lines = read_temp_raw()
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = read_temp_raw()
    equals_pos = lines[1].find('t=')
    if equals_pos != -1:
        temp_string = lines[1][equals_pos+2:]
        temp_c = float(temp_string) / 1000.0
        temp_f = temp_c * 9.0 / 5.0 + 32.0
        return '%.2f' % temp_c
```

Як бачимо, функції модуля виконують ту ж роботу, що й команди в консолі, але ще й форматують отримане значення.

Датчик DS18B20 є водонепроникним, тож будемо використовувати його для вимірювання температури ґрунту.

Роботу програми по збору інформації від сенсорів наведено на рис. 3.6. Запити до сенсорів робляться через певні проміжки часу, результат виводиться в заданих одиницях в консоль.

```
pi@raspberrypi:~/soil_observer $ sudo pigpiod
pi@raspberrypi:~/soil_observer $ python main.py
Humidity: 42.30%
Temperature: 24.30C
Soil Temperature: 25.44C
Humidity: 42.30%
Temperature: 24.30C
Soil Temperature: 25.50C
Humidity: 45.50%
Temperature: 24.80C
Soil Temperature: 25.50C
```

Рисунок 3.6 – Приклад роботи скрипта, який звертається до сенсорів через задані часові проміжки

3.2.1.4 3G модуль

Для підключення до мережі Інтернет та встановлення незалежного автоматичного підключення із перезапуском після виникнення проблем було обрано модуль Vodafone Connect 4100. Вибір був зроблений на користь цього рішення, так як модуль може бути підключений через USB-роз'єм та не потребує спаювання, що значно полегшує виконання завдання та мінімізує можливість пошкодити мікрокомп'ютер.

Після підключення модема потрібно налаштувати пристрій так, щоб модем розпізнавався в якості модему. Натомість спочатку він розпізнається як дисковий пристрій. Для цього налаштування потрібно встановити додаткову бібліотеку usb-modeswitch. Для запобігання рецидиву такої проблеми, яка може виникати при автоматичному вимиканні-вмиканні пристрою створюємо конфігураційний файл, який буде кожного разу виконуватись утилітою. Файл має назву /etc/udev/rules.d/41-usb_modeswitch.rules та містить такий контент:

```
ATTRS{id_Виробника}=="12d1",ATTRS{id_Продукту}=="155b",RUN+="u
sb_modeswitch '%b/%k'"
```

Для налаштування модему використаємо бібліотеку Sakis3G. Після запуску утиліти в інтерактивному режимі командою `sudo ./sakis3g –`

`interactive` у графічному меню виконаємо налаштування, серед яких вибір характеру з'єднання, порту з'єднання та параметри APN (точки доступу).

Налаштування перепідключення після втрати з'єднання потребує встановлення утиліти umtskeeper. Встановку виконуємо таким набором консольних команд:

```
wget http://zool33.uni-graz. at/petz/umtskeeper/src/umtskeeper.tar.gz
tar -xzvf umtskeeper.tar.gz
chmod +x umtskeeper
```

Після запуску утиліти з парамертами для зазначення номеру підключення, ідентифікаційного номеру модему а також параметрів точки доступу перепідключення має бути налаштоване.

Для налаштування встановлення підключення при кожному запуску пристрою потрібно внести команду з викликом утиліти umtskeeper разом з параметрами у файл /etc/rc.local.

3.2.1.5 Обмін інформацією з сервером

Для HTTP-запитів будемо використовувати бібліотеку requests. Функція, яка відправляє дані може бути реалізована таким чином:

```
def sendData(data):
    requests.post('{0}/observations'.format(domain), data = json.dumps(data))
```

Де domain – задана константою адреса сервера, а data – вже підготовані до відправки дані. Для того, щоб надсилати дані неперервно через певні проміжки часу, визначимо функцію, яка робить запит до сенсорів та відправляє відформатовані дані з певним інтервалом:

```
def sendDataContinuously(sleepTime):
    while True:
        hum, temp = observer.readDHT22()
        s_temp = s_observer.read_temp()
        data = {
            'observation': {
                'humidity': hum,
                'temperature': temp,
```

```
            's_temperature': s_temp
        }
    }
    send_data(data)
    sleep(sleepTime)
```

де `observer` та `s_observer` – модулі, які виконують доступ до сенсорів та зчитування інформації.

3.2.2 Проектування серверної частини

Для реалізації серверної частини системи моніторингу та прогнозування стану середовища було обрано мову Ruby та Ruby on Rails середовище. Фреймворк реалізує патерн MVC [16]. Вибір на користь такої платформи мав декілька причин. Оскільки ми розробляємо лише макет системи, то важливість швидкодії обробки запитів не є першочерговою, якби було так, то слід було б звернути увагу на середовище Java або Elixir. Другим аспектом вибору є наявність бібліотек, здатних забезпечити вимоги до побудови прогнозуючої моделі. З такої точки зору ідеальним є мовне середовище R, оскільки воно містить велику кількість бібліотек для статистичного аналізу, побудови моделей, а також візуалізації даних. Але оскільки в разі використання цього середовища систему довелося б розбивати на окремі сервіси (окремо для аналізу та прийому-обробки-відображення даних), а також організовувати їх взаємодію, то таке рішення не можна вважати оптимальним. Натомість Ruby on Rails має середовище, що включає в себе велику кількість бібліотек, серед яких є й бібліотеки або ж версії бібліотек для аналізу даних, які задовольняють наші вимоги.

Для зберігання даних були обрані 2 типи сховищ: база даних PostgreSQL та зберігання даних від сторонніх API та даних від користувачів та тимчасове Redis сховище для зберігання результатів аналізу у вигляді пар ключ-значення, які не потрібно зберігати протягом великої кількості часу. Середовище Rails включає підтримку PostgreSQL баз даних на рівні основного сховища. Для інтеграції з Redis використаємо бібліотеку.

Основні процеси пов’язані з аналізом даних та зверненням до сторонніх API винесемо в окремі процеси, які будуть працювати в фоновому режимі, що сприятиме зменшенню загального навантаження. Для цього використовується технологія sidekiq, що дозволяє розгортати в фоні урізану версію додатку та виконувати обрахунки, використовуючи потоки, а не процеси (на відміну від rescue з його малою швидкодією).

Схематично структура додатку разом з усіма компонентами та взіємодіями між ними представлена на рис. 3.7.

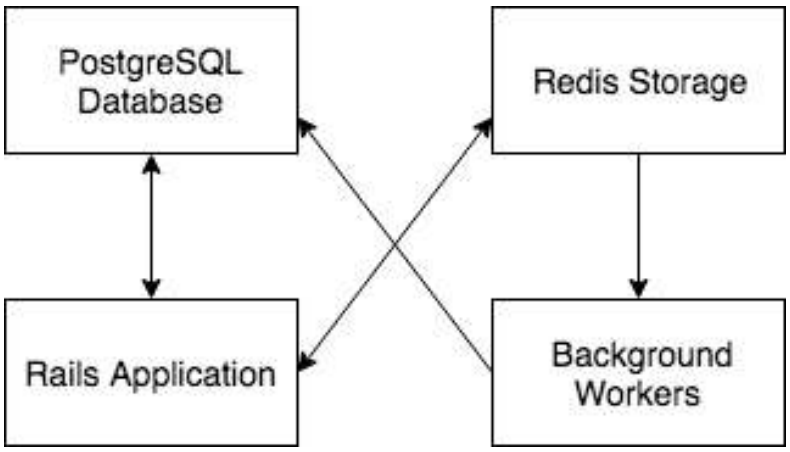

Рисунок 3.7 – Архітектура веб-додатку

Програмний код основних компонентів серверного застосунку наведено в додатку. При проектування були використані такі основні бібліотеки:

- Devise – Ruby-бібліотека, яка забезпечує автентифікацію користувачів, має модульну будову, тож дозволяє використовувати лише потрібні функції;
- Cancancam – Ruby-бібліотека для впровадження авторизації користувачів за правами доступу. Потрібна, щоб забезпечити можливість доступу користувача до даних, які призначені саме для нього;
- Active Model Serializers – бібліотека для формування даних для відповідей на запити у різних форматах, наприклад, JSON API;

- Geocoder – бібліотека, що дозволяє зручно працювати з різними форматами географічних даних, а також виконувати пошук та перетворення різних форматів геоданих;
- Statsample – бібліотека для роботи з статистичними даними та побудови моделей [17];
- Redis – адаптер для зв’язку зі сховищем даних Redis;
- Sidekiq – бібліотека для зручної організації та керування фоновими процесами в застосунку;
- Rspec – бібліотека для юніт-тестування.

3.2.2.1 Проектування сутностей та зв’язків між ними

Оскільки ми використовуємо об’єктно-орієнтований підхід до розробки і відповідну мову програмування, то потрібно виділити основні сутності а також зв’язки між ними. Найпростіше відобразити ці компоненти за допомогою діаграми зв’язків сутностей у базі даних (рис. 3.8). Фреймворк також дає можливість використовувати найпростіші методи доступу та запису полів, а також валідаційні методи для полів, тож пропустимо це.

На діаграмі найоб’ємнішими сутностями є сутність користувача (user), пристрою (device), вимірювання (observation) та історичних даних (historical_data). Це пояснюється тим, що кількість даних, які можуть виявитись потрібними при моделюванні (або перемоделюванні в майбутньому) моделі, невідома і будь-які дані можуть виявитись цінними. В базі даних ми зберігаємо майже всі параметри прогнозуючих моделей (для plan_forecast, harvest_forecast), не зберігаємо лише для прогнозування температурних показників.

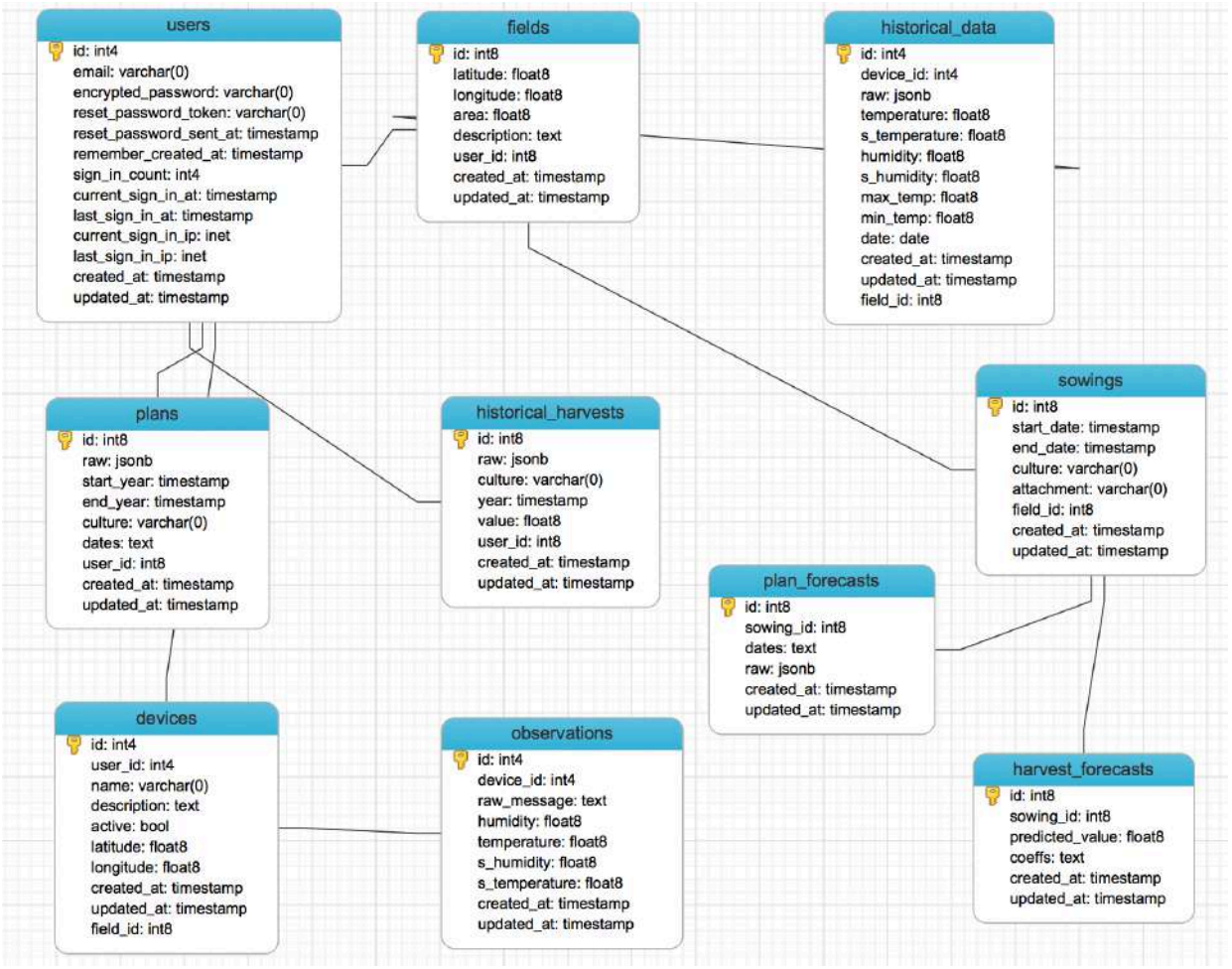

Рисунок 3.8 – Діаграма зв’язків сутностей

Збереження прогнозуючої моделі для температури та вологості будемо зберігати в тимчасовому сховищі (Redis), так як ця прогнозуюча модель потребуватиме перенавчання з доволі інтенсивною частотою (після кожного оновлення даних від зовнішніх ресурсів), інші ж моделі можуть вважатись досить сталими, так як показники, які впливають на їх навчання досить рідко змінюються.

Дані про історичні значення показників врожайності (historical_harvests) та плани внесення добрив (plans) будемо співвідносити безпосередньо з користувачем, щоб спростити доступ до цих даних.

Параметри прогнозуючих моделей та результатів прогнозування зв’язані безпосередньо з сутністю посіву(sowing), так як розраховуються саме для нього.

Сутність вимірювання (observation) очікувано пов’язана з сутністю пристрою (device). Вона зберігає значення температури та вологості, отримані від пристрою, а також все повідомлення у форматі json на випадок, якщо дані треба буде повторно використати, розділивши їх іншим чином та щоб зберегти так зване “single source of truth” – початкове повідомлення.

3.2.2.2 Генерація даних для моделі

Генерація даних є першим кроком моделювання прогнозуючої моделі. Перш за все потрібно обрати джерела даних для моделі.

Даних, отриманих від пристрою, може виявитись не достатньо, адже їх кількість варіюється в залежності від тривалості роботи пристрою. Тож треба залучати сторонні джерела. Для розробки макету системи було використано відкрите API ресурсу worldweatheronline.com. Воно здатне надавати повну інформацію про стан середовища в будь-який момент часу за наданими координатами. Єдиним обмеженням, яке потрібно враховувати при розробці, є обмеження кількості запитів до 1000 за добу. Формат даних, отриманих від API, має такий вигляд:


```
{
	"data": {
		"request": [
			{
				"type": "LatLon",
				"query": "Lat 46.14 and Lon 29.70"
			}
		],
		"weather": [
			{
				"date": "2017-05-02",
				"astronomy": [
					{
						"sunrise": "05:47 AM",
						"sunset": "08:11 PM",
						"moonrise": "11:21 AM",
						"moonset": "01:32 AM"
					}
				],
				"maxtempC": "23",
				"maxtempF": "74",
				"mintempC": "13",
				"mintempF": "56",
				"totalSnow_cm": "0.0",
				"sunHour": "0.0",
				"uvIndex": "0",
				"hourly": [
					{
						"time": "24",
						"tempC": "23",
```

```
                                        "tempF": "74",
                                        "windspeedMiles": "5",
                                        "windspeedKmph": "8",
                                        "winddirDegree": "61",
                                        "winddir16Point": "ENE",
                                        "weatherCode": "116",
                                        "weatherIconUrl": [
                                                {
                                                        "value":
"http://cdn.worldweatheronline.net/images/wsymbols01_png_64/wsymbol_0002_sunny_intervals.png"
                                                }
                                        ],
                                        "weatherDesc": [
                                                {
                                                        "value": "Partly cloudy"
                                                }
                                        ],
                                        "precipMM": "0.0",
                                        "humidity": "65",
                                        "visibility": "10",
                                        "pressure": "1018",
                                        "cloudcover": "15",
                                        "HeatIndexC": "18",
                                        "HeatIndexF": "65",
                                        "DewPointC": "10",
                                        "DewPointF": "50",
                                        "WindChillC": "18",
                                        "WindChillF": "64",
                                        "WindGustMiles": "8",
                                        "WindGustKmph": "13",
                                        "FeelsLikeC": "18",
                                        "FeelsLikeF": "64"
                                }
                        ]
                }
        ]
        }
}
```


API віддає велику кількість даних, серед яких є й потрібні температура та вологість, які ми й зберігаємо окремо в сутності, Інші дані, які на даний момент ніякої цінності не становлять, також зберігаємо в полі типу json, мак як є можливість, що в майбутньому вони можуть знадобитись.

Слід зазначити також важливість уніфікованості формату даних (градуси в шкалі за Цельсієм, вологість у відсотках і т.д.), дані мають постачатись в одному форматі, або ж зводитись до нього, щоб не виникало проблем із тренуванням моделі та представленням користувачу в системі.

Що стосується даних про календарні плани внесення добрив та показників врожайності, то можна виділити доволі невелику кількість джерел, які можуть надати таку інформацію. Враховуючи те, що ми намагатимемось побудувати залежності між цими показниками та температурними показниками, то дані мають якимось чином відноситись до тієї ж місцевості.

Серед можливих джерел інформації можна виділити самого користувача, який вірогідно вже має такі відомості та спеціалізовані дослідницькі установи. На етапі початкового запуску системи доцільно обрати перший варіант, так як користувач, зацікавлений у якості прогнозу, є одночасно зацікавленим у наданні правдивої інформації. Дані користувача можуть завантажуватись різними способами, включаючи завантаження вручну та за допомогою CSV-файлу. При розробці макету використаємо саме такий підхід, адже він дозволить максимально швидко та зручно завантажити великі об’єми даних. Дані про календарні плани внесення добрив мають такий формат:

```
culture,years,harvest,days
Corn,2010-2011,120.1,[29.10  30.10  31.10  1.11  15.03  16.03
17.03]
Corn,2012-2013,100.5,[21.10  22.10  23.10  25.10  23.02  24.02
25.02]
Wheat,2010-2011,"200,3",[29.10  30.10  31.10  1.11  15.03  16.03
17.03]
```

Таблиця 3.1 – Приклад файлу для внесення календарних планів у табличному форматі

culture	years	harvest	Days
Corn	2010-2011	120.1	[29.10 30.10 31.10 1.11 15.03 16.03 17.03]
Corn	2012-2013	100.5	[21.10 22.10 23.10 25.10 23.02 24.02 25.02]
Wheat	2010-2011	200,3	[29.10 30.10 31.10 1.11 15.03 16.03 17.03]

Код сервісів, які виконують діставання даних та їх форматування, наведений у додатку А.

3.2.2.3 Реалізація навчання моделі та генерації прогнозів

Для навчання моделі та виконання прогнозів використовуємо бібліотеку для роботи зі статистичним аналізом statsample. А саме такі її функції та модулі:

- `Statsample::Analysis.store` (назва моделі, блок для виконання) – для створення об’єкту моделі аналізу без запуску аналізу;

- `Statsample::Analysis.run` – для запуску аналізу на основі створеного об’єкту;
- `Daru::Vector` – модуль для генерації об’єктів типу вектор зі списків даних;
- Модуль `Statsample::Regression::Multiple` та функція `Statsample::Regression.multiple`(датасет із підписами, назва залежної змінної) – для тренування регресійної моделі багатьох змінних;
- `Statsample::Regression::Simple.new_from_dataset`(датасет з підписами, назва залежної змінної) – для тренування регресійної моделі;
- Модуль `Statsample::TimeSeries::ARIMA` та функція `Statsample::TimeSeries::ARIMA.ks` – для тренування ARMA моделі.

Функції, вказані вище, приймають у вигляді даних структуру типу `dataset`, унікальну для цієї бібліотеки. Для перетворення масиву даних у структуру такого типу потрібно створити новий вектор типу `Daru::Vector`, а потім за допомогою функцій `dataset` та `to_dataset` поєднати всі потрібні вектори у структуру типу dataset.

Підготовка даних до моделювання виконується в сервісних об’єктах, які є частиною Rails методології. Код сервісів, які виконують моделювання, наведений в додатку.

3.3 Огляд роботи макету системи

Після реєстрації (рис. 3.9) у сервісі користувач має багато можливостей: додавати нові пристрої для моніторингу (рис. 3.10), відстежувати дані, які присилаються пристроями, на зручних графіках (рис. 3.11), побачити прозноз температури та вологості для конкретного поля, виконано спеціально для нього за його координатами (рис 3.12).

Також реалізовано можливість завантажувати дані з календарними планами у форматі csv, так само можна виконувати завантаження даних про

показники врожайності за минулі роки. Отримати прогноз для посіву можна, відвідавши відповідні вкладки на його сторінці (рис 3.13, 3.14).

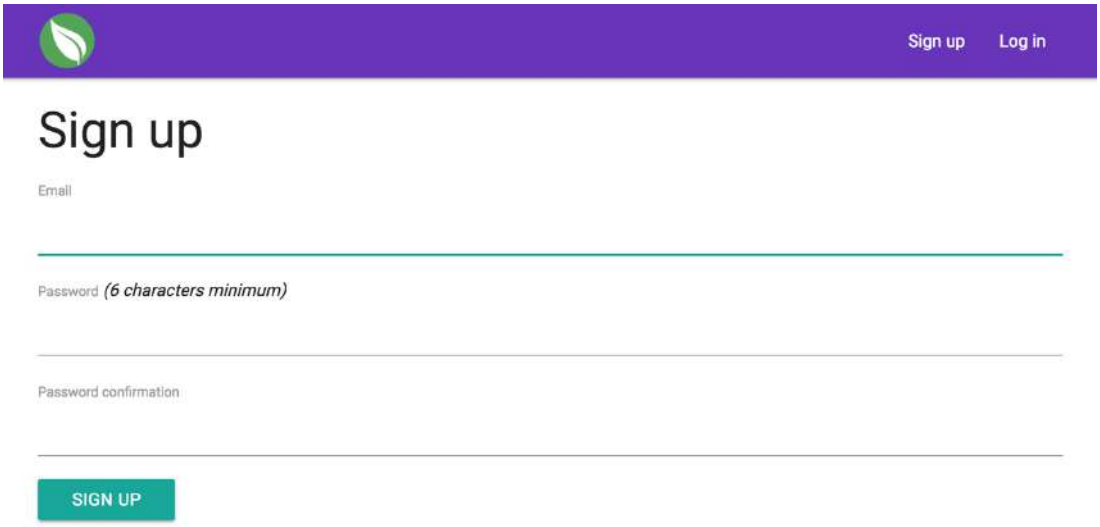

Рисунок 3.9 – Реєстрація на сервісі

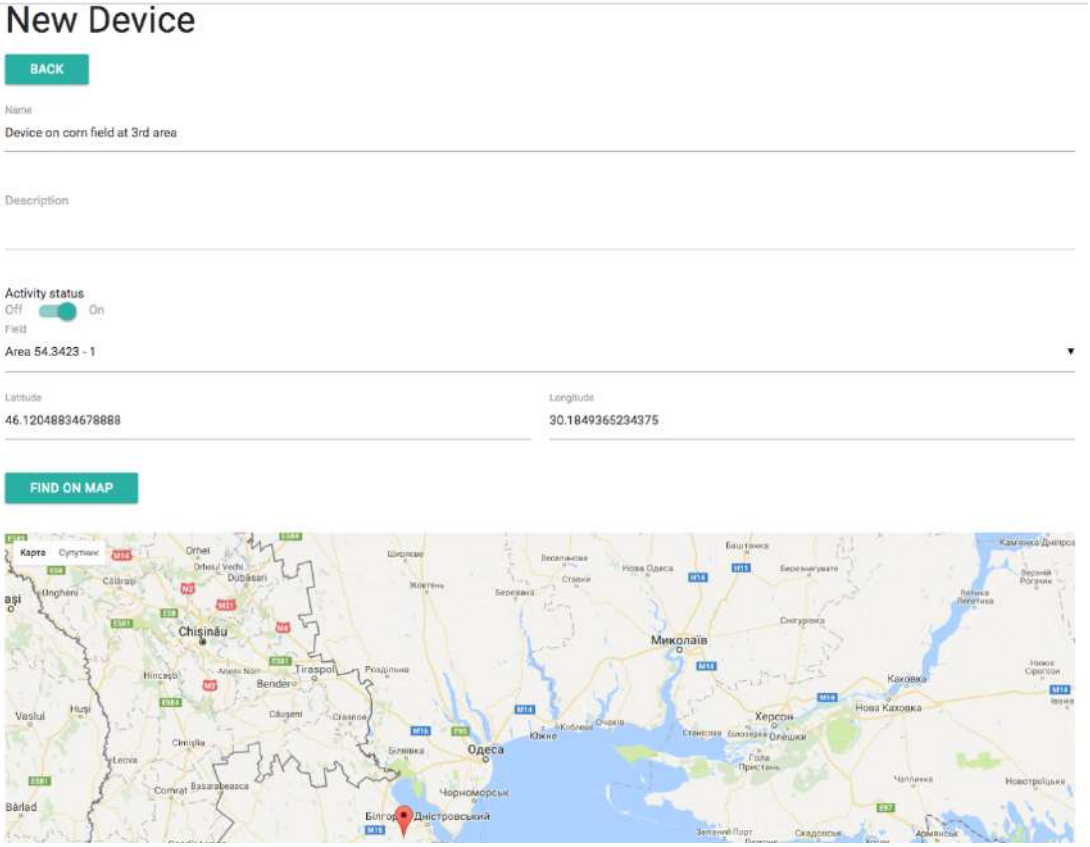

Рисунок 3.10 – Додавання нового моніторингового пристрою

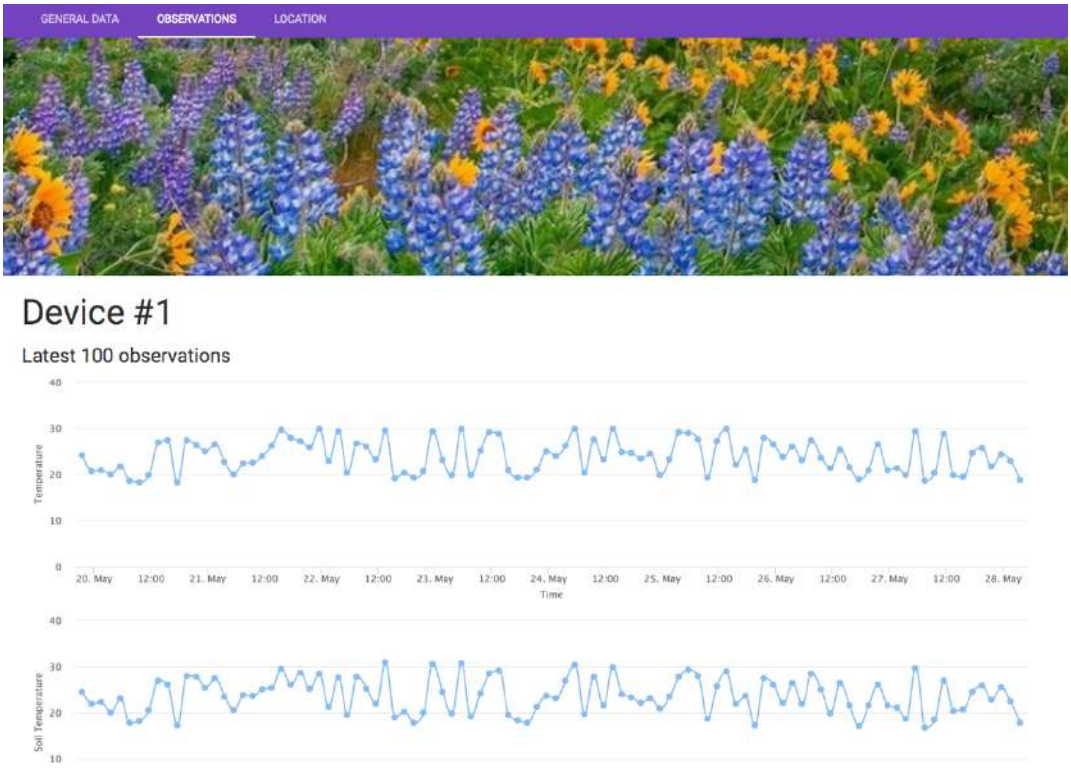

Рисунок 3.11 – Графіки з замірами, проведеними тестовим пристроєм

Рисунок 3.12 – Щоденний прогноз температури для вказаного поля

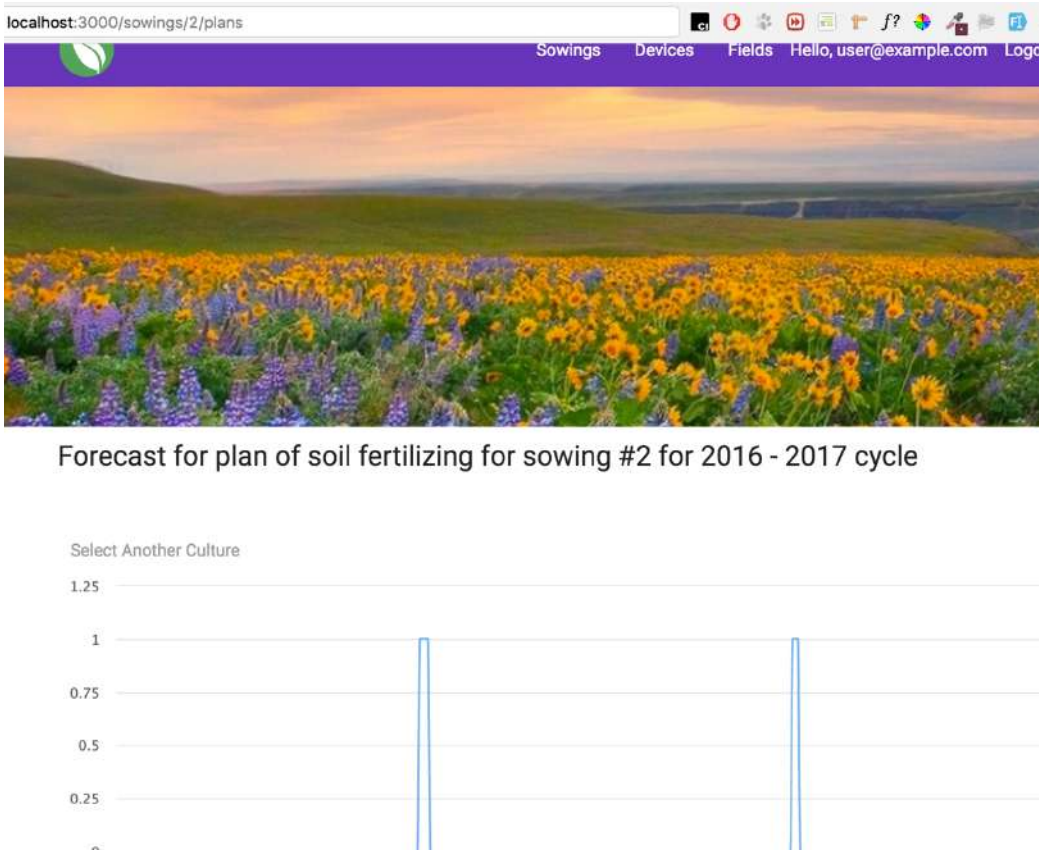

Рисунок 3.13 – Прогноз плану внесення добрив на 2017 рік для соняшнику

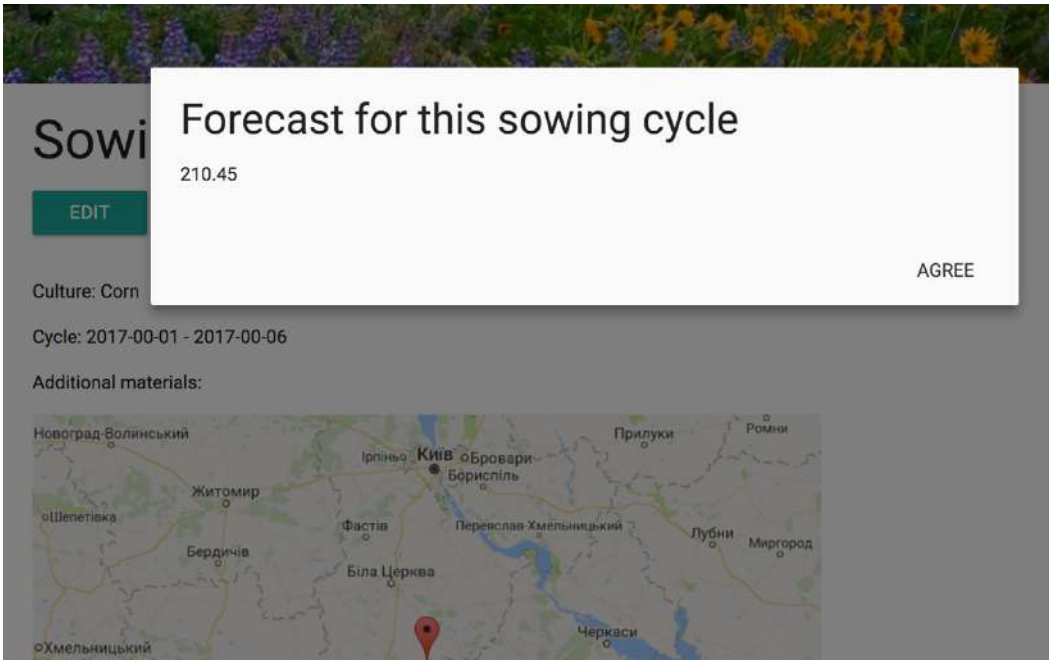

Рисунок 3.14 – Прогноз урожаю соняшнику на 2017 рік

3.4 Висновок

В ході роботи над 3 розділом було сформульовано основні вимоги до проектованої системи. На основі поставлених вимог було обрано архітектуру системи – клієнт-серверна архітектура з одним сервером та багатьма клієнтами, де в якості клієнта виступає пристрій для збору інформації. Зв'язок між сервером та клієнтами відбувається на основі протоколу HTTP/2.

Був розроблений пристрій на основі платформи Raspberry Pi 2 Model B із таким набором сенсорів: DHT22 для вимірювання температури та вологості повітря, DS18B20 для вимірювання температури ґрунту та Vodafone Connect 4100 для організації мобільного Інтернет-з'єднання.

Для реалізації серверу було обрано мову програмування Ruby та фреймворк Ruby on Rails основою якого є патерн MVC. Для проектування веб-застосунку в контексті стеку було обрано сервісний підхід, за якого основна логіка програми виконується в сервісних об'єктах. Тому основні процеси програми, такі як запити до Weather API та побудова моделей прогнозування, виконуються у сервісах.

4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі проводиться оцінка основних характеристик системи, що на базі програмного продукту виконує моніторинг та прогнозування стану середовища. Програмне забезпечення було розроблене за допомогою мов програмування Ruby і Python та JavaScript. Апаратний засіб був побудований на основі Raspberry Pi.

Програмний продукт призначено для використання на персональних комп'ютерах під управлінням операційної системи Linux або Mac OS X.

Нижче наведено аналіз різних варіантів реалізації модулю з метою вибору оптимальної, з огляду при цьому як на економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. Як прямі, так і побічні витрати розподіляються по продуктам та послугам у залежності від потрібних на кожному етапі виробництва обсягів ресурсів. Виконані на цих етапах дії у контексті метода ФВА називаються функціями.

Мета ФВА полягає у забезпеченні правильного розподілу ресурсів, виділених на виробництво продукції або надання послуг, на прямі та непрямі витрати. У даному випадку – аналізу функцій програмного продукту й виявлення усіх витрат на реалізацію цих функцій.

Фактично цей метод працює за таким алгоритмом:

– визначається послідовність функцій, необхідних для виробництва продукту. Спочатку – всі можливі, потім вони розподіляються по двом групам: ті, що впливають на вартість продукту і ті, що не впливають. На цьому ж етапі

оптимізується сама послідовність скороченням кроків, що не впливають на цінність і відповідно витрат.

– для кожної функції визначаються повні річні витрати й кількість робочих часів.

– для кожної функції наоснові оцінок попереднього пункту визначається кількісна характеристика джерел витрат.

– після того, як для кожної функції будуть визначені їх джерела витрат, проводиться кінцевий розрахунок витрат на виробництво продукту.

4.1 Постановка задачі техніко-економічного аналізу

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки системи аналізу місцезнаходження об’єктів та його прогнозування. Оскільки основні проектні рішення стосуються всієї системи, кожна окрема підсистема має їм задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для вибору методу прогнозування місцезнаходження об’єктів у контекстно-залежних системах.

Відповідно цьому варто обирати і систему показників якості програмного продукту.

Технічні вимоги до продукту наступні:

– програмний продукт повинен функціонувати на Raspberry PI з визначеним набором компонентів;

– інтерфейс користувача повинен нормально функціонувати на будь-якій операційній системі у нових версіях веб браузерів;

– забезпечувати високу швидкість обробки великих об’ємів даних у реальному часі;

– забезпечувати зручність і простоту взаємодії з користувачем або з розробником програмного забезпечення у випадку використовування його як модуля;

– передбачати мінімальні витрати на впровадження програмного продукту.

4.1.1 Обґрунтування функцій програмного продукту

Головна функція F_0– розробка системи моніторингу та прогнозування стану середовища на основі технології Інтернет речей. Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

- F_1 – вибір мови програмування, що використовується для реалізації системи;
- F_2 – вибір апаратних компонентів;
- F_3 – використання готових бібліотек для реалізації прогнозуючої моделі.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F_1:

а) мова програмування C++;

б) мова програмування Python;

Функція F_2:

а) мікрокомп’ютер Raspberry Pi;

б) електрична платформа Arduino;

Функція F_3:

а) написання алгоритмів вручну;

б) використання готових бібліотек;

4.1.2 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (таблиця 4.1).

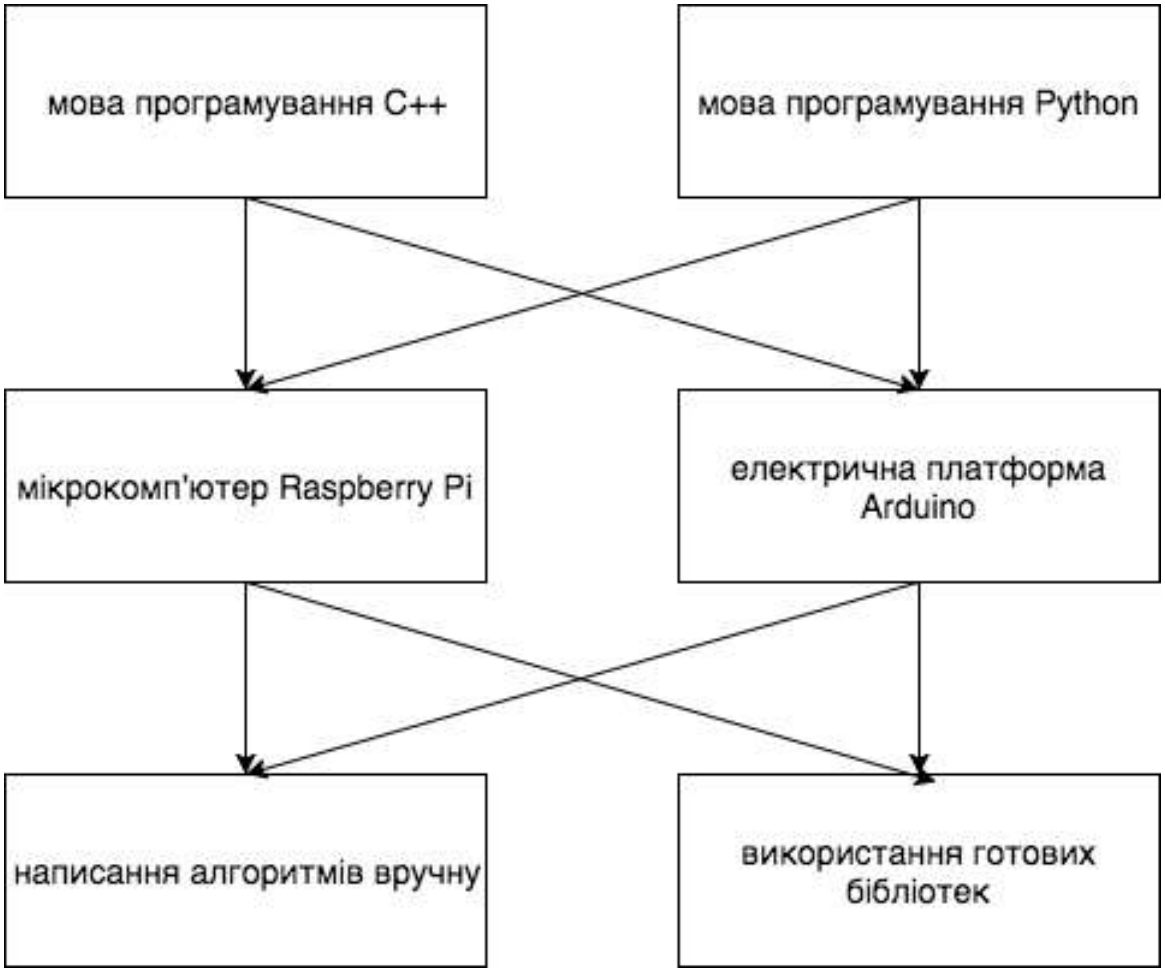

Рисунок 4.1 – Морфологічна карта

Морфологічна карта відображає всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

Таблиця 4.1 – Позитивно-негативна матриця

Основні функції	**Варіанти реалізації**	**Переваги**	**Недоліки**
F1	*А*	Висока швидкість виконання коду, менше помилок завдяки статичній типізації	Вищі витрати часу на написання коду
	Б	Простіший код, який легше писати та розуміти, велика кількість бібліотек	Виконання коду займає більший час

Таблиця 4.1 – Позитивно-негативна матриця (закінчення)

Основні функції	**Варіанти реалізації**	**Переваги**	**Недоліки**
F2	А	Простота у використанні	Відсутність АЦП
	Б	Наявність АЦП, найбільше підходить при розробці мікроконтролерів	Більші витрати часу на проектування
F3	*А*	Більша гнучкість у використанні	Більше часу на написання коду
	Б	Менше часу на написання коду	Менша гнучкість у використанні

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам.

Функція *F1:*

Оскільки проводиться підключення великої кількості датчиків, потрібно швидко реалізовувати обробку даних, тому варіант а) має бути відкинутий.

Функція *F2:*

Оскільки для прогнозуючої моделі достатньо даних, яку можна отримати через цифрові датчики, то варіант б) можна відкинути.

Функція *F3:*

Оскільки методи написані вручну та за допомогою готових бібліотек будуть давати однакові результати, вважаємо варіанти а) та б) гідними розгляду.

Таким чином, будемо розглядати такі варіанти реалізації ПП:

1. *F1б – F2а – F3а*
2. *F1ю – F2а – F3б*

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.2 Обґрунтування системи параметрів ПП

4.2.1 Опис параметрів

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

– *X1* – швидкодія мови програмування;

– *X2* – простота використання програмного забезпечення;

– *X3* – потенційний об’єм програмного коду;

– *X4* – простота реалізації прогнозуючої моделі.

X1: Відображає швидкодію операцій залежно від обраної мови програмування.

X2: Відображає рівень складності використання та налаштування програмного забезпечення користувачем.

X3: Показує розмір програмного коду який необхідно створити безпосередньо розробнику.

X4: Відображає складність написання моделі, виражену у затраченому часі на написання коду.

4.2.2 Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл. 4.2.

За даними таблиці 4.2 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.5.

Таблиця 4.2 – Основні параметри ПП

Назва Параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія мови програмування	X1	Оп/мс	2000	11000	19000
Простота використання програмного забезпечення	X2	Хвилини на підключення та налаштування модулю	600	300	60
Потенційний об’єм програмного коду	X3	Кількість строк коду	2000	1500	1000
Простота реалізації прогнозуючої моделі	X4	Строк коду / година	10	35	70

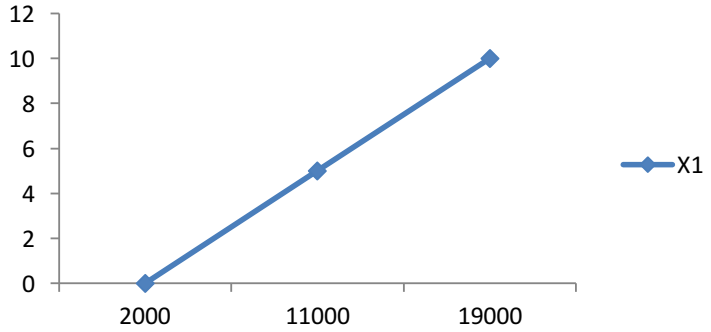

Рисунок 4.2 – X1, швидкодія мови програмування

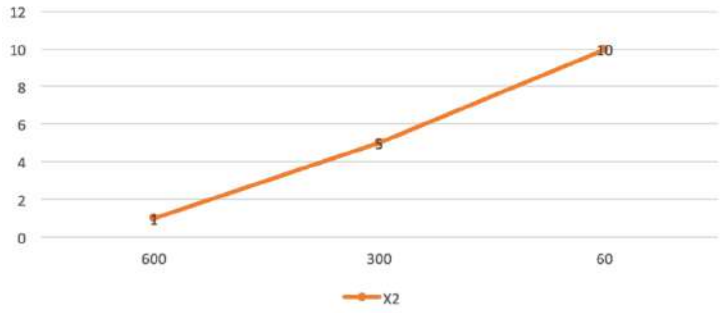

Рисунок 4.3 – X2, простота використання програмного забезпечення

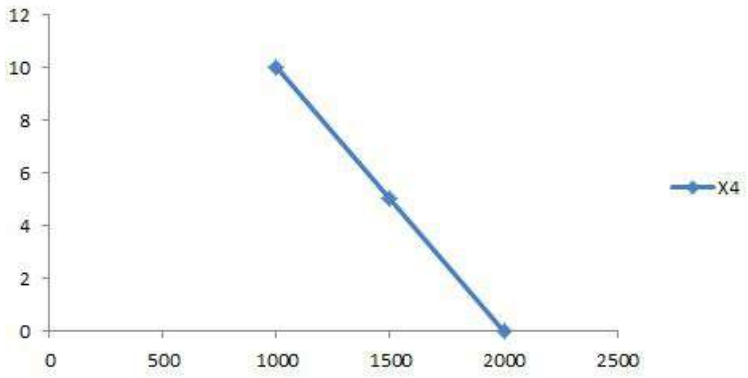

Рисунок 4.4 – X3, потенційний об’єм програмного коду

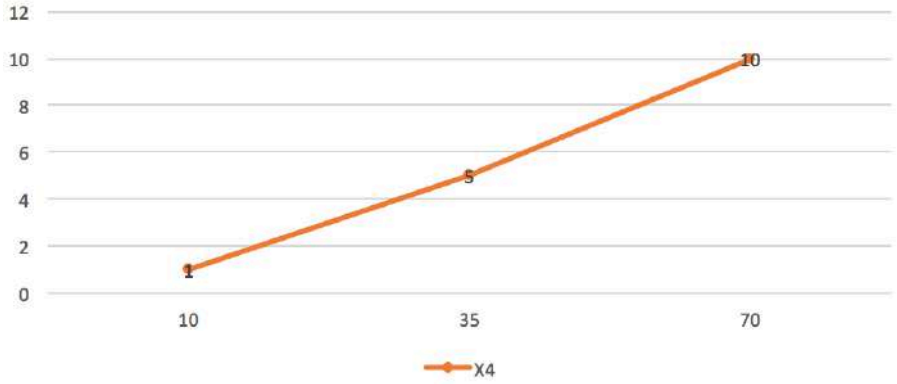

Рисунок 4.5 – X4, простота реалізації прогнозуючої моделі

4.2.3 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

– визначення рівня значимості параметра шляхом присвоєння різних рангів;

– перевірку придатності експертних оцінок для подальшого використання;

– визначення оцінки попарного пріоритету параметрів;

– обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 – Результати ранжування параметрів

Позначення параметра	**Назва параметра**	**Одиниці виміру**	**Ранг параметра за оцінкою експерта**							**Сума рангів R_i**	**Відхилення Δ_i**	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	Оп/мс	4	3	4	4	4	4	4	27	0.75	0.56
X2	Простота використання програмного забезпечення	хвилина	4	4	4	3	4	3	3	25	-1.25	1.56
X3	Потенційний об’єм програмного коду	кількість строк коду	2	2	1	2	1	2	2	12	-14.25	203.06
X4	Простота реалізації прогнозуючої моделі	строк коду / година	5	6	6	6	6	6	6	41	14.75	217.56
	Разом		15	15	15	15	15	15	15	105	0	420.75

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 105 \qquad (4.1)$$

де N – число експертів, n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 26.25 \tag{4.2}$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T \tag{4.3}$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 420.75 \tag{4.4}$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 420.75}{7^2(5^3 - 5)} = 1{,}03 > W_k = 0.67 \tag{4.5}$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0.67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4.

Таблиця4.4 – Попарне порівняння параметрів

Параметри	**Експерти**							**Кінцева оцінка**	**Числове значення**
	1	2	3	4	5	6	7		
X1 і X2	=	>	=	<	=	<	<	<	0.5
X1 і X3	<	<	<	<	<	<	<	<	0.5
X1 і X4	>	>	>	>	>	>	>	>	1.5
X2 і X3	<	<	<	<	<	<	<	<	0.5
X2 і X4	>	>	>	>	>	>	>	>	1.5
X3 і X4	>	>	>	>	>	>	>	>	1.5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij}визначається за формулою:

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases} \quad (4.6)$$

З отриманих числових оцінок переваги складемо матрицю A=$\| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{вi}$ за наступними формулами:

$$K_{\text{вi}} = \frac{b_i}{\sum_{i=1}^{n} b_i}, \quad (4.7)$$

де bi розраховується за наступною формулою:

$$b_i = \sum_{i=1}^{N} a_{ij}. \quad (4.8)$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%).На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{вi}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \quad (4.9)$$

де b`$_i$ розраховується за наступною формулою

$$b_i' = \sum_{i=1}^{N} a_{ij} b_j. \quad (4.10)$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 – Розрахунок вагомості параметрів

Параметри x_i	**Параметри x_j**				**Перша ітер.**		**Друга ітер.**		**Третя ітер**	
	X1	X2	X3	X4	b_i	$K_{\text{вi}}$	b_i^1	$K_{\text{вi}}^1$	b_i^2	$K_{\text{вi}}^2$
X1	1.0	0.5	0.5	1.5	3.5	0.219	22.25	0.216	100	0.215
X2	1.5	1.0	0.5	1.5	4.5	0.281	27.25	0.282	124.25	0.283
X3	1.5	1.5	1.0	1.5	5.5	0.344	34.25	0.347	156	0.348
X4	0.5	0.5	0.5	1.0	2.5	0.156	14.25	0.155	64.75	0.154
Всього:					16	1	98	1	445	1

4.3 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X1* (швидкодія мови програмування) та *X2* (простота використання програмного забезпечення) відповідають технічним вимогам умов функціонування даного ПП.1

Варіант б) більш простий для реалізації: при його використанні потрібно 1150 рядків коду (X3), у той час як варіант а) вимагає 1700 рядків. За умови вибору варіанту а) реалізація прогнозуючої моделі займе більший час: швидкість написання коду (X4) становитиме 30 строк/година, тоді як варіант б) забезпечує швидкість на рівні 60 строк/година.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6):

$$K_K(j) = \sum_{i=1}^{n} K_{\text{в}i,j} B_{i,j}, \tag{4.11}$$

де n – кількість параметрів;$K_{вi}$– коефіцієнт вагомості i–го параметра;B_i – оцінка i–го параметра в балах.

За даними з таблиці 4.6 за формулою

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}], \tag{4.12}$$

визначаємо рівень якості кожного з варіантів:

K_{K1} = 1,075 + 1,392 + 0,462 + 2,2357 = 5,1647

K_{K2} = 1,075 + 2,784 + 1,309 + 2,2357 = 7,4037

Як видно з розрахунків, кращим є другий варіант, для якого коефіцієнт технічного рівня має найбільше значення.

Таблиця 4.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1(X1)	Б	11000	5	0,215	1,075
F3(X4)	А	30	4	0,348	1,392
	Б	60	8		2,784
F3(X3)	Б	1150	8,5	0,154	1,309
	А	1700	3		0,462
F2(X2)	А	160	7,9	0,283	2,2357

4.4 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;
2. Розробка програмної реалізації системи;

При реалізації варіанту а) з’являється додаткова задача до завдання 2:

3. Реалізація алгоритмів прогнозуючих моделей.

Варіант б) вимагає виконання такої підзадачі у контексті виконання задачі 2:

4. Дослідження сторонніх бібліотек.

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 2.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М}, \quad (4.13)$$

де T_P – трудомісткість розробки ПП;$K_П$ – поправочний коефіцієнт;$K_{СК}$ – коефіцієнт на складність вхідної інформації; K_M – коефіцієнт рівня мови програмування;$K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;$K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: T_P =90 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $K_П$ = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх чотирьох завдань рівний 1: $K_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $K_{СТ}$ = 0.8. Тоді, за формулою 5.1, загальна трудомісткість програмування першого завдання дорівнює:

T_1 = 90*1.7*0.8 = 122.4 людино-днів.

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм 2-ї групи складності, степінь новизни Б), тобто T_P = 27 людино-днів, $K_П$ = 1.08, $K_{СК}$ = 1,$K_{СТ}$ = 0.8:

T_2 = 27 * 1.08 * 0.8 = 23.328 людино-днів.

Для третього завдання (використовується алгоритм 1-ї групи складності, степінь новизни В), тобто T_P = 43 людино-дні, $K_П$ = 0,81, $K_{СК}$ = 1,$K_{СТ}$ = 0.8:

$Т_2$ = 43 * 0.81 * 0.8 = 27.864 людино-днів.

Для четвертого завдання (використовується алгоритм 3-ї групи складності, степінь новизни В), тобто $Т_Р$ = 12 людино-днів, $К_П$ = 0,60, $К_{СК}$ = 1,$К_{СТ}$ = 0.8:

$Т_2$ = 12 * 0.60 * 0.8 = 5.76 людино-днів.

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$Т_I$ = (122.4 + 23.328 + 27.864) · 8 = 1388.74 людино-годин;

$Т_{II}$ = (122.4 + 23.328 + 5.76) · 8 = 1211.904 людино-годин;

Найвищу трудомісткість має варіант І.

В розробці беруть участь два програмісти з окладом 8000 грн., один розробник апаратного забезпечення з окладом 10000грн. Визначимо зарплату за годину за формулою:

$$СЧ = \frac{М}{T_m \cdot t} грн., \quad (4.14)$$

де М – місячний оклад працівників;T_m – кількість робочих днів у місяць; t – кількість робочих годин в день.

$$С_ч = \frac{8000 + 8000 + 10000}{3 \cdot 21 \cdot 8} = 51{,}59 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою

$$СЗП = С_ч \cdot Т_i \cdot КД, \quad (4.15)$$

де $С_ч$– величина погодинної оплати праці програміста;$Т_i$ – трудомісткість відповідного завдання; $К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП}$ = 51,59 · 1388.74 · 1.2 = 85974,12 грн.

II. $С_{ЗП}$ = 51,59 · 1211.904 · 1.2 = 75026,55 грн.

Відрахування на єдиний соціальний внесок в залежності від групи професійного ризику (II клас) становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 85974.12 \cdot 0.22 = 18914.31$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 75026,55 \cdot 0.22 = 16505.84$ грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного програміста з окладом 8000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 8000 \cdot 0,2 = 19200$ грн.

З урахуванням додаткової заробітної плати:

$С_{ЗП} = С_Г \cdot (1 + К_З) = 19200 \cdot (1 + 0.2) = 23040$ грн.

Відрахування на єдиний соціальний внесок:

$С_{ВІД} = С_{ЗП} \cdot 0.22 = 23040 \cdot 0.22 = 5068.8$ грн.

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 8000 грн.

$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 8000 = 2300$ грн.,

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; $К_А$– річна норма амортизації; $Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 8000 \cdot 0.05 = 460$ грн.,

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 8 – 16) \cdot 8 \cdot 0.9 = 1706.4$ годин,

де $Д_К$ – календарна кількість днів у році; $Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів; $Д_Р$ – кількість днів планових ремонтів устаткування; t –кількість робочих годин в день; $К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1706,4 \cdot 0,156 \cdot 0,2 \cdot 1,94177 = 103.379$грн.,

де N_C – середньо-споживча потужність приладу; $K_З$– коефіцієнтом зайнятості приладу; $Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$С_Н = Ц_{ПР} \cdot 0.67 = 8000 \cdot 0,67 = 5360$ грн.

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н$$

$С_{ЕКС} = 23040 + 5068.8 + 2300 + 460 + 103.379 + 5360 = 36332.179$ грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 36332.179 / 1706,4 = 21.29$ грн/час.

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$С_М = С_{М\text{-}Г} \cdot Т$$

I. $С_М = 21.29 \cdot 1388.74 = 29566.28$ грн.;

II. $С_М = 21.29 \cdot 1211.904 = 25801.44$ грн.;

Накладні витрати складають 67% від заробітної плати:

$$С_Н = С_{ЗП} \cdot 0,67$$

I. $С_Н = 85974,12 \cdot 0,67 = 57602.66$ грн.;

II. $С_Н = 75026,55 \cdot 0,67 = 50267.79$ грн.;

Отже, вартість розробки ПП за варіантами становить:

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_М + С_Н$$

I. $С_{ПП} = 85974.12 + 18914.31 + 29566.28 + 57602.66 = 192057.37$грн.;

II. $С_{ПП} = 75026.55 + 16505.84 + 25801.44 + 50267.79 = 167601.62$грн.;

4.5 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕРj} = К_{Кj} / С_{Фj}, \qquad (4.16)$$

$К_{ТЕР1} = 5.1647 / 192057.37 = 2.689 \cdot 10^{-5}$;

$К_{ТЕР2} = 7.4037 / 167601.62 = 4.417 \cdot 10^{-5}$;

Як бачимо, найбільш ефективним є другий варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 4.417 \cdot 10^{-5}$.

4.6 Висновок

В даному розділі проведено повний функціонально-вартісний аналіз ПП, який було розроблено в рамках дипломного проекту. Процес аналізу можна умовно розділити на дві частини.

В першій з них проведено дослідження ПП з технічної точки зору: було визначено основні функції ПП та сформовано множину варіантів їх реалізації; на основі обчислених значень параметрів, а також експертних оцінок їх важливості було обчислено коефіцієнт технічного рівня, який і дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій ПП.

Другу частину ФВА присвячено вибору із альтернативних варіантів реалізації найбільш економічно обґрунтованого. Порівняння запропонованих варіантів реалізації в рамках даної частини виконувалось за коефіцієнтом ефективності, для обчислення якого були обчислені такі допоміжні параметри, як трудомісткість, витрати на заробітну плату, накладні витрати.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є другий варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР} = 4.417 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

– мова програмування – Python;

– мікрокомп’ютер Raspberry Pi;

– написання алгоритмів вручну.

ВИСНОВКИ

У ході виконання дипломного проекту було побудовано систему для моніторингу та прогнозування стану середовища на основі технології Інтернет речей.

Було досліджено технологію Інтернет речей, на основі якої виконувалось проектування системи, зроблено огляд типів систем такого плану, проаналізовано основні недоліки технології. Інша суттєва частина роботи полягала у побудові моделі, здатної прогнозувати врожайність поточного посівного періоду для визначених культур. В ході розв’язання цієї задачі було чітко виділено предмет прогнозування та зроблено огляд частини агрономічної галузі, яка стосується методів прогнозування та поліпшення якості врожаю, для виявлення можливих стійких залежностей між параметрами, які можуть впливати на прогнозований показник. На основі проведеного дослідження було виділено параметри, які потрібні для реалізації прогнозуючої моделі та зв’язки між ними. Також було досліджено методи та моделі прогнозування, досліджено основні сфери їх застосування та переваги із недоліками.

На основі проведеного дослідження було обрано комбіновану модель для прогнозування, яка складається з трьох ступенів обробки та передбачення, причому результат роботи кожного з етапів обробки може бути використаний окремо. Перевага такої моделі в тому, що вона дозволяє не прив’язуватись лише до одного методу прогнозування, а обирати найбільш зручний для прогнозування кожної складової частини системи.

Така модель має недолік, пов’язаний із втратою точності на кожному кроці, але так як функція прогнозування розроблюваної системи носить рекомендаційний характер, підтримка точності не є основною задачею.

Поряд із побудовою прогнозуючої моделі було спроектовано загальну архітектуру веб-додатку, який призначений для організації зручної роботи з даними від пристрою, а також різними типами даних про пристрої, поля та

культури, які обрані користувачем. Інформація виводиться у зручному графічному вигляді, що полегшує її сприйняття користувачем.

Така система вигідно відрізняється від існуючих рішень у сфері забезпечення точного землеробства. Використання датчиків дає можливість отримувати найактуальнішу інформацію про стан середовища. Кількість та призначення датчиків є розширюваним, адже вони не прив’язуються до конкретної платформи, що, за необхідності дозволить виконати ширший спектр вимірювань. Прогнози, які надає система, допоможуть вчасно виявити недоліки планування та змінити їх в залежності від ситуації. Після зміни таких показників система автоматично оновлює прогнози. Це дозволяє також оцінити якість обраної стратегії.

ПЕРЕЛІК ПОСИЛАНЬ

1. H. Geng. Internet of Things and Data Analytics Handbook / H. Geng. –USA: John Wiley & Sons, 2017. — С. 44.
2. Forecasting fundamentals. – Режим доступа: http://mech.at.ua/Forecasting.pdf. – Дата доступа: 22.05.2017.
3. Engineering statistics handbook. – Режим доступа: http://www.itl.nist.gov/div898/handbook/. – Дата доступа: 23.05.2017.
4. John D. Kelleher. Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies 1st Edition / John D. Kelleher, Brian Mac Namee, Aoife D'Arcy. – USA: MIT Press, 2015. – С. 145.
5. George E.P. Box. Time Series Analysis: Forecasting and Control 5th Edition / George E.P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, Greta M. Ljung. – USA: John Wiley & Sons P, 2016. – С. 250-252.
6. Rob J. Hyndman. Forecasting: principles and practice / Rob J. Hyndman, George Athanasopoulos. – USA: OTexts, 2013. – С. 34 – 56.
7. Robert H. Shumway. Time Series Analysis and Its Applications: With R Examples 3rd edition / Robert H. Shumway, David S. Stoffer. USA: Springer Texts in Statistics, 2011. – С. 67.
8. Iain Pardoe. Applied Regression Modeling 2nd Edition / Iain Pardoe. – USA: John Wiley & Sons, 2012. – С. 78-79.
9. John C. Chambers. How to choose the right forecasting technique / John C. Chambers, Satinder K. Mullick, Donald D. Smith. – USA: Harvard Business Review, 1971 – С. 23.
10. Ушкаренко В.О. Зрошуване землеробство / Ушкаренко В.О. – Херсон, Україна: Урожай, 1994. – С. 120.
11. Фактори врожаю та стабілізації землеробства. – Режим доступа: https://tehngaluzy.wordpress.com/2011/10/25/1-2-фактори-врожаю-та-стабілізації-земле/. – Дата доступа: 25.05.2017.

12. Офіційна документація Raspberry Pi. – Режим доступа: https://www.raspberrypi.org/resources/. – Дата доступа: 25.05.2017.
13. Офіційна документація Python 2.7. – Режим доступа: https://docs.python.org/2/. – Дата доступа: 25.05.2017.
14. Офіційна документація та джерело Pigpio бібліотеки. – Режим доступа: http://abyz.co.uk/rpi/pigpio/. – Дата доступа: 25.05.2017.
15. Adafruit's Raspberry Pi Lesson 11. DS18B20 Temperature Sensing. – Режим доступа: https://cdn-learn.adafruit.com/downloads/pdf/adafruits-raspberry-pi-lesson-11-ds18b20-temperature-sensing.pdf – Дата доступа: 28.05.2017.
16. Exploring Model View Controller. – Режим доступа: https://www.packtpub.com/books/content/exploring-model-view-controller. – Дата доступа: 25.05.2017.
17. Офіційна документація та джерело Statsample бібліотеки для Ruby. – Режим доступа: https://github.com/clbustos/statsample. – Дата доступа: 25.05.2017.

ДОДАТОК А

Фрагменти лістингу програми

s_observer.py

```
import os
import glob
import time

os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')

base_dir = '/sys/bus/w1/devices/'
device_folder = glob.glob(base_dir + '28*')[0]
device_file = device_folder + '/w1_slave'

def read_temp_raw():
    f = open(device_file, 'r')
    lines = f.readlines()
    f.close()
    return lines

def read_temp():
    lines = read_temp_raw()
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = read_temp_raw()
    equals_pos = lines[1].find('t=')
    if equals_pos != -1:
        temp_string = lines[1][equals_pos+2:]
        temp_c = float(temp_string) / 1000.0
        temp_f = temp_c * 9.0 / 5.0 + 32.0
        return '%.2f' % temp_c
```

observer.py

```
import pigpio
from time import sleep
# Pigpio DHT22 module should be in same folder as your
program
import DHT22
pi = pigpio.pi()
s = DHT22.sensor(pi, 17)
s.trigger()
```

```
sleep(.01) # Necessary on faster Raspberry Pi's

sleepTime = 3

def readDHT22():
        s.trigger()
        hum = '%.2f' % (s.humidity())
        temp = '%.2f' % (s.temperature())
        return (hum, temp)
```

main.py

```
import observer
import s_observer
from time import sleep
import requests

sleepTime = 5

def sendData(data):
    requests.post('{0}/api/data'.format(domain),
data=json.dumps(data))

def sendDataContinuously(sleepTime):
    while True:
        hum, temp = observer.readDHT22()
        s_temp = s_observer.read_temp()
        data = {
            'observation': {
                'humidity': hum,
                'temperature': temp,
                's_temperature': s_temp
            }
        }
        send_data(data)
        sleep(sleepTime)


while True:
        hum, temp = observer.readDHT22()
        s_temp = s_observer.read_temp()
        print("Humidity: " + hum + "%")
        print("Temperature: " + temp + "C")
        print("Soil Temperature: " + s_temp + "C")
        data = {
            'observation': {
```

```
                'humidity': hum,
                'temperature': temp,
                's_temperature': s_temp
            }
        }
        send_data(data)
        sleep(sleepTime)
```

initial_historic_data_creator.rb

```
class InitialHistoricDataCreator
  attr_reader :errors

  MONTH_DAYS_COUNT = [31, 28, 31, 30, 31, 30, 31, 31, 30,
31, 30, 31]

  # coords - [46.139,29.698]
  # date - 2009-04-21
  # max interval - 1 month
  def initialize(device_id: nil)
    @device_id = device_id
    device = Device.find_by_id(device_id)
    @errors = []

    if device == nil
      errors    <<    "Couldn't    find    Device    with
'id'=#{device_id}"
      return false
    end

    @coords = [device.latitude, device.longitude]
  end

  def perform
    get_past_years
    get_current_year
    get_29_of_february
    get_februaries
  end

private

  def start_year
    (Time.now - 5.years).strftime("%Y")
  end
```

```
  def end_year
    Time.now.strftime("%Y")
  end

  def start_month
    '01'
  end

  def end_month
    Time.now.strftime("%m")
  end

  def end_day
    Time.now.strftime("%d")
  end

  def get_past_years
    (start_year...end_year).each do |year|
      return if year == end_year

      MONTH_DAYS_COUNT.each_with_index do |month, index|
        start_date = "#{year}-#{index+1}-1"
        end_date = "#{year}-#{index+1}-#{month}"

        extractor  =  HistoricalDataExtractor.new(coords:
@coords, start_date: start_date, end_date: end_date)
        extractor.perform
        month_data = extractor.data

parse_month_data_to_db(month_data['data']['weather'])
      end
    end
  end

  def get_current_year
    (start_month..end_month).each_with_index  do  |month,
index|
      if month != end_month
        start_date = "#{end_year}-#{month}-1"
        end_date         =         "#{end_year}-#{month}-
#{MONTH_DAYS_COUNT[index]}"
      else
        start_date = "#{end_year}-#{end_month}-1"
        end_date = "#{end_year}-#{end_month}-#{end_day}"
      end
```

```
      extractor = HistoricalDataExtractor.new(coords:
@coords, start_date: start_date, end_date: end_date)
      extractor.perform
      month_data = extractor.data

parse_month_data_to_db(month_data['data']['weather'])
    end
  end

  def get_29_of_february
    even_years = (start_year..end_year).select { |x|
x.to_i/4 == 0 }

    even_years.each do |year|
      start_date = "#{year}-02-1"
      end_date = "#{year}-02-29"

      extractor = HistoricalDataExtractor.new(coords:
@coords, start_date: start_date, end_date: end_date)
      extractor.perform
      month_data = extractor.data

parse_month_data_to_db(month_data['data']['weather'])
    end
  end

  def get_februaries
    even_years = (start_year..end_year).select { |x|
x.to_i/4 }

    even_years.each do |year|
      start_date = "#{year}-02-1"
      end_date = "#{year}-02-28"

      extractor = HistoricalDataExtractor.new(coords:
@coords, start_date: start_date, end_date: end_date)
      extractor.perform
      month_data = extractor.data

parse_month_data_to_db(month_data['data']['weather'])
    end
  end

  def parse_month_data_to_db(month_data)
```

```
    month_data.each do |current_day|
      h_data_params = {}
      h_data_params['device_id'] = @device_id
      h_data_params['date'] = current_day['date']
      h_data_params['max_temp'] = current_day['maxtempC']
      h_data_params['min_temp'] = current_day['mintempC']
      h_data_params['temperature']                          =
current_day['hourly'][0]['tempC']
      h_data_params['s_temperature']                        =
current_day['hourly'][0]['tempC']
      h_data_params['humidity']                             =
current_day['hourly'][0]['humidity']
      h_data_params['s_humidity']                           =
current_day['hourly'][0]['humidity']
      h_data_params['raw'] = current_day

      HistoricalData.create!(h_data_params)
    end
  end
end
```

historical_data_extractor.rb

```
class HistoricalDataExtractor
  attr_reader :data

  # coords - [46.139,29.698]
  # date - 2009-04-21
  # max interval - 1 month
  def initialize(coords: nil, start_date: nil, end_date:
nil)
    @coords = coords.join(',')
    @start_date = start_date
    #.strftime("%Y-%m-%d")
    @end_date = end_date
    #.strftime("%Y-%m-%d")
    @data = nil
  end

  def perform
    url         =         "#{ENV['WEATHER_ARI_URI']}past-
weather.ashx?key=#{ENV['WEATHER_API_KEY']}&q=#{@coords}&f
ormat=json&date=#{@start_date}&enddate=#{@end_date}&tp=24
"
    @data = ActiveSupport::JSON.decode(open(url).read)
```

```
  end
end
```

user_historical_info_creator.rb

```
Class UserHistoricalInfoCreator
  attr_reader :errors

  def initialize(csv_file:, user_id:)
    @csv = csv_file
    @user = User.find(user_id)
  end

  def perform
    validate_csv && store_data
  end

private

  def validate_csv
    return true if @csv.blank? || @csv.content_type ==
'text/csv'

    errors[:csv] << 'is invalid'
    false
  end

  def store_data
    csv_text = File.read(@csv.path)
    csv = CSV.parse(csv_text, :headers => true)
    ActiveRecord::Base.transaction do
      csv.each do |row|
        hashed_row = row.to_hash
        HistoricalHarvest.create(
          culture: hashed_row['sowing'].downcase,
          year:                                    "01-01-
#{hashed_row['end_year']}".to_date,
          value: hashed_row['harvest'].to_f,
          user: @user
        )
        Plan.create(
          culture: hashed_row['sowing'].downcase,
          start_year:                              "01-07-
#{hashed_row['start_year']}".to_date,
```

```
          end_year:                                    "01-07-
#{hashed_row['end_year']}".to_date,
          user: @user,
          dates:  hashed_row['days'].split.map  {  |day|
day.to_date }
        )
      end
      return true
    end

    return false
  end
end
```