ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Адаптивний алгоритм прогнозування попиту на товари харчової промисловості для оптимізації запасів у торговельній мережі»

на здобуття освітнього ступеня магістра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

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

_____ Микита КОДЕНЦЕВ
(підпис)

Виконав: здобувач вищої освіти групи ПДМ-63
Микита КОДЕНЦЕВ

Керівник: Ірина ЗАМРІЙ
д-р техн. наук, професор

Рецензент: _____
науковий ступінь, вчене звання *Ім'я, ПРІЗВИЩЕ*

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Магістр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Коденцеву Микиті Ігоровичу

1. Тема кваліфікаційної роботи: «Адаптивний алгоритм прогнозування попиту на товари харчової промисловості для оптимізації запасів у торговельній мережі»

керівник кваліфікаційної роботи Ірина ЗАМРІЙ, доктор технічних наук, професор,

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «30» жовтня 2025 р. № 467.

2. Строк подання кваліфікаційної роботи «19» грудня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, мурашиний алгоритм.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Дослідження методів прогнозування об’єктів в системі.
2. Аналіз технологій глибокого навчання для прогнозування.
3. Розробка та валідація методу прогнозування на базі глибокого навчання.

5. Перелік ілюстративного матеріалу: *презентація*

1. Методи прогнозування.
2. Математична оцінка точності прогнозування для глибокого навчання.
3. Схема методу глибокого навчання нейромереж.
4. Схема методу прогнозування через мурашиний алгоритм.
5. Демонстрація роботи методу на реальних прикладах

6. Дата видачі завдання «31» жовтня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	31.10-07.11.2025	
2	Вивчення матеріалів для аналізу розвитку методів прогнозування попиту	08.11-13.11.2025	
3	Дослідження методів прогнозування попиту	14.11-17.11.2025	
4	Аналіз особливостей впливу факторів на прогнозування попиту	18.11-22.11.2025	
5	Дослідження технологій прогнозування	23.11-28.11.2025	
6	Застосування мурашиного алгоритму в алгоритмах прогнозування попиту	29.11-09.12.2025	
7	Оформлення роботи: вступ, висновки, реферат	10.12-15.12.2025	
8	Розробка демонстраційних матеріалів	16.12-19.12.2025	

Здобувач вищої освіти _____ Микита КОДЕНЦЕВ
(підпис)

Керівник
кваліфікаційної роботи _____ Ірина ЗАМРІЙ
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 71 стор., 2 табл., 28 рис., 30 джерел.

Мета роботи – підвищення ефективності управління запасами торговельної мережі шляхом розробки адаптивного алгоритму прогнозування попиту на основі нейронної мережі та мурашиного алгоритму.

Об'єкт дослідження – процес прогнозування попиту та управління запасами товарів харчової промисловості у торговельній мережі.

Предмет дослідження – адаптивний алгоритм прогнозування попиту з використанням нейронної мережі та мурашиного алгоритму.

Короткий зміст роботи: у роботі використано різноманітні методи, такі як мурашиний алгоритм, алгоритм через нейромережі, алгоритми глибокого навчання.

Проведено аналіз сучасних методів прогнозування попиту на товари.

Розроблено та оптимізовано алгоритм прогнозування на базі мурашиного алгоритму.

Проведено експерименти для оцінки стабільності роботи мурашиного алгоритму та його порівняння з алгоритмом глибокого навчання.

КЛЮЧОВІ СЛОВА: ТОРГІВЛЯ, ПОПИТ, ПРОГНОЗУВАННЯ, АЛГОРИТМ, НЕЙРОМЕРЕЖІ, ГЛИБОКЕ НАВЧАННЯ, МУРАХА, МУРАШИНИЙ АЛГОРИТМ, СИСТЕМА.

ABSTRACT

Text part of the master's qualification work: pages, pictures, tables, sources.

The purpose of the work is to evaluate the use of the ant demand forecasting algorithm.

Object of research – increase the efficiency of the demand forecasting algorithm for goods using a deep learning method based on ant algorithm methods.

Subject of research – algorithms for forecasting demand for goods using artificial intelligence with deep learning algorithms based on gradient descent and ant algorithm.

Summary of the work:

Various methods are used in the work, such as an ant algorithm, an algorithm through neural networks, and deep learning algorithms.

An analysis of modern methods of forecasting the demand for goods was carried out.

A forecasting algorithm based on the ant algorithm was developed and optimized.

Experiments were conducted to assess the stability of the ant algorithm and compare it with the deep learning algorithm.

KEYWORDS: TRADE, DEMAND, FORECASTING, ALGORITHM, NEURAL NETWORKS, DEEP LEARNING, ANT, ANT ALGORITHM, SYSTEM.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

АРКС – авторегресія та ковзне середнє.

AP – Середня точність (Average Precision).

BN – Нормалізація пакету (Batch Normalization).

CNN – Згорткова нейронна мережа (Convolutional Neural Network)

DNN – Глибока нейронна мережа (Deep Neural Network).

ІІС – Інтелектуальна інформаційна система

KNN – метод k-найближчих сусідів (K Nearest Neighbors)

CLR – Common Language Runtime

БД – База даних

ВСТУП

У сучасному світі, де ринки змінюються швидше, ніж прогнози погоди, здатність передбачати попит — це не просто перевага, а життєва необхідність для бізнесу. Люди змінюють свої вподобання, тренди злітають і зникають, а даних — більше, ніж часу на їх аналіз. У такій реальності старі методи вже не працюють. Потрібні нові підходи — гнучкі, адаптивні, здатні вловити неочевидне.

Алгоритми прогнозування попиту — це інструмент, який дозволяє бізнесу не просто реагувати, а діяти на випередження. Вони допомагають краще планувати виробництво, уникати перевантажень складів, зменшувати втрати і, головне, — розуміти своїх клієнтів. Але створити справді ефективний алгоритм — це не лише про математику. Це про розуміння ринку, поведінки людей, сезонності, трендів і навіть несподіванок.

Особливу актуальність ця тема набуває в умовах нестабільності — економічної, соціальної, геополітичної. Коли бізнес змушений адаптуватися до нових реалій, здатність прогнозувати попит стає запорукою виживання. Саме тому дослідження в цій сфері має не лише прикладне, а й стратегічне значення.

Ця магістерська робота присвячена розробці алгоритму прогнозування попиту, який здатен враховувати складні взаємозв’язки, адаптуватися до змін і давати точні результати в реальних умовах. Мета дослідження — створити модель, яка буде не просто працювати, а допомагати бізнесу приймати розумні рішення, зберігаючи баланс між точністю, гнучкістю та практичністю.

Обраний підхід поєднує сучасні методи машинного навчання з класичними принципами статистики, що дозволяє досягти високої точності навіть у складних сценаріях. У процесі дослідження буде проаналізовано реальні дані, протестовано різні моделі та оцінено їхню ефективність у контексті змінного ринку.

Мета роботи – підвищення ефективності управління запасами торговельної мережі шляхом розробки адаптивного алгоритму прогнозування попиту на основі нейронної мережі та мурашиного алгоритму.

Об’єкт дослідження – процес прогнозування попиту та управління запасами товарів харчової промисловості у торговельній мережі.

Предмет дослідження – адаптивний алгоритм прогнозування попиту з використанням нейронної мережі та мурашиного алгоритму.

Для виконання роботи основний акцент зроблено на:

1. Огляд тематичної літератури для отримання знань про практичні результати, що вже були отримані, для покращення чи перевірки їх роботи в рамках прогнозування попиту;
2. Експериментальні дослідження, що покажуть актуальність та певні ризики від виконаної роботи.
3. Моделювання та програмування алгоритму, для отримання реальних значень виконаної роботи.
4. Аналіз результатів для виведення певних висновків щодо отриманого алгоритму.
5. Порівняльний аналіз існуючих алгоритмів та створеного для виявлення ефективності та продуктивності алгоритмів.
6. Оптимізація та удосконалення алгоритму для зменшення можливих ризиків та проблем в роботі алгоритму.

Вирішення цих задач сприяє аналізу, тестуванню та розвитку різних підходів у дослідженні та розробці методів прогнозування попиту.

Методи дослідження: аналіз, синтез, моделювання, спостереження, порівняння, експеримент.

Наукова новизна отриманих результатів: отриманий алгоритм демонструє більш швидке та стабільне навчання алгоритму без втрати точності та без здатності до перенавчання, що позитивно впливає на швидкість адаптації алгоритму до реальної задачі та, відповідно, до швидкості його інтеграції у систему.

Практичне значення одержаних результатів: алгоритм може бути використаний організаціями чи магазинами, яким необхідно інтегрувати в систему алгоритм для прогнозування попиту на товари чи послуги.

1.1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.2 Аналіз предметної області

Торгівля є невід'ємною частиною нашого життя. Раніше єдиним видом торгівлі був бартер (обмін товарами або послугами без використання грошей), тоді як зараз існують обмін (обмін рівноцінними матеріальними цінностями), бартер, купівля-продаж (коли одна зі сторін, що зветься покупцем, замість рівноцінного товару віддає визначену продавцем, ціну у вигляді грошей) і багато інших видів торгівлі [1].

Якщо подивитись на це питання з більш філософської точки зору, купівлею-продажем можна назвати будь-який аспект нашого життя, починаючи оцінками в школі, романтичними стосунками і закінчуючи виконанням робочих завдань перед керівником.

В повсякденному житті купівля-продаж зайняла одну з основних ролей у життєдіяльності в соціуму та досить сильно відтіснила інші типи торгівлі, а обсяги подібних операцій перевалили за мільйони в день. Один із таких прикладів – це магазини, кіоски та бутіки, як офлайн, так і онлайн.

Для вирішення таких проблем була створена дисципліна маркетингу, що поєднує в собі такі науки, як психологія, економіка, дизайн, аналітика та багато інших дисциплін [24]. Маркетинг створений для збільшення уваги покупця на товарі чи послузі, що пропонує компанія.

В сучасному світі маркетологи є одним із ключових елементів у продажі. Вони створюють рекламну кампанію товарів чи послуг, аналізують дані та виходячи з того, що саме цікаво клієнту, обирає шлях для роботи магазину.

Проте, це дуже клопітлива робота, яку виконують більшість часу.

Полегшення та зменшення виконання рутинних задач в сфері аналітики зможе зменшити та покращити інші процеси. Одним із таких процесів є

прогнозування – процес передбачення ймовірного майбутнього на основі попередньо отриманих фактів [25].

На жаль, використання штучного інтелекту не можна назвати панацеєю для магазинів, оскільки це лише алгоритм. У використанні штучного інтелекту можна виділити наступні проблеми, як на етапі розробки, так і на етапі експлуатації:

- точність налаштування, а саме, штучний інтелект – це математичний алгоритм, який будується на базі математичних обчислень, тому точність цих обчислень повністю залежить від чітких та оптимальних коефіцієнтів, підбір яких є доволі складним процесом;
- штучний інтелект спочатку потребує навчальних і тестових даних для калібрування та підвищення точності;
- на початку роботи необхідна наявність оператора, який контролюватиме результати й допомагатиме системі коригуватися.

Використання штучного інтелекту має вагомі недоліки, що вплинуть на бізнес в перший період використання, проте цей вплив з часом буде поступово зменшуватись, через що, в один момент вплив стане зростати.

З книги «Теорія алгоритмів» було вивчено принцип оцінки складності алгоритмів, оскільки такий принцип обов’язково до використання під час оптимізації алгоритму. У випадку практичної роботи, необхідно оцінити, наскільки оптимізація мурашиного алгоритму вплине на його продуктивність [22].

1.3 Нейронні мережі

Нейронна мережа — це структурована система, що складається з великої кількості взаємопов’язаних між собою елементів, які називаються штучними нейронами. Кожен нейрон виконує просту математичну операцію, але в сукупності вони здатні моделювати складні залежності та приймати рішення на основі вхідних даних. Взаємодіючи між собою, ці нейрони формують багаторівневу архітектуру, яка дозволяє системі обробляти, аналізувати та інтерпретувати інформацію, поступово наближаючись до бажаного результату.

Кожен окремий нейрон можна уявити як математичну функцію, що приймає на вхід певний набір значень, обробляє їх із урахуванням відповідних вагових коефіцієнтів (які визначають важливість кожного входу), і видає певне значення на виході. Ці ваги не є фіксованими — вони змінюються в процесі навчання мережі, адаптуючись до даних і поступово зменшуючи похибку між очікуваним і фактичним результатом.

Формально, вихід нейрона можна обчислити за формулою:

$$f = \sum_{i=0}^{n} \quad x_i * w_i, \tag{1.1}$$

де f – результат розрахунку в нейроні,

x_i – вхідні дані з і-то зв’язку,

w_i – коефіцієнт і-го зв’язку.

На рисунку 1.1 представлено графічне зображення нейрону.

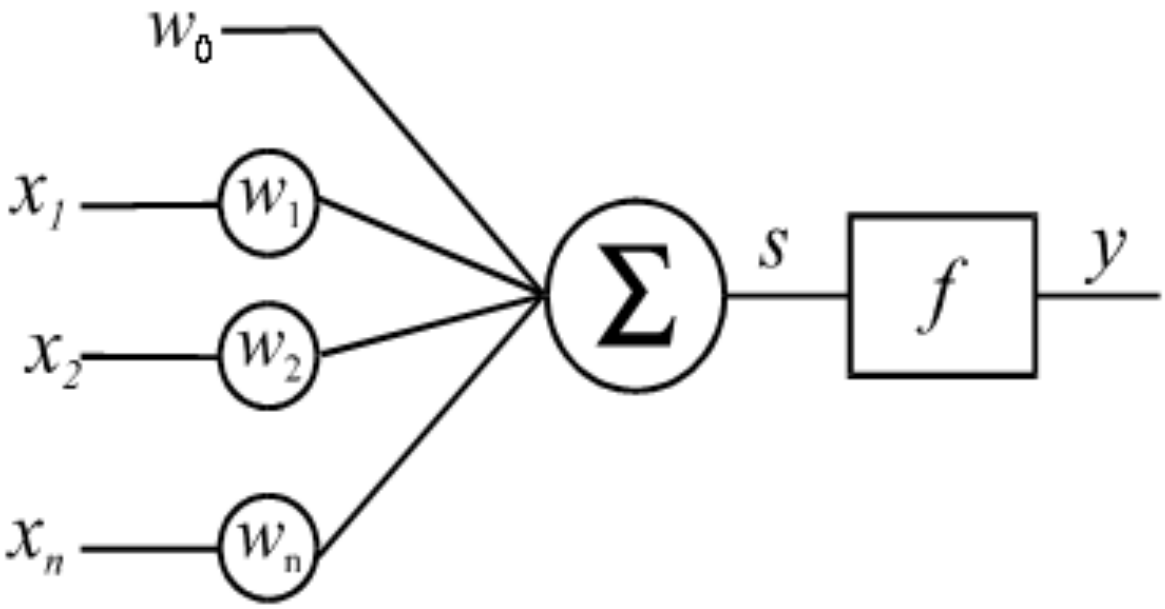


Рис. 1.1 Графічне представлення нейрону

Нейронна мережа складається з кількох послідовних рівнів, кожен з яких виконує свою специфічну функцію в процесі обробки даних. Стандартна архітектура включає три основні типи рівнів: вхідний, приховані та вихідний. Вхідний рівень приймає початкові дані — це може бути числова інформація, зображення, текст або будь-який інший формат, який система здатна обробити.

Далі ці дані передаються до одного або кількох прихованих рівнів, де відбувається основна обчислювальна робота: нейрони аналізують інформацію, застосовують ваги, активаційні функції та формують проміжні результати. Нарешті, вихідний рівень збирає ці результати та видає фінальний прогноз, класифікацію або іншу форму відповіді.

Цей принцип побудови нейронної мережі не випадковий — він був натхненний біологічною моделлю роботи мозку. У живих організмах нейрони передають сигнали один одному, формуючи складні мережі, які дозволяють обробляти сенсорну інформацію, приймати рішення, запам’ятовувати та адаптуватися до змін. Саме ця здатність до навчання і стала основою для створення штучних нейронних мереж, які імітують поведінку біологічних нейронів, але в цифровому середовищі.

Таким чином, нейронна мережа в комп’ютерній системі виконує роль обчислювального ядра, здатного самостійно аналізувати дані, виявляти закономірності та формувати висновки. Її багаторівнева структура забезпечує гнучкість і масштабованість, що дозволяє застосовувати нейромережі в найрізноманітніших сферах — від медицини до фінансів, від розпізнавання образів до прогнозування поведінки користувачів.

На рисунку 1.2 представлено графічне зображення нейромережі.

Рис. 1.2 Графічне представлення нейронної мережі

1.4 Глибоке навчання

Глибоке навчання є спеціалізованим напрямом у межах ширшої галузі машинного навчання. Воно базується на використанні багаторівневих штучних нейронних мереж, які здатні самостійно виявляти складні закономірності у великих обсягах даних. Цей підхід активно застосовується в сучасній промисловості для вирішення різноманітних практичних задач, серед яких — комп’ютерний зір, автоматична обробка природної мови, розпізнавання мовлення, а також прогнозування поведінки систем або користувачів на основі історичних даних.

На відміну від класичних алгоритмів, глибоке навчання не є окремим способом вирішення задач, а радше інструментом, який дозволяє суттєво покращити ефективність роботи нейронних мереж. Його сила полягає в здатності моделі самостійно адаптуватися до змін у даних, без потреби в ручному налаштуванні параметрів або втручанні з боку людини. Це означає, що система, побудована на основі глибокого навчання, може навчатися на власних помилках, поступово вдосконалюючи якість своїх прогнозів або класифікацій.

Такий підхід особливо корисний у складних сценаріях, де дані мають високу варіативність або містять приховані залежності, які важко виявити традиційними методами. Глибоке навчання дозволяє моделі не просто реагувати на вхідну інформацію, а й формувати узагальнене уявлення про структуру даних, що значно підвищує її здатність до перенесення знань на нові, раніше невідомі приклади.

У підсумку, глибоке навчання — це не магія, а результат еволюції алгоритмів, які навчилися краще бачити, чути і передбачати, використовуючи багаторівневу логіку та самонавчання. Його роль у сучасних системах — не замінити людину, а дати їй інструмент, здатний працювати з даними на глибшому рівні, ніж будь-коли раніше.

На рисунку 1.3 представлено зображення розділення нейромережевих технологій на шари.

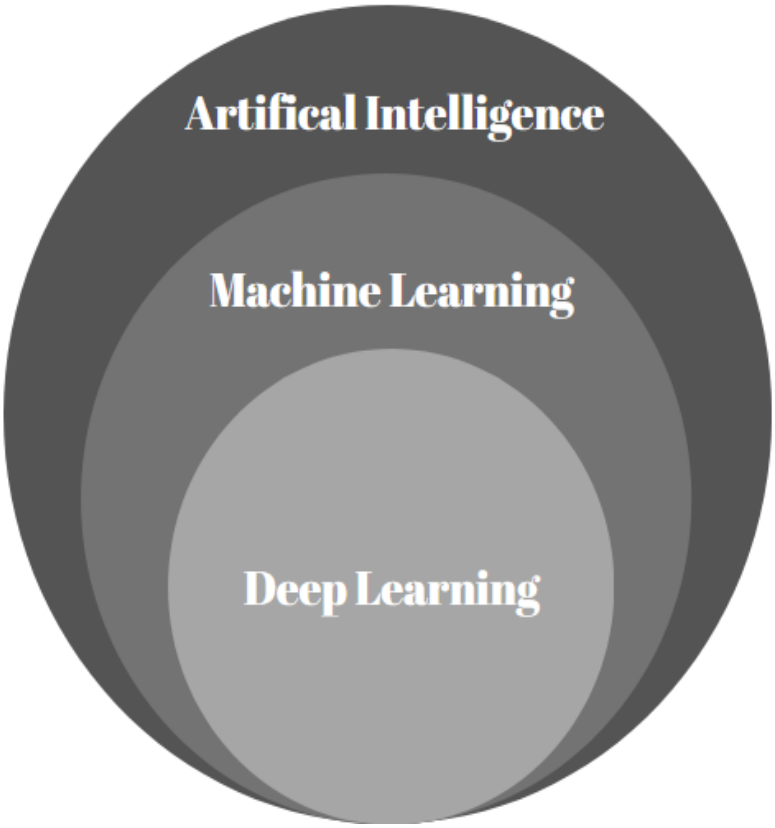


Рис. 1.3 Візуальне представлення складових штучного інтелекту

Без перебільшення можна стверджувати, що саме методи глибокого навчання є ключовими у вирішенні більшості завдань, пов’язаних зі штучним інтелектом. Хоча нейронна мережа є основною структурною одиницею таких систем, сама по собі вона є лише математичною конструкцією — набором формул, об’єднаних у вигляді багаторівневої матриці. Без навчання ця структура не має жодної практичної цінності: вона не здатна приймати обґрунтовані рішення чи робити прогнози.

Справжню силу нейромережі отримують саме завдяки методам глибокого навчання. Ці методи дозволяють автоматично змінювати вагові коефіцієнти зв’язків між нейронами, поступово адаптуючи модель до конкретного завдання. У процесі навчання мережа аналізує вхідні дані, порівнює свої результати з очікуваними, і на основі цієї різниці коригує внутрішні параметри. Саме завдяки цьому механізму вона здатна «вчитися» — тобто покращувати свою здатність до узагальнення, класифікації або прогнозування.

Таким чином, глибоке навчання не просто активує нейромережу — воно перетворює її з пасивної математичної структури на динамічну систему, здатну адаптуватися до нових даних і вирішувати складні завдання без прямого втручання людини. Одним з найкращих прикладів методу глибокого навчання є метод градієнтного спуску. Цей метод полягає в постійній квадратичній зміні помилки, наче з гори.

На рисунку 1.4 представлено графічний процес навчання методу градієнтного спуску.

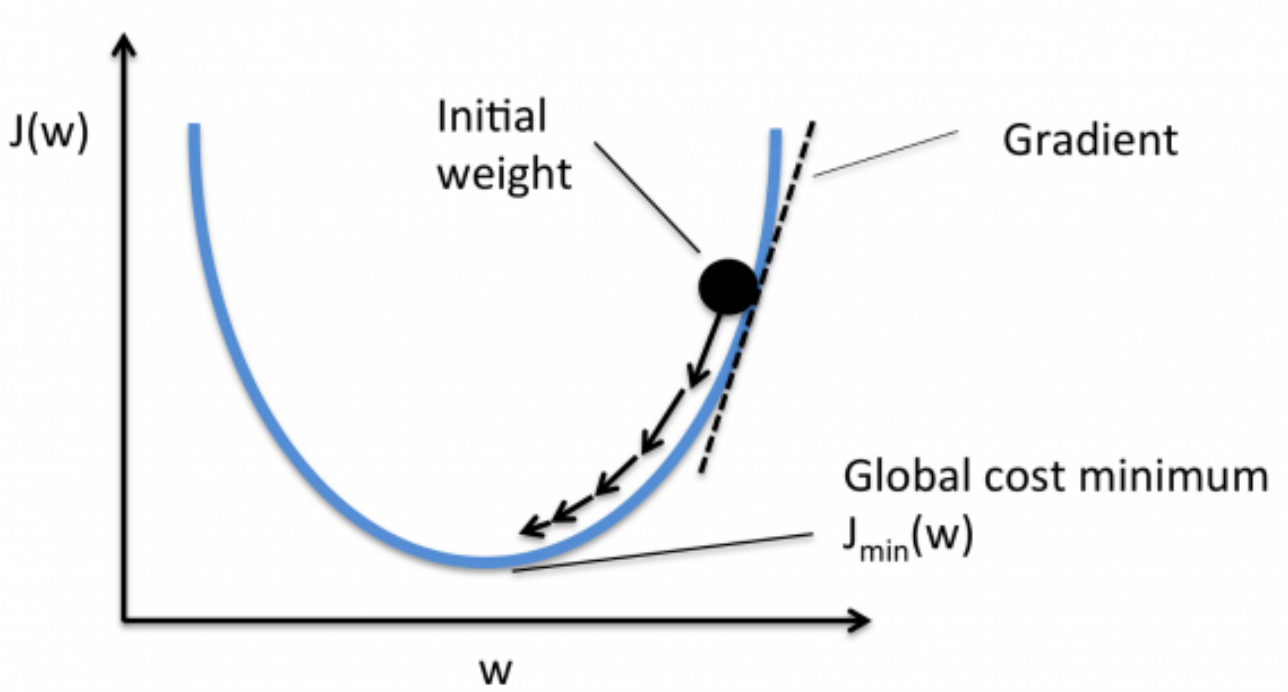


Рис. 1.4 Візуальне представлення градієнтного спуску

1.5 Мурашиний алгоритм

Для виконання завдання було обрано модель мурашиного алгоритму оптимізації [23]

Принцип роботи мурашиного алгоритму дуже простий:

1. Створюється покоління мурах.
2. Пошук найменшого шляху до цілі (в нашому випадку – найменшу похибку).
3. Мурахи залишають феромони під час переміщення.
4. Завершення циклу.

5. Створюється нове покоління мурах та повторюються кроки 2-4.

Принцип роботи алгоритму добре представлено у роботі Горносталя О. та Дорогого Я [10].

При цьому, хотілось би зазначити, що феромони, що залишені мурахами, з часом зникають, тобто, чим частіше мурахи рухаються в одному шляху, тим більше феромонів буде залишатись на ньому, а, отже, в подальшому ще більше мурах буде переміщатись по цьому шляху, тобто, похибка з кожним проходом буде геометрично менше.

На рисунку 1.5 представлено візуальне представлення руху мурах в мурашиному алгоритмі.

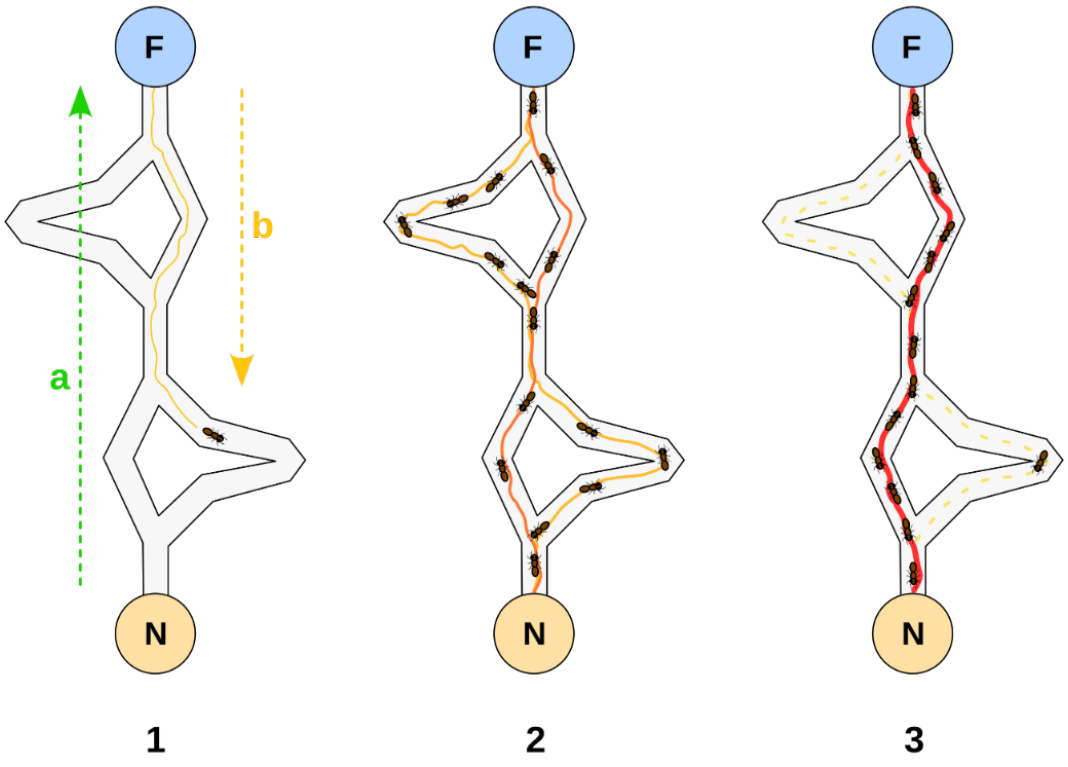


Рис. 1.5 Візуальне представлення мурашиного алгоритму

В основі мурашиного алгоритму знаходиться формула:

$$P_i = \frac{l_i^q * f_i^p}{\sum_{k=0}^{N} l_k^q * f_k^p}, \tag{1.2}$$

де P_i – ймовірність проходу шляхом i;

l_i – величина, обернена до ваги i-го переходу;

f_i – кількість феромонів на i-му шляху;

q – величина, що визначає жадібність алгоритму;

p – величина, що визначає «стадність алгоритму»

При цьому має виконуватись рівняння $q + p = 1$.

У випадку з попитом, шукається не шлях, а коефіцієнти важливості для виконання формули прогнозування. Це не має ніяк вплинути на виконання методу, оскільки в задачі існують базові коєфіцієнти (від 0 до 1) і оцінка попиту.

1.6 Огляд актуальних наукових робіт у науковій галузі

У статті Smart Decision HUB прогнозування попиту описується як важливий процес, що полягає в оцінці майбутнього попиту на основі аналізу історичних даних, інформації та впливу різних додаткових факторів. Ефективне прогнозування попиту надає компаніям цінну інформацію про можливості на поточних і потенційних ринках, що допомагає менеджерам приймати зважені рішення щодо обсягів замовлень, стратегій просування товарів та загального управління бізнесом. Це, в свою чергу, сприяє оптимізації ресурсів і підвищенню конкурентоспроможності компанії [2].

У фрагменті роботи Овсієнка О.М. «Прогнозування попиту та пропозиції послуг підприємства» підкреслюється значущість науково обґрунтованого підходу до прогнозування. Це передбачення можливого майбутнього стану підприємства є критично важливим для стратегічного планування та управління. Прогнозування дозволяє компаніям адаптувати свої стратегії у відповідь на зміни в попиті та пропозиції, що, в свою чергу, сприяє підвищенню їхньої ефективності та конкурентоспроможності на ринку [16].

У роботі Порадюка Т. «Інтелектуальні системи для прогнозу споживання альтернативної енергії.» описано принцип прогнозування споживання альтернативної енергії. В роботі описано принцип прогнозування та алгоритми, які використовуються в області прогнозування, що є досить суттєвим внеском для розуміння роботи [3].

Оскільки різниця подібної системи з системою, що буде розроблена, лише у вхідних даних, для розуміння принципу побудови це має великий внесок для вирішування задачі, хоча й не прямий.

В своїй роботі «Інтелектуальна система прогнозування вартості комерційних компаній» Мальченко. П.О. описує часові ряди, які є невід'ємною частиною будь-якого методу прогнозування, наступним чином: Дані часового ряду – це послідовність числових спостережень, природно впорядкованих у часі [11].

Ця інформація є цінною в проекті, оскільки будь-яке прогнозування базується на основі часового ряду, оскільки це є точною та стабільною точкою в розрахунках. Для прогнозування нам необхідно не лише розуміння, які цифри мають бути в результаті, а й коли саме вони мають бути (в який часовий період) та тенденцію їх росту.

Також, акцент на часові ряди вказує у своїй роботі і Фінажин М.Ф., що підтверджує те, що роль часових рядів у прогнозуванні є ключовою [5].

Вікіпедія зазначає, що задача оптимізації — це процес, спрямований на виявлення найкращого можливого рішення серед усіх можливих варіантів, що відповідають певним критеріям. У контексті математичного аналізу, оптимізація часто зводиться до пошуку точки або точок екстремуму (максимуму або мінімуму) заданої функції [7].

Так як в роботі виконується саме прогнозування попиту, що є задачею оптимізації, в першу чергу, необхідно розуміти саме термінологію цього принципу.

Оптимізація — це процес знаходження найкращого рішення серед множини можливих варіантів. Вона може включати мінімізацію або максимізацію функції за певних умов [6].

В рамках практичної роботи результатом вважається коректний прогноз попиту товару, проте, для роботи проекту основною .

У роботі Сізової Н.Д. можна зрозуміти її бачення визначення «Інтелектуальної системи».

Інтелектуальні інформаційні системи (ІІС) підтримки ухвалення рішень представляють собою комплекс програмних, лінгвістичних, логіко-математичних засобів та, що важливо, сучасних технологій машинного навчання. Їхнє основне завдання – підтримка діяльності людини шляхом надання можливості проводити поширений діалог на природній мові, а також шляхом забезпечення функціоналу для пошуку, аналізу даних, генерації рекомендацій та автоматизації процесів ухвалення рішень [13].

Також, ІІС гарно описуються в роботах Бідюка П. та Савченко А., «Методи інтелектуального аналізу даних» [12] та Доброї Н., Корнілової Є. та Самохіної Ж., «Інтелектуальні інформаційні технології та системи» [14].

В своїй роботі «Аналіз існуючих методів прогнозування попиту та способів оцінки їх якості» Яновський Д. та Граф М. описують список методів для прогнозування попиту, що є основним рушієм для роботи [4].

Оскільки задачею проекту є оптимізація розробки системи, оцінка інших алгоритмів є ключовим фактором для роботи.

Також, про методи прогнозування рівня попиту описано в роботі Заруби В. «Методи прогнозування рівня попиту» [15].

Кормен Т. у роботі «Алгоритми. Побудова та аналіз» описує принципи побудови алгоритмів та їх аналізу, що є основною задачею алгоритмізації та оптимізації [18].

Ця робота обов’язкова для оцінки, оскільки обраний алгоритм має певні складності в побудові. Оцінка виконання алгоритму показує, наскільки алгоритм придатний до використання, а отже, для того, щоб алгоритм був максимально продуктивним, його буде необхідно оцінювати та корегувати під необхідні реалії.

Величко О., Гордієнко Т. в своїй роботі «Основи системного аналізу і прийняття оптимальних рішень» описують принцип системного аналізу та ухвалення рішень. Це є частиною основного функціоналу повного алгоритму прогнозування, оскільки для повної картини необхідно максимально абстрагувати людину від ухвалення рішення власноруч. Робота має великий вплив для подальшого розвитку проекту, проте, не слід забувати, що системи

прогнозування часто бувають не точні та некоректні, а отже для максимально коректного процесу автоматизованого ухвалення рішень слід надати системі час на максимальну оптимізацію [20].

«Математичні методи економічного аналізу. Навчальний посібник.» Приймака В. представляє процес прогнозування з математичного боку. Це має сенс, оскільки робота прогнозування, перш за все, представляє собою саме математичну задачу, а не алгоритмічну [21].

Булгакова О., Зосімов В., Поздєєв В. описують у «Методи та системи штучного інтелекту: теорія та практика.» сам принцип роботи штучного інтелекту.

Для цілі роботи штучний інтелект в його стандартному розумінні не має такого сенсу, оскільки такі алгоритми мають певні проблеми зі складністю та швидкістю навчання, проте, оцінюючи вплив штучного інтелекту та кількість рішень саме через подібний підхід, доцільно зрозуміти, наскільки обраний метод вирішення задачі буде оптимальнішим за попередні [19].

Для роботи ми будемо використовувати мурашиний алгоритм. Його ключові аспекти гарно описують Тимчук, Проценко та Парамонов у своїй статті [9].

Генетичний алгоритм – це еволюційний метод пошуку, який застосовується для розв'язання задач оптимізації та моделювання. Він працює шляхом підбору, комбінації та варіювання параметрів, імітуючи біологічні процеси еволюції [8].

Маршрут. Як і в класичному алгоритмі, один мураха – це простий агент, який вирішує всю задачу від початку до кінця. Такий самий принцип ми збережемо і зараз – один мураха послідовно проходить шлях спочатку одного комівояжера, потім “перестрибує” у невідвідане місто і починає проходити шлях другого комівояжера, і так далі, доки не будуть пройдені всі міста. Таким чином, маршрут мурахи складається з “піших переходів” та “стрибків”, причому стрибків рівно m–1, тобто, шлях мурахи складається з m фрагментів, кожен з яких є шляхом одного комівояжера.

Феромон. У класичному алгоритмі реалізується ідея феромону як спосіб непрямої взаємодії між мурахами колонії. У TSP мурахи залишають феромони на переходах (i, j) і коли черговий мураха колонії виявляється в і-й вершині, він вибирає один з наявних переходів з ймовірністю, пропорційною кількості феромону на доступних переходах.

У процесі аналізу було встановлено, що більшість попередніх розробників при побудові системи зосереджувалися на використанні класичних методів машинного навчання, зокрема нейронних мереж. Найчастіше застосовувався базовий одношаровий персептрон — простий, але ефективний інструмент для вирішення задач класифікації та регресії. Такий підхід дозволяв швидко реалізувати базову модель, проте мав обмеження в контексті складних сценаріїв, де потрібна глибша обробка даних і виявлення прихованих закономірностей.

Основна задача, яка стоїть перед системою, полягає в оптимізації алгоритму прогнозування попиту. Це завдання належить до категорії високонавантажених, оскільки передбачає обробку великого обсягу історичних даних, виявлення трендів, сезонних коливань та потенційних аномалій. У таких умовах прості моделі не завжди здатні забезпечити достатню точність або адаптивність.

Для ефективного вирішення цієї задачі необхідно враховувати не лише обсяг даних, а й їхню складність, варіативність та динаміку. Обробка таких даних є ресурсомістким процесом, що вимагає не тільки обчислювальної потужності, а й ретельно підібраної архітектури моделі. Саме тому виникає потреба у переході від базових нейромереж до більш складних структур, зокрема методів глибокого навчання, які дозволяють автоматично адаптувати модель до змін у даних і підвищувати якість прогнозів без ручного втручання.

У межах даної роботи доцільно висловити припущення, що застосування методів генетичних алгоритмів може продемонструвати вищу ефективність у вирішенні задач прогнозування попиту, порівняно з традиційними нейромережевими підходами. Генетичні алгоритми, як еволюційні методи оптимізації, мають здатність до пошуку глобального мінімуму в складних,

багатовимірних просторах, що особливо актуально для задач з великою кількістю параметрів і високим рівнем шуму в даних.

На відміну від нейромереж, які часто потребують ретельного налаштування архітектури та гіперпараметрів, генетичні алгоритми працюють за принципом природного добору: вони генерують популяцію рішень, оцінюють їхню якість, а потім поступово вдосконалюють її шляхом селекції, кросоверу та мутацій. Такий підхід дозволяє уникати локальних мінімумів і забезпечує адаптивність до змін у структурі даних.

У контексті задач прогнозування попиту, де важливо не лише точність, а й стабільність моделі при зміні вхідних умов, генетичні алгоритми можуть забезпечити більш гнучке та надійне рішення. Вони здатні оптимізувати як саму модель, так і її параметри, що відкриває можливість побудови гібридних систем, де генетичні алгоритми використовуються для налаштування нейромереж або інших моделей машинного навчання.

Таким чином, включення генетичних алгоритмів до дослідницького процесу не лише розширює спектр доступних інструментів, а й дозволяє критично переосмислити підходи до побудови прогнозних моделей, орієнтуючись на ефективність, адаптивність і стійкість до змін.

2 АНАЛІЗ МЕТОДІВ ПРОГНОЗУВАННЯ ПОПИТУ НА ОСНОВІ ГЛИБОКОГО НАВЧАННЯ

2.1 Аналіз існуючих методів прогнозування

2.1.1 Класичні методи прогнозування

Основними методами прогнозування можна вважати методи ковзного середнього, експоненційного згладжування, та авторегресії.

Ковзне середнє є одним із найпростіших і водночас ефективних методів прогнозування, який базується на аналізі середніх значень за попередні періоди. Його суть полягає в тому, щоб згладити коливання в даних, обчислюючи середнє значення для певного набору точок, що послідовно зміщується в часі. Такий підхід дозволяє виявляти загальні тенденції, зменшувати вплив випадкових флуктуацій і формувати більш стабільну картину змін.

Кількість точок, що враховуються при обчисленні ковзного середнього, може бути фіксованою або змінюваною — залежно від потреб користувача чи специфіки задачі. Наприклад, коротке вікно (менша кількість точок) дозволяє швидше реагувати на зміни, але є більш чутливим до шуму, тоді як довше вікно забезпечує плавніший тренд, але може втрачати деталі. Така гнучкість робить метод ковзного середнього придатним для широкого спектра застосувань.

Варто зазначити, що ковзне середнє, як і більшість статистичних методів прогнозування, не дає абсолютно точного результату. Його призначення — надати приблизну оцінку майбутніх значень на основі минулих даних. Це особливо корисно в задачах, де важливо не передбачити конкретне число, а зрозуміти загальну динаміку або виявити циклічні закономірності.

Найчастіше ковзне середнє застосовується для аналізу часових рядів — наприклад, у фінансових даних, продажах, температурних вимірюваннях тощо. Воно допомагає згладити раптові стрибки, підкреслити сезонні цикли та зробити дані більш придатними для подальшої обробки. У цьому сенсі ковзне середнє

виконує роль фільтра, який очищає дані від шуму, а не повноцінного прогнозного механізму.

Саме тому ковзне середнє часто використовується в комбінації з іншими методами — наприклад, нейронними мережами, регресійними моделями або генетичними алгоритмами. Воно може слугувати як попередній етап обробки, що покращує якість вхідних даних і підвищує точність складніших моделей.

Процес ковзного середнього може обчислюватись від довільних даних, але, найчастіше його використовують для аналізу часових рядів чи зглажуванні раптових коливань та підкреслення циклів. Іншими словами, ковзне середнє можна вважати більш фільтром, самим методом прогнозування, що може добре спрацювати в комбінації з іншими методами.

Під час виконання методу прогнозування можливо виставити також і коефіцієнти важливості змінних для виставлення пріоритету даних під час оцінювання.

Математично класичне ковзне середнє можна представити формулою:

$$f_t = \sum_{i=n}^{m} a_i * x_{i+t}, \quad (2.1)$$

де n – початок часового ряду,

m – кінець часового ряду,

a_i – коефіцієнт ваги i,

x_{i+t} – значення ваги $i + t$.

На рисунку 2.1 представлено приклад ковзного середнього.

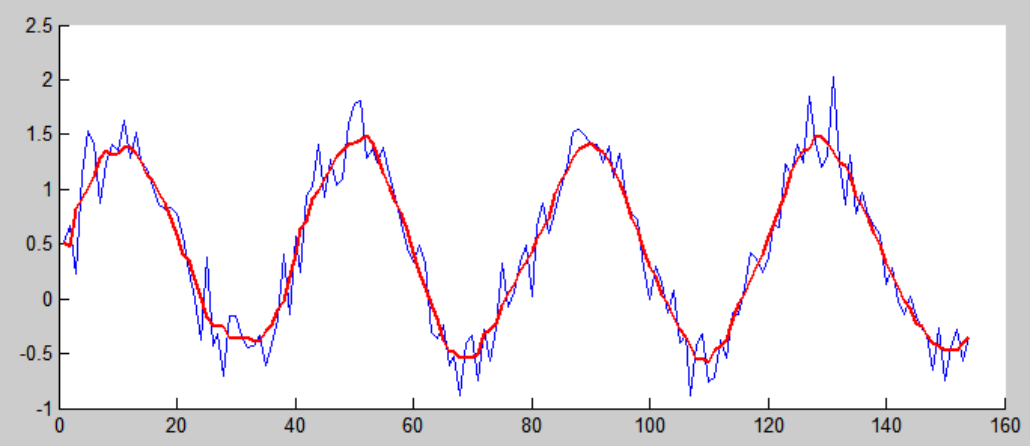


Рис. 2.1 Візуальне представлення ковзного середнього

Метод експоненційного згладжування – це метод прогнозування, що базується на принципі згладжування значень, але, на відміну від ковзного середнього, експоненційне згладжування враховує всі значення ряду, проте чим старіше значення, тим менший вплив воно має на прогноз.

Математично, експоненційне згладжування можна представити у вигляді формули:

$$s_t = \{c_1 : t = 1\, s_{t-1} + \alpha * (c_t - s_{t-1}) : t > 1\,, \tag{2.2}$$

де s_t – згладженний ряд,

c_t –первинний ряд,

α –коефіцієнт згладження.

На рисунку 2.2 представлено приклад експоненційного згладжування.

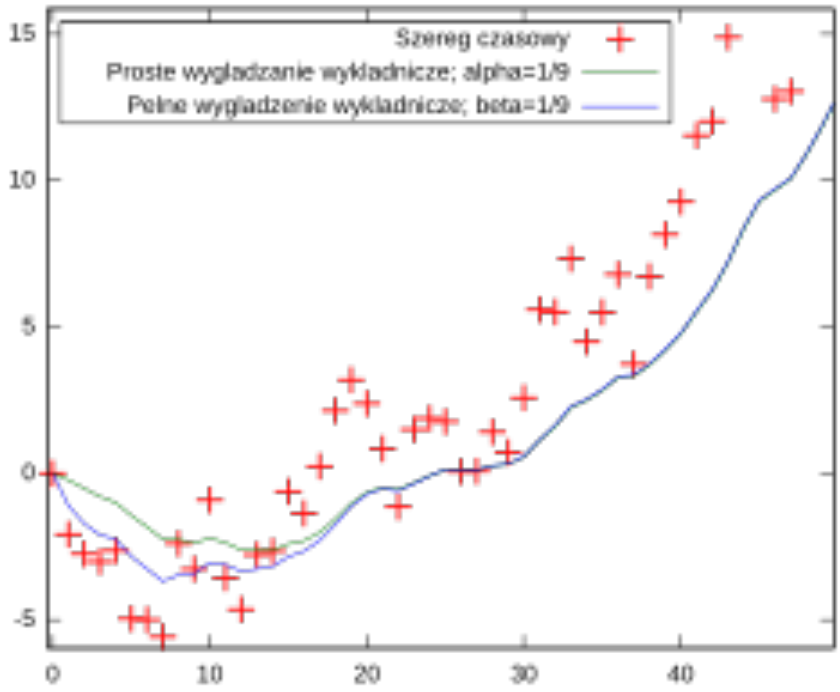


Рис. 2.2 Візуальне представлення експоненційного згладжування

Авторегресія, частіше за все, поєднує два процеси: авторегресію та ковзне середнє, створюючи модель АРКС, що використовується для розуміння та ймовірного передбачення майбутніх значень ряду.

Принцип роботи АРКС доволі простий: авторегресія передбачає регрес змінної за власним спізнюванням, а ковзне середнє передбачає моделювання похибок, що стаються в поточний та минулі моменти.

Формула АРКС може бути представлена, як:

$$X_t = c + \sum_{i=1}^{p} (\varphi_i * X_{t-i}) + \varepsilon_t, \quad (2.3)$$

де $\varphi_1 \dots \varphi_p -$ параметри оцінки,

c −стала величина,

ε_t −білий шум графіку.

На рисунку 2.3 представлено графічне зображення моделі АРКС.

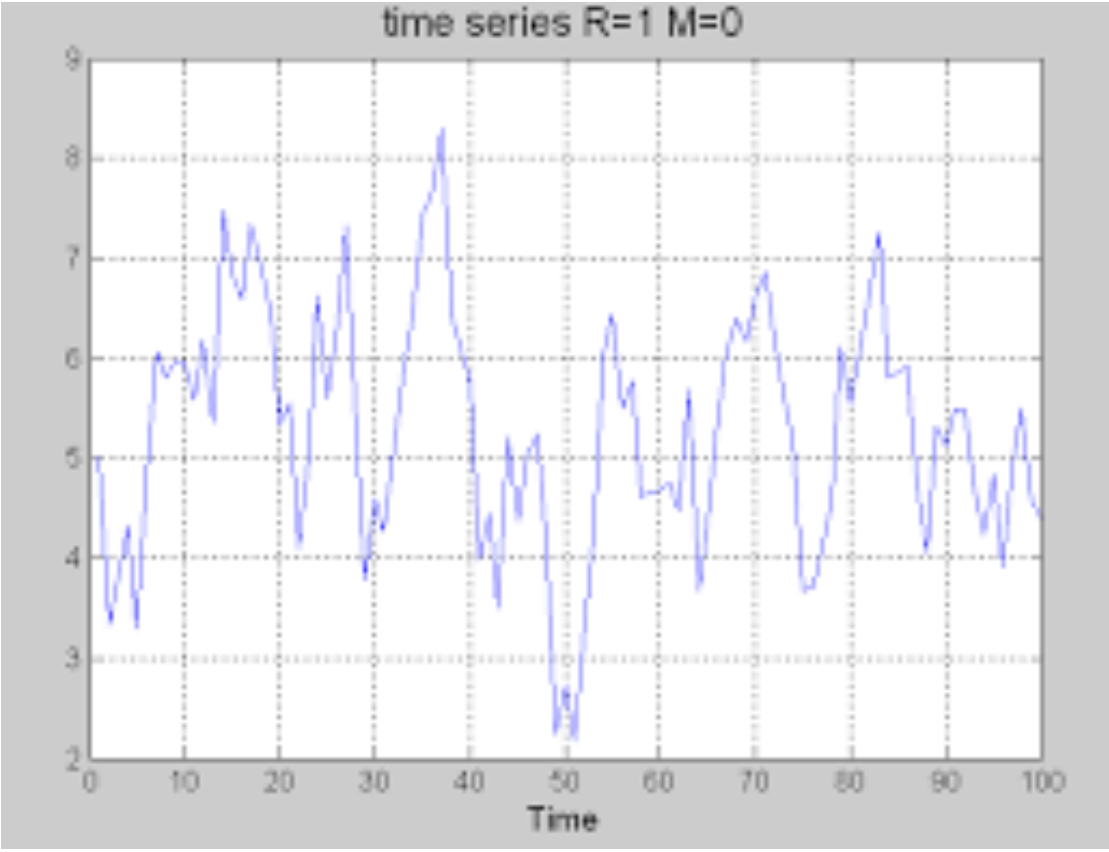


Рис. 2.3 Візуальне представлення АРКС

2.1.2 Методи прогнозування на базі штучного інтелекту

Основними методами прогнозування штучного можна назвати дерева рішень, глибоке навчання, алгоритм KNN та нейромережі.

Дерево рішень – це алгоритм для аналізу даних для прогнозних моделей. Структура дерева рішень складається з гілок, що відповідають вимогам до рішення та листя, що відповідає результату, що видає дерево.

Мета дерева рішень полягає у прогнозуванні значень на основі відповідності певним умовам. В цьому полягає і проблема – назвати дерево рішення цілком штучним інтелектом доволі складно, а рішення базуються не на математичному рішенні, а саме на можливих наслідках, тому для прогнозування попиту використання дерев рішень не має великого сенсу, проте, саме як метод прогнозування, дерево рішень активно використовується, наприклад, коли справа йде про видачу кредиту, оцінки окупності проектів та подібних задачах.

На рисунку 2.4 представлено приклад дерева рішень.

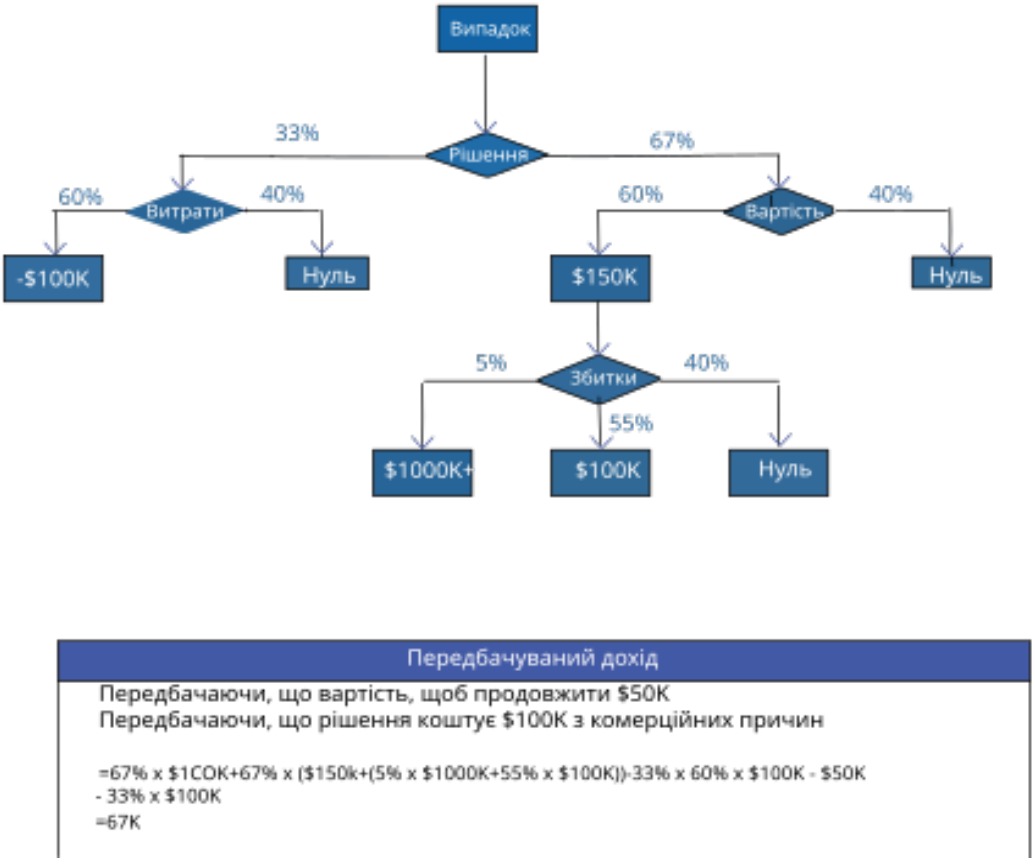


Рис. 2.4 Візуальне представлення дерева рішень

Алгоритм KNN або алгоритм найближчих сусідів – це метод, що пропонує прогнозування на базі класифікації результатів. Як можна зрозуміти з назви, метод оцінює найближчих сусідів та, виходячи з цього, переглядає, до якого результату ближче за все належить шуканий елемент, приписуючи його до них. Таким чином, можна сказати, що алгоритм набагато простіше використовувати у парі з деякими іншими класичними алгоритмами, такими як ковзне середнє чи регресії.

Дуже сильною стороною можна вважати те, що алгоритм може досить гарно опрацьовувати будь-які аномальні значення, таким чином зупиняючи будь які викиди та ситуації з доволі сильними аномаліями.

На рисунку 2.5 представлено приклад методу KNN.

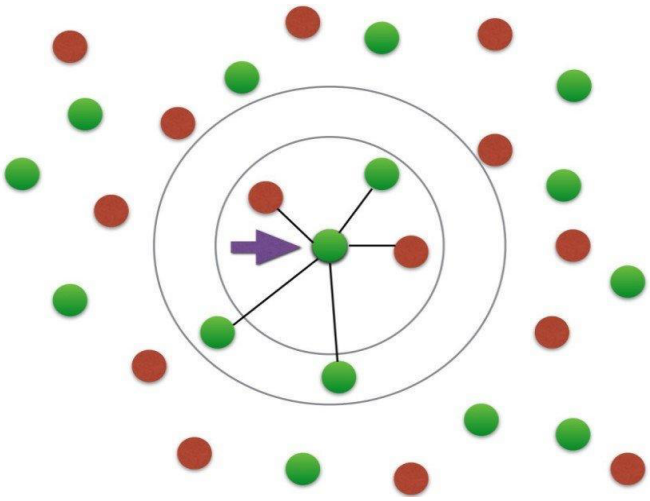


Рис. 2.5 Візуальне представлення методу KNN

Нейронні мережі – метод прогнозування, що використовується частіше за все в задачах прогнозування попиту чи продаж. Як вже було описано раніше, нейронна мережа – це набір нейронів, які, взаємодіючи між собою, представляють результат.

Основний момент у роботі алгоритму нейромереж йде на методах глибокого навчання, таких як градієнтний спуск, методі «гаряче/холодно» та інших. Також, для оптимізації функції прогнозування можна використовувати алгоритми з сімейства генетичних, такі як мурашиний алгоритм, рою бджіл та інші.

2.2 Аналіз переваг та недоліків методів прогнозування

Визначивши основні методи, можна провести аналіз та визначити алгоритм, що буде максимально близько до потреб. Порівняти методи треба з 5-ти основних характеристик.

Ефективність алгоритму – демонструє загальну оцінку алгоритму відносно виконуваної задачі.

Обсяг даних – демонструє, для якої кількості даних можна використовувати алгоритм.

Швидкість – демонструє, наскільки швидкі відповіді можуть видати алгоритми.

Стійкість до викидів демонструє, чи будуть різкі дані впливати на прогноз.

Завдяки цим основним характеристикам можна буде оцінити методи та порівняти їх між собою. Таким чином, можна виділити найбільш привабливий для алгоритму метод. Це дозволить обрати найкращий метод, після чого можливо буде оптимізувати його для покращення швидкості, стійкості чи будь чого ще.

Завдяки перегляду ми можемо зробити певні висновки, що більш привабливими залишаються такі методи, як ковзне середнє та нейронні мережі, проте, багато з методів залишаються доволі однаковими, тому, хотілось би виділити статистичну оцінку кожному методу, для виявлення найбільш ефективного варіанту, який вже буде цікаво та корисно використати за основу та модифікувати.

Після того, як будуть виставлені оцінки, буде проведено підрахунок середньої оцінки ефективності методу, що покаже, який саме алгоритм є найбільш ефективним.

Виходячи з обраного алгоритму, буде побудована математична модель та запропоновано базовий варіант для оптимізації.

Для оцінки обрано 5 основних параметрів з оцінкою за п’ятибальною шкалою, а саме:

– точність – відображає, наскільки точно дані буде спрогнозовано, де 1

– найменш точні, а 5 – найбільш точні дані;

- складність реалізації – відображає, наскільки складно реалізовано алгоритм, де 1 – дуже складно, а 5 – легко;
- вимоги до даних – відображає, наскільки сильно алгоритм залежить від складності даних, де 1 – дуже залежать, а 5 – практично не залежать;
- здатність до узагальнення – відображає можливість узагальнення даних, де 1 – майже не може узагальнювати дані, а 5 – гарна можливість узагальнення;
- стійкість до шуму – відображає, як сильно шум впливає на результати, де 1 – дуже чутливий до шуму, а 5 – практично повністю стійкий.

Статистична оцінка у вигляді числових коефіцієнтів для більш точної оцінки представлено у таблиці 2.1.

Таблиця 2.1

Порівняння методів прогнозування

Назва Хар-ка	Ковзне середнє	Експоненційне згладження	АРКС	Дерево рішень	Метод KNN	Нейронні мережі
Точність	3	2	1	3	3	5
Складність реалізації	5	3	2	3	4	3
Вимоги до даних	2	2	3	2	2	2
Здатність до узагальнення	2	4	4	3	3	4
Стійкість до шуму	3	5	3	2	3	3

Для пояснення можна описати результати у наступному вигляді:

- точність: ковзне середнє забезпечує базовий рівень точності для стаціонарних рядів без тренду/сезонності. Експоненціальне згладжування (особливо holt-winters) дає високу точність для рядів із трендом та сезонністю. АРКС моделі демонструють високу точність

для лінійних процесів з урахуванням зовнішніх факторів. Дерева рішень та KNN забезпечують середню точність, особливо на складних, нелінійних даних, але схильні до перенавчання. Нейромережі досягають найвищої точності на великих, складних даних, але потребують ретельної настройки та великого обсягу навчальної вибірки.

- складність реалізації: ковзне середнє та експоненціальне згладжування – найпростіші у реалізації. АРКС моделі потребують глибшого математичного апарату, перевірки стаціонарності, підбору порядків. Дерева рішень та KNN – середньої складності, але для великих даних потребують оптимізації. Нейромережі – найскладніші, вимагають глибоких знань, обчислювальних ресурсів та часу на навчання.
- вимоги до даних: ковзне середнє та експоненціальне згладжування працюють навіть на коротких рядах, але для складних патернів цього недостатньо. АРКС, дерева рішень, KNN та особливо нейромережі потребують великих, якісних вибірок без пропусків.
- здатність до узагальнення: ковзне середнє не узагальнює, а лише згладжує. Експоненціальне згладжування має обмежену здатність до узагальнення. АРКС, дерева рішень, KNN та нейромережі здатні до узагальнення, але ризик перенавчання зростає зі складністю моделі.
- стійкість до шуму: ковзне середнє та експоненціальне згладжування добре згладжують випадковий шум, але чутливі до викидів. АРКС моделі чутливі до шуму при оцінці параметрів. Дерева рішень та KNN схильні до перенавчання на шумових даних, але ансамблі та регуляризація підвищують стійкість. Нейромережі можуть бути стійкими до шуму за умови достатньої кількості даних та правильної регуляризації.

Таким чином, оцінюючи ефективність відносно коефіцієнтів, можна сказати, що найбільш ефективними методами можна вважати методи ковзного середнього,

KNN та нейромереж, при цьому, нейромережі показують себе найбільш ефективно.

2.3 Математична модель методу прогнозування на базі мурашиного алгоритму

Математична модель допомагає при побудові реальних методів та рішень, які будуть вирішувати проблеми, виведені в науковій роботі. Основними характеристиками математичної моделі можна вважати узагальнення реальності (тобто, абстрагування від реальності для спрощення та можливості виконання роботи відносно неточних даних, а будь-якої ситуації в цілому), формалізація зв’язків між даними для виведення важливих частин та проблемних точок системи, аналіз результатів та ефективність моделі.

Хотілось би зазначити, що математична модель має можливість бути обмеженою, якщо цього потребує дослідження, що дає можливість переглядати будь які ситуації, які можуть виникнути.

У даній роботі передбачається прогнозування попиту на товар в певний проміжок часу. Таким чином, необхідно зазначити, що прогнозування цілком може відноситись до будь-якого числового ряду, з певними корекціями. Так, для прогнозування попиту необхідно отримувати не лише останні дані з попиту, а й сезони, коли саме відбулись минулі продажі, щоб оцінити тенденцію та спрогнозувати результат. Для виявлення прогнозу використовуватиметься нейронна мережа, яка складається з нейронів. Як було показано раніше, нейрон – це формула вигляду:

$$f = \sum_{i=0}^{n} \quad x_i * w_i, \tag{2.4}$$

де x_i – дані на і-му вході,

w_i – ваг і-го зв’язку.

Для обраної нейронної мережі представлено вхідний вектор, що можна описати наступним чином:

$$x \in R^{24}, \tag{2.5}$$

де R^{24} – вхідний вектор із 24 даних (21 вхідні дні + мінімальне значення періоду + максимальне значення періоду + середнє арифметичне періоду).

Вхідний шар можна представити, як:

$$h_1 = \sigma(W_1 x + b_1), \tag{2.6}$$

де σ – функція активації,

x – вхідні дані,

W_1 – ваги, що проходять навчання,

b_1 – параметри зміщення.

Перший прихований шар можно представити, як:

$$h_2 = \sigma(W_2 h_1 + b_2), \tag{2.7}$$

де σ – функція активації,

h_1 – результат розрахунку вхідного шару,

W_2 – ваги, що проходять навчання,

b_2 – параметри зміщення.

Другий прихований шар можно представити, як:

$$h_3 = \sigma(W_3 h_2 + b_3), \tag{2.8}$$

де σ – функція активації,

h_2 – результат розрахунку першого прихованого шару,

W_3 – ваги, що проходять навчання,

b_3 – параметри зміщення.

Вихідний шар можно представити, як:

$$h_4 = \sigma(W_4 h_3 + b_4), \tag{2.9}$$

де σ – функція активації,

h_3 – результат розрахунку вхідного шару,

W_4 – ваги, що проходять навчання,

b_4 – параметри зміщення.

Функція активації, що використовується в роботі системи – ReLU, що

описана у вигляді:

$$ReLU(x) = \{0,\ x \leq 0\ x,\ x > 0\ , \tag{2.10}$$

де x – результат розрахунку нейрону.

Таким чином, нейронна мережа може бути навчена як градієнтним спуском, так і за допомогою мурашиного алгоритму.

Мурашиний алгоритм пропонує пошук необхідного значення (у випадку з нейронними мережами – ті суми вагів, які необхідні для виконання прогнозу), цей алгоритм має бути хорошою альтернативою градієнтного спуску. Формула мурашиного алгоритму виглядає наступним чином:

$$P_i = \frac{l_i^q * p_i^q}{\sum_{k=0}^{N} l_k^q * f_k^q}, \tag{2.11}$$

де $P_i -$ ймовірність переходу до шляху i,

l_i −величина, зворотня вагу шляху i,

f_i −кількість феромону на шляху i,

q −величина жадності алгоритму,

p −величина стайності алгоритму.

Таким чином, побудована математична модель нейронної мережі з використанням функції активації ReLU та альтернативних методів навчання, серед яких градієнтний спуск і мурашиний алгоритм, дозволяє здійснювати прогнозування попиту на товар у визначені часові проміжки. Модель забезпечує формалізацію зв’язків між даними, враховує сезонні коливання та здатна адаптуватися до різних умов ринку.

Застосування мурашиного алгоритму як методу оптимізації вагів відкриває можливості для підвищення точності прогнозів у випадках, коли класичні методи демонструють обмежену ефективність.

У результаті отримана система може бути використана як інструмент для підтримки управлінських рішень, планування виробництва та оптимізації логістики, що підтверджує практичну значимість дослідження.

3 РОЗРОБКА МЕТОДУ ПРОГНОЗУВАННЯ ПОПИТУ НА ОСНОВІ ГЛИБОКОГО НАВЧАННЯ

3.1 Опис розробки методу

Основною метою задачі являється побудова ефективного алгоритму прогнозування попиту на товар, що буде не лише правильно відображати дані а й робити це максимально ефективно та точно.

Оскільки обраним методом був метод через нейронні мережі, вони будуються лише на базі зв’язку примітивних формул, тому, основний напрям роботи зосереджений на глибокому навчанні. Для порівняння було прийнято рішення будувати систему одночасно на двох базах – з використанням градієнтного спуску та за допомогою мурашиного алгоритму. Таким чином, можливо провести більш чіткий аналіз та зрозуміти, чи працюють ці алгоритми однаково, чи все ж таки є досить сильні розрізнення.

Для того, щоб побудувати такий алгоритм, було виділено одну нейрону мережу, яка, в свою чергу, абсолютно ідентична у двох методах, що, також, дозволить вивчати поведінку цих способів навчання в одній і тій самій ситуації.

Оскільки прогнозування являє собою метод, що прогнозує значення відносно інших значень, було виділено первинне число необхідного циклу, а саме 24 вхідних значень, з яких 21 відповідають за дні тижня, 22-й за максимальне значення в періоді, 23-й – за мінімальне значення, а 24-й – за середнє арифметичне значення даних.

Таким чином, буде переглянуто принцип роботи та оцінено, наскільки алгоритм гарно себе показує в рамках сезонності.

Таким чином, побудова алгоритму базується в подальшому на використанні класичного багатошарового перцептрону з двома методами глибокого навчання.

Принцип роботи градієнтного спуску базується на отриманні числового значення помилки, яка являє собою різницю між помилкою та результатом. Якщо

у звичайних ситуаціях основним аспектом розуміння навченості алгоритму є його точність відповіді, то у випадку з алгоритмами для прогнозування такої точності, на жаль, отримати фізично неможливо, проте, відповіді мають максимально близько прогнозувати вже відомі дії (необов'язково ідеально точно, проте, максимально точно, як це може бути), а значить, можна прибрати необхідність в навчанні до останнього, достатньо лише певної кількості ітерацій.

Принцип роботи мурашиного алгоритму дуже простий:

1. Створюється покоління мурах.
2. Пошук найменшого шляху до цілі (в нашому випадку – найменшу похибку).
3. Мурахи залишають феромони під час переміщення.
4. Завершення циклу.
5. Створюється нове покоління мурах і продовжується процес відносно пунктів 2-4.

Принцип роботи алгоритму добре представлено у роботі Горносталя О. та Дорогого Я [10].

При цьому, хотілось би зазначити, що феромони, що залишені мурахами, з часом зникають, тобто, чим частіше мурахи рухаються в одному шляху, тим більше феромонів буде залишатись на ньому, а, отже, в подальшому ще більше мурах буде переміщатись по цьому шляху, тобто, похибка з кожним проходом буде геометрично менше.

В нашому випадку ми шукаємо не шляхи, а коефіцієнти важливості для виконання формули прогнозування попиту на товар.

Також, після навчання системи доцільним буде включати мурашиний алгоритм саме в період сильної невідповідності даних з реальністю, для більш чіткого розуміння системою, що саме відбувається з ринком та як кожен фактор впливає на нинішню ситуацію.

На рисунку 3.1 представлено блок-схему алгоритму прогнозування з навчанням через мурашиний алгоритм.

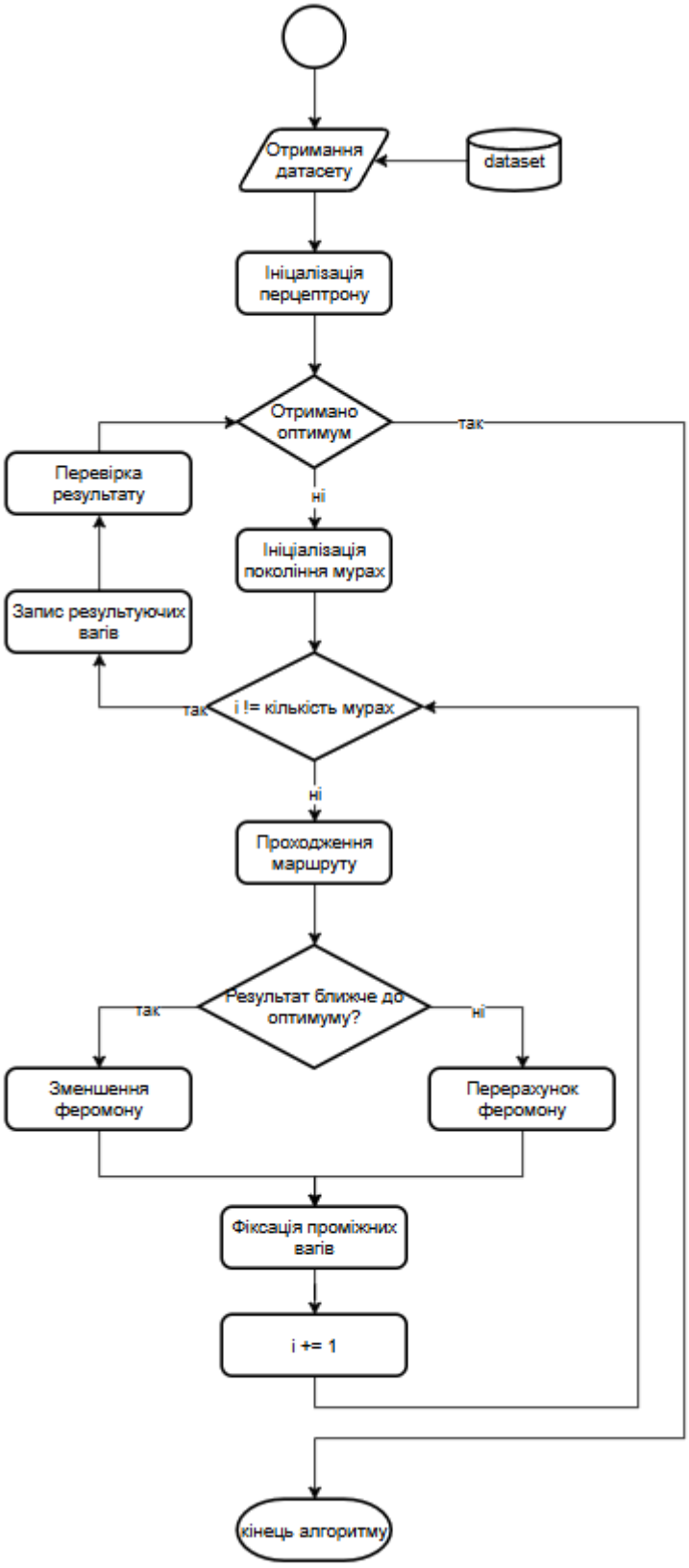


Рис. 3.1 Алгоритм навчання за допомогою мурашиного алгоритму

Для побудови нейромережі було обрано побудова багатошарового перцептрону з 2-ма прихованими шарами. Таким чином, перцептрон матиме наступні параметри:

- вхідний шар на 24 нейрони. Перші 21 нейрон відповідають за період, завдяки якому буде виконуватись прогнозування, 22-й нейрон отримує день з максимальним прибутком, 23-й – день з мінімальним прибутком, а 24-й – середнє арифметичне значення за обраний період;
- перший прихований шар на 16 нейронів;
- другий прихований шар на 8 нейронів;
- вихідний шар на 1 нейрон.

Для навчання було обрано використання 20 мурах та проходження 100 ітерацій навчання. Такої кількості достатньо для виконання процесу навчання.

Така модель має продемонструвати найбільш оптимальну оцінку та прогнозування має забезпечити стабільність та точність.

На рисунку 3.2 представлено графічне зображення перцептрону.

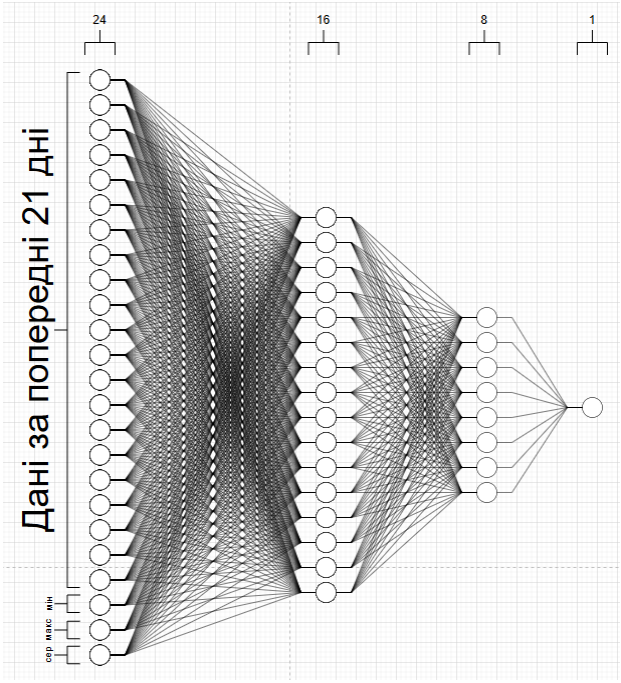


Рис. 3.2 Графічне представлення багатошарового перцептрону

Використання мурашиного алгоритму дозволяє нам прискорити навчання, оскільки одночасно навчається колектив, а не одна мережа. Також, таким чином можливо буде зменшити витрати на доучування при використанні системи в «бойовому» режимі.

Перед виконанням розробки алгоритму необхідно виділити функціональні та нефункціональні вимоги. До функціональних вимог можна віднести:

– алгоритм має видавати прогнози відповідними прогнозованим критеріям величинами;
– алгоритм має приймати навчальні та тестові вибірки;
– алгоритм має використовувати принципи мурашиної оптимізації для налаштування параметрів нейронної мережі;
– алгоритм має оновлювати модель при отриманні нових даних, покращуючи точність прогнозу;
– алгоритм має пропонувати прогнозування в зручному форматі.

До нефункціональних вимог до алгоритму можна віднести наступні вимоги:

– прогнози, що будуть видані алгоритмом, мають досягати не менше ніж 70% середньої точності;
– час навчання та прогнозування має бути оперативним, задля зменшення витрати часу в бізнес-процесі;
– алгоритм має коректно працювати при збільшенні даних;
– алгоритм має бути стійким до шумних та неповних даних;

алгоритм має бути сумісним із сучасними системами.

3.2 Опис використаних програмних засобів

Для розробки додатку «Записна книжка» необхідно обрати досить потужний та оптимізований стек технологій, який дозволить швидко та ефективно записувати нотатки, незважаючи на «залізо» системи, в якій використовується. Для виконання роботи було обрано наступні інструменти:

- середовище розробки Microsoft Visual Studio — середовище розробки, що ідеально спроектоване для розробки мовами сімейства CLR – C++, C#, IronPython, IronRuby і так далі. Visual Studio пропонує користувачу не просто інструмент для написання коду та побудови проєкту, а й доволі потужний інструмент для оформлення документації (за необхідності), побудові графічних рішень, з'єднання та роботи з базами даних, тощо.
- фреймворк .NET 9 – на сьогодні це найсучасніша та стабільна версія фрейворку. Фреймворк .NET має в собі увесь базовий набір для написання систем на базі мов сімейства CLR, а також, як було зазначено раніше, завдяки .NET 5+, код, написаний на мовах CLR є кросплатформеним. Також, завдяки .NET фреймворку (але це не залежить від версії) розробник отримує можливість писати одночасно на декількох мовах програмування, наприклад, розрахункові задачі та процеси – на F#, графічні вікна – на Visual Basic, а алгоритми та основні операції - на C#, отримуючи, таким чином, всі сильні сторони різних мов та створюючи потужний інструмент.
- платформа WinForms – це налаштування над фреймворком для побудови графічних додатків відносно просто. Основним недоліком перед своїми колегами (а саме, перед WPF та MAUI) є обмеженість в візуальних можливостях, проте, для побудови демонстраційного додатку, щоб просто показати, як саме працює система, WinForms дозволить досить легко та на рівні з іншими, тим паче, що основна задача дипломного проєкту полягає саме у побудові алгоритму, а не системи.
- мова програмування C# — Об'єктно-орієнтована мова програмування, розроблена компанією Microsoft. Являє собою нащадка C++ та, можна сказати, «сестрою-близнючкою» Java. Має статичну типізацію та підтримує більшість інструментів та процедур, що були притаманні C та C++, що робить мову доволі швидкою та відмовостійкою, що дуже

необхідно в умовах поточної задачі. Також, окрім переваг своїх батьків, отримала й ряд сучасних переваг, таких як об’єктно-орієнтованих синтаксис, синтаксичний цукор для більш чіткого розуміння написаного коду, автоматичного збирання сміття, тощо. Також, з приходом .NET 5+ мова набула процесу кросплатформеності, що надає змогу писати .NET додаток так, що він зможе запускатись із Windows, Linux, MacOS, iOS та Android.

SQLite — це легка, вбудована база даних, яка вирізняється своєю автономністю та простотою використання. Її головна перевага полягає в тому, що для роботи з нею не потрібно встановлювати окремий сервер або спеціалізоване програмне забезпечення для управління базами даних. Уся система зберігається в одному файлі, що дозволяє інтегрувати SQLite безпосередньо в додаток і запускати його без додаткових налаштувань. Завдяки такій архітектурі SQLite ідеально підходить для невеликих проєктів, мобільних застосунків, локальних утиліт або демонстраційних систем, де немає потреби в складній багатокористувацькій інфраструктурі. Вона забезпечує достатній рівень функціональності для зберігання, запиту та оновлення даних, при цьому залишаючись максимально простою в розгортанні.

3.3 Опис структури проекту

Архітектура проекту побудована на чіткому розділенні відповідальностей між категоріями файлів, що дозволяє підтримувати чисту структуру, спрощує навігацію та забезпечує гнучкість при розширенні функціоналу. Основні категорії включають: контролери, елементи управління, розширення, моки, моделі даних, інтерфейси (вікна), а також внутрішні підгрупи, що деталізують логіку кожного компонента.

Контролери відповідають за координацію процесів між окремими частинами системи. Вони реалізують логіку взаємодії між вікнами, моделями та елементами управління, забезпечуючи стабільну передачу даних і реакцію на дії користувача. Такий підхід дозволяє уникнути жорсткої прив’язки між

компонентами, що критично важливо для підтримки масштабованості: при зміні одного модуля немає потреби перечитувати весь код — достатньо оновити відповідний контролер.

Елементи управління — це кастомні компоненти інтерфейсу, створені для зручного відображення сутностей, таких як товари або категорії. Вони дозволяють повторно використовувати візуальні шаблони, забезпечуючи єдність стилю та поведінки в межах усієї системи.

Розширення — набір утиліт і додаткових методів, що модифікують або доповнюють стандартні можливості службових класів C#. Вони спрощують розробку, дозволяючи уникати дублювання коду та підвищуючи ефективність проектування інтерфейсу.

Моки — це набір тестових або базових значень, які використовуються для початкового наповнення системи. Вони дозволяють швидко перевірити працездатність окремих модулів без необхідності підключення до реальної бази даних або введення вручну.

Моделі даних — одна з ключових категорій, що описує структуру сутностей, їх властивості та зв’язки. Саме завдяки моделям даних система може зберігати інформацію про товари, історію продажів, категорії тощо. Ці моделі є основою для побудови запитів, формування інтерфейсу та реалізації логіки взаємодії з базою даних.

Інтерфейси (вікна) — це візуальні компоненти, які реалізують взаємодію з користувачем. Сюди входять форми для створення товарів і категорій, перегляду історії продажів, роботи з чеками та інші. Кожне вікно має чітко визначену роль і пов’язане з відповідними моделями та контролерами, що забезпечує логічну цілісність системи.

Загалом така структура дозволяє підтримувати проект у чистому, модульному вигляді, де кожен компонент виконує свою роль, а зміни в одному місці не порушують стабільність усієї системи. Це особливо важливо для демонстраційних або дослідницьких проектів, де гнучкість і прозорість архітектури мають вирішальне значення.

Частина графічного представлення структури проекту можна побачити на рисунку 3.6.

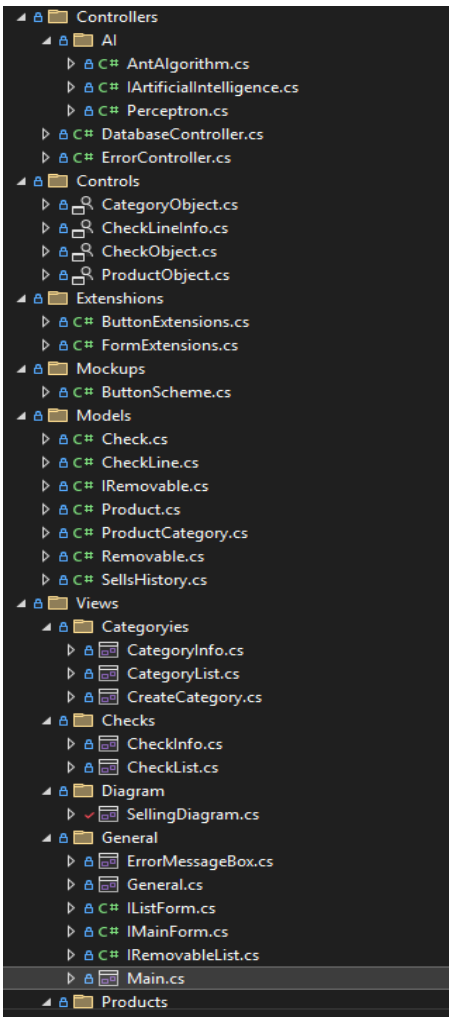


Рис. 3.6 Файлова система

На рисунку 3.7 представлено список класів, що використовуються у системі.

Рис. 3.7 Список класів системи

У структурі системи чітко виділяються кілька класів, кожен з яких відповідає за окрему форму інтерфейсу. Серед них — Main, що виконує роль головного хаба, General як базовий контейнер, ErrorMessageBox для відображення помилок, а також функціональні форми: CreateProduct, CreateCategory, CategoryInfo, CategoryList, ProductList, CheckInfo, CheckList і SellingDiagram. Кожна з них реалізує окрему частину логіки взаємодії з користувачем і даними.

Для забезпечення гнучкості та повторного використання логіки в системі застосовано кілька інтерфейсів:

– IListForm — визначає, чи є форма списковою (тобто такою, що відображає перелік елементів). Його реалізують CheckList, ProductList та CategoryList, що дозволяє уніфікувати роботу з таблицями та списками.

– IMainForm — забезпечує доступ до методів головного вікна (Main) з інших форм. Це дозволяє, наприклад, перемикатися між вікнами або викликати загальні функції навігації.
– IRemovableList — додає можливість реалізовувати поведінку видалення елементів з бази. Його реалізують ті форми, де передбачено роботу з редагуванням або очищенням даних.

Завдяки такій структурі система залишається модульною, легко розширюваною та зручною для тестування. Кожен клас і інтерфейс виконує чітко визначену роль, що відповідає принципам чистої архітектури та дозволяє уникати дублювання логіки.

Схематична архітектура вікон додатку зображена на рисунку 3.8.

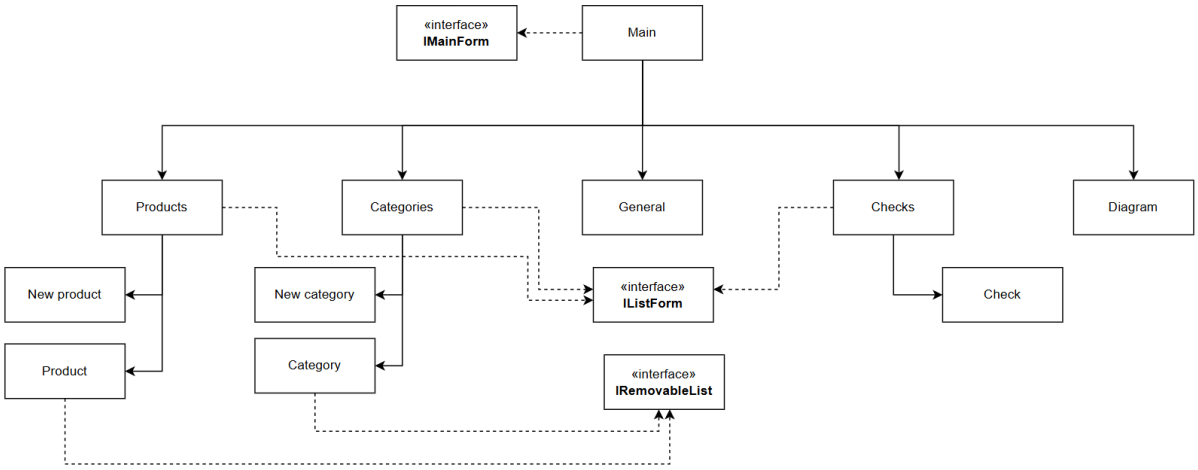


Рис. 3.8 Схема зв’язків вікон додатку

3.4 Опис інтерфейсу

Вікно інформації про категорію призначене для того, щоб користувач міг швидко отримати базові відомості про поточну категорію, яку він переглядає. У цьому вікні відображається не лише назва чи опис категорії, а також повний список продуктів, що до неї належать. Це дозволяє користувачу краще орієнтуватися в асортименті та швидше знаходити потрібні товари.

У майбутньому планується розширити функціонал цього вікна. Зокрема, розглядається можливість переходу до вікна окремого продукту, а також функція

переміщення кількох товарів між категоріями. Такі оновлення мають на меті зробити роботу з системою більш зручною та ефективною.

Основні функції:

– відображення інформації про категорію;
– демонстрація списку товарів, що належать до категорії;
– закриття вікна.

На рисунку 3.9 представлено вікно опису категорії товарів.

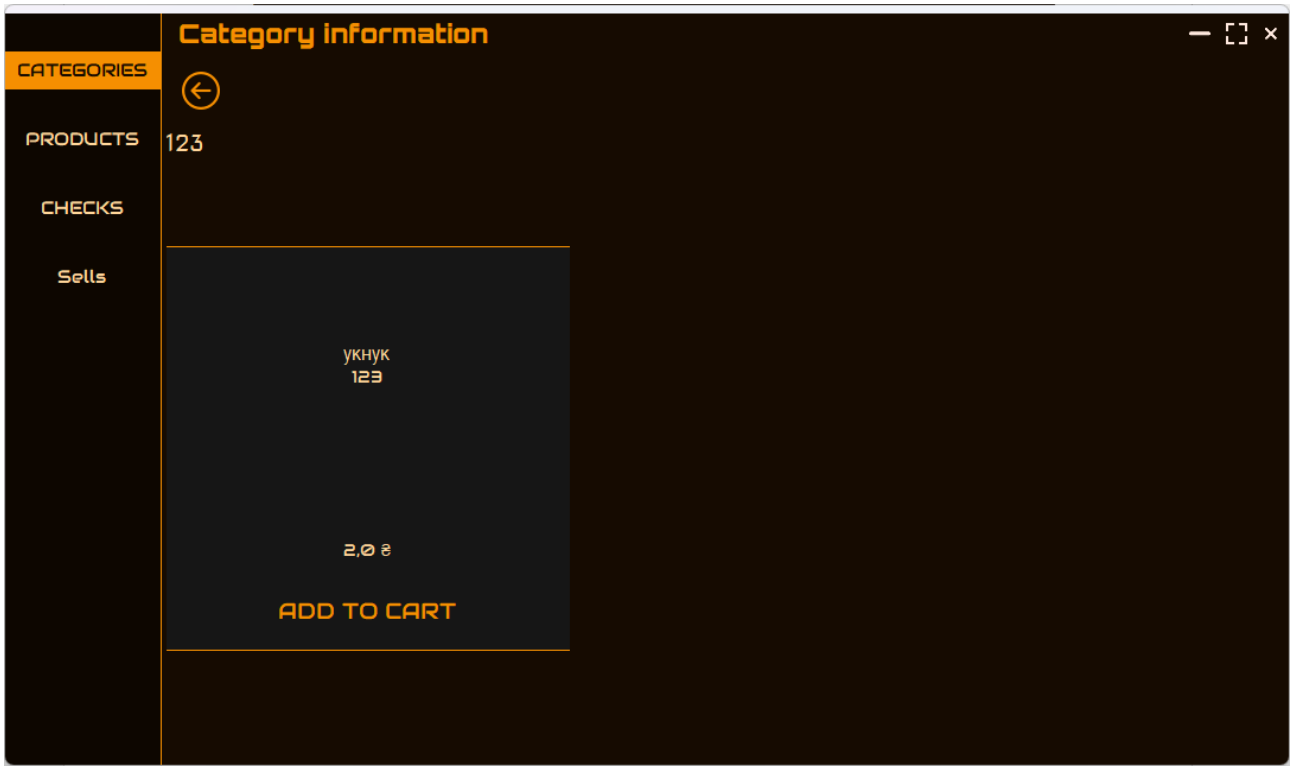


Рис. 3.9 Вікно інформації про категорію

Вікно списку категорій виконує функцію навігаційного елементу, який дозволяє користувачеві швидко ознайомитися з усіма категоріями, що були раніше створені іншими користувачами. Такий підхід забезпечує більш комфортний та інтуїтивно зрозумілий пошук потрібної інформації або товарів.

У поточній демонстраційній версії це вікно має обмежений набір функцій. На даний момент користувач може лише переглянути інформацію про вже існуючу категорію або створити нову. Ці дві дії є основними і достатніми для виконання поставлених задач у рамках цієї моделі.

З огляду на специфіку поточного сценарію, додаткові функції не передбачаються, оскільки вони не є необхідними для досягнення цілей, визначених у цьому етапі розробки.

На рисунку 3.10 представлено вікно списку категорій.

Рис. 3.10 Вікно списку категорій

Вікно створення категорії є інструментом, що дозволяє користувачеві додавати нові категорії товарів до системи. Це можуть бути будь-які типи продукції — від базових продуктів харчування, таких як молоко чи яйця, до аксесуарів і техніки, наприклад, ремінці або смартфони. Така гнучкість у створенні категорій відкриває можливість для проведення більш реалістичних і наближених до реальних умов тестувань, особливо у випадках, коли використовується метод прогнозування.

Завдяки цьому вікну можна моделювати різні сценарії, що відповідають справжнім умовам роботи системи, що, своєю чергою, сприяє точнішій оцінці її функціональності та стабільності. Основною функцією вікна є створення категорії товару.

Інші функції:

- зв’язування даних з моделлю (для покращеного та спрощенного редагування);
- додавання даних до БД;
- закриття вікна.

Функціональність кожного з вікон не передбачає в собі якихось дуже потужних функцій, проте, кожне з вікон спроектовано таким чином, щоб виконувати лише свої певні задачі.

На рисунку 3.11 представлено вікно створення категорії.

Рис. 3.11 Вікно створення категорій

Вікно інформації про чек призначене для відображення детальних даних щодо конкретного чека, який був сформований у системі. Це вікно дозволяє користувачу переглянути основні параметри покупки: дату створення, загальну суму, перелік товарів, що були додані до чека, а також відповідні категорії кожного з них.

Такий формат подачі інформації забезпечує зручність у аналізі проведених операцій, особливо якщо потрібно перевірити склад покупки або провести

тестування функціоналу прогнозування. Вікно не перевантажене зайвими елементами — воно виконує свою задачу чітко та лаконічно, що особливо важливо на етапі демонстраційної моделі.

У майбутньому можливе розширення функціоналу: додавання можливості редагування чека, експорту даних або інтеграції з іншими модулями системи. Але наразі, в рамках поточного завдання, реалізованих функцій достатньо для ефективного тестування та оцінки роботи системи.

Вікно представлено на рисунку 3.12.

Рис. 3.12 Вікно інформації про чек

Вікно списку чеків слугує для зручного перегляду всіх чеків, які були сформовані в системі. Кожен чек у цьому списку представлений у стислому форматі — з базовою інформацією, такою як дата створення, загальна сума покупки та кількість товарів. Це дозволяє швидко оцінити історію операцій без потреби відкривати кожен чек окремо.

Функціонал цього вікна наразі обмежений лише переглядом існуючих чеків. Користувач може вибрати будь-який з них для детального ознайомлення,

але додаткові дії, такі як редагування або видалення, не передбачені в поточній моделі.

Такий підхід є достатнім для демонстраційного етапу, оскільки основна задача — надати можливість тестувати базову логіку роботи з чеками та перевірити коректність відображення даних. У майбутньому можливе розширення функцій, якщо це буде необхідно для більш складних сценаріїв використання.

Вікно можна побачити на рисунку 3.13.

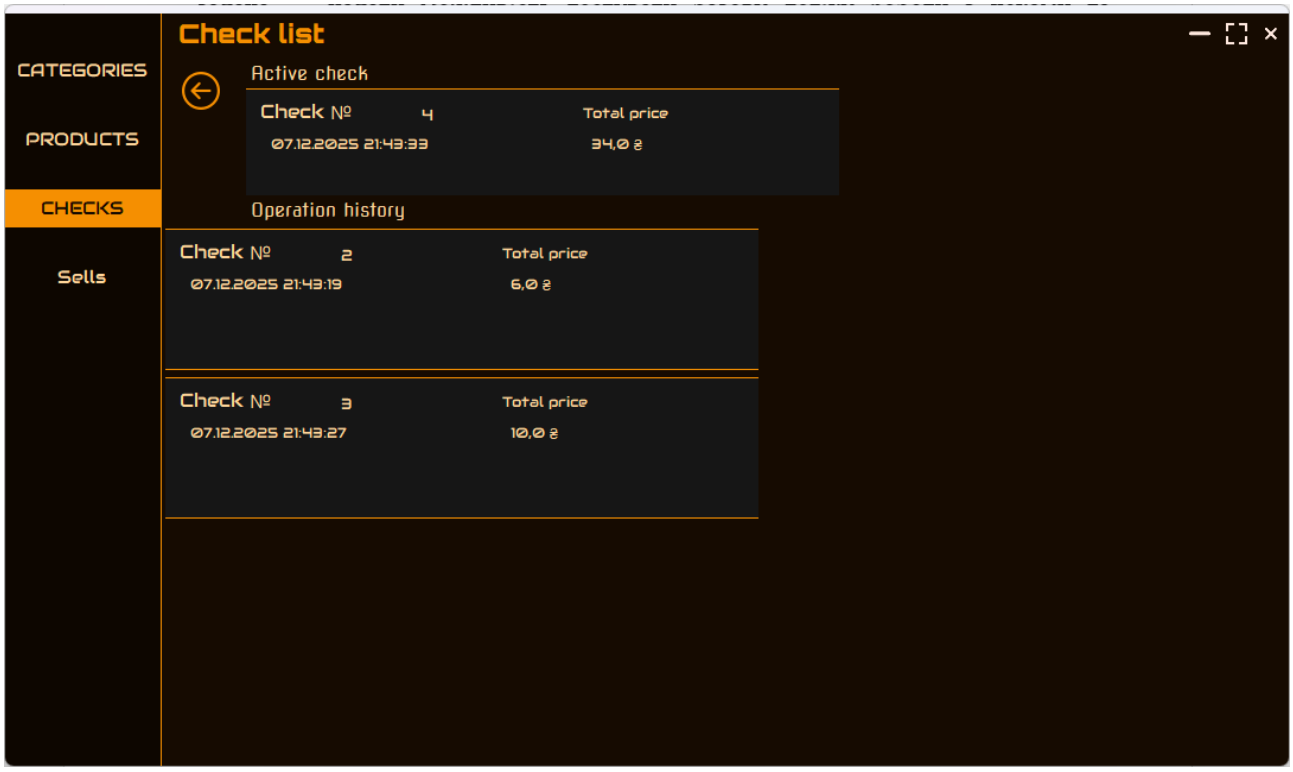


Рис. 3.13 Вікно списку чеків

Вікно для відображення помилки слугує для інформування користувача про некоректну дію або збій у роботі системи. Воно показує коротке повідомлення з поясненням ситуації, що виникла, і, за потреби, підказку щодо подальших дій. Простота та лаконічність цього вікна дозволяють швидко зорієнтуватися та повернутися до роботи без зайвих труднощів.

Вікно представлено на рисунку 3.14.

Рис. 3.14 Вікно помилки про помилку

Базове вікно є простим елементом інтерфейсу, що містить лише зображення. Воно не виконує жодної функціональної ролі та не взаємодіє з даними чи користувачем. Його призначення — суто візуальне, використовується як частина загального оформлення або для демонстраційних цілей у поточній моделі. Це вікно відкривається одразу під час запуску програми.

Вікно представлено на рисунку 3.15.

Рис. 3.15 Базове вікно

Головне вікно системи виконує роль центрального хаба, з якого користувач розпочинає взаємодію з усіма основними модулями. Його структура побудована

таким чином, щоб забезпечити швидкий доступ до різних функціональних вікон — через зручне ліве меню, яке постійно доступне незалежно від того, де саме перебуває користувач у межах інтерфейсу.

У цьому меню представлені ключові розділи: перегляд категорій, створення нової категорії, список чеків, інформація про окремий чек, а також додаткові вікна, що відповідають за тестування, створення даних або перегляд помилок. Кожен пункт меню веде до відповідного вікна, дозволяючи перемикатися між ними без втрати контексту.

Головне вікно не містить складної логіки чи обробки даних — його основна задача полягає в тому, щоб бути точкою входу та навігаційним центром. Такий підхід дозволяє зберігати чисту архітектуру інтерфейсу, де кожен модуль відповідає лише за свою частину функціоналу, а головне вікно — за зручність переходів між ними.

Вікно представлено на рисунку 3.16.

Рис. 3.16 Головне вікно

Вікно створення товару дозволяє користувачеві додати новий продукт до системи, вказавши його назву, категорію, ціну та інші базові характеристики. Це необхідно для формування реалістичних чеків і подальшого тестування

функціоналу, зокрема методів прогнозування. Інтерфейс побудований просто: користувач швидко вводить дані, система зберігає товар — і він одразу доступний для роботи в інших модулях. На цьому етапі додаткові функції не передбачені, оскільки основна задача — забезпечити базову можливість створення товарів для тестових сценаріїв.

В перспективі буде додано можливість масового імпорту товарів через файл, що дозволить швидше наповнювати систему даними для складніших сценаріїв тестування.

Рисунок 3.17 містить демонстрацію вікна нових товарів.

Рис. 3.17 Створення нового товару

Вікно списку товарів призначене для перегляду всіх продуктів, що були додані до системи. Кожен товар представлений у вигляді окремого рядка з базовими характеристиками — назвою, категорією, ціною та, за потреби, додатковими параметрами. Це дозволяє швидко оцінити наявні дані, перевірити коректність введеної інформації та сформувати чек або протестувати інші модулі.

Інтерфейс побудований так, щоб користувач міг легко знайти потрібний товар, навіть якщо їх у системі вже багато. У майбутньому можливе додавання фільтрів, сортування або пошуку, але наразі реалізовано лише базову функцію перегляду, що повністю відповідає задачам демонстраційної моделі.

На рисунку 3.18 представлено графічне вікно списку товарів.

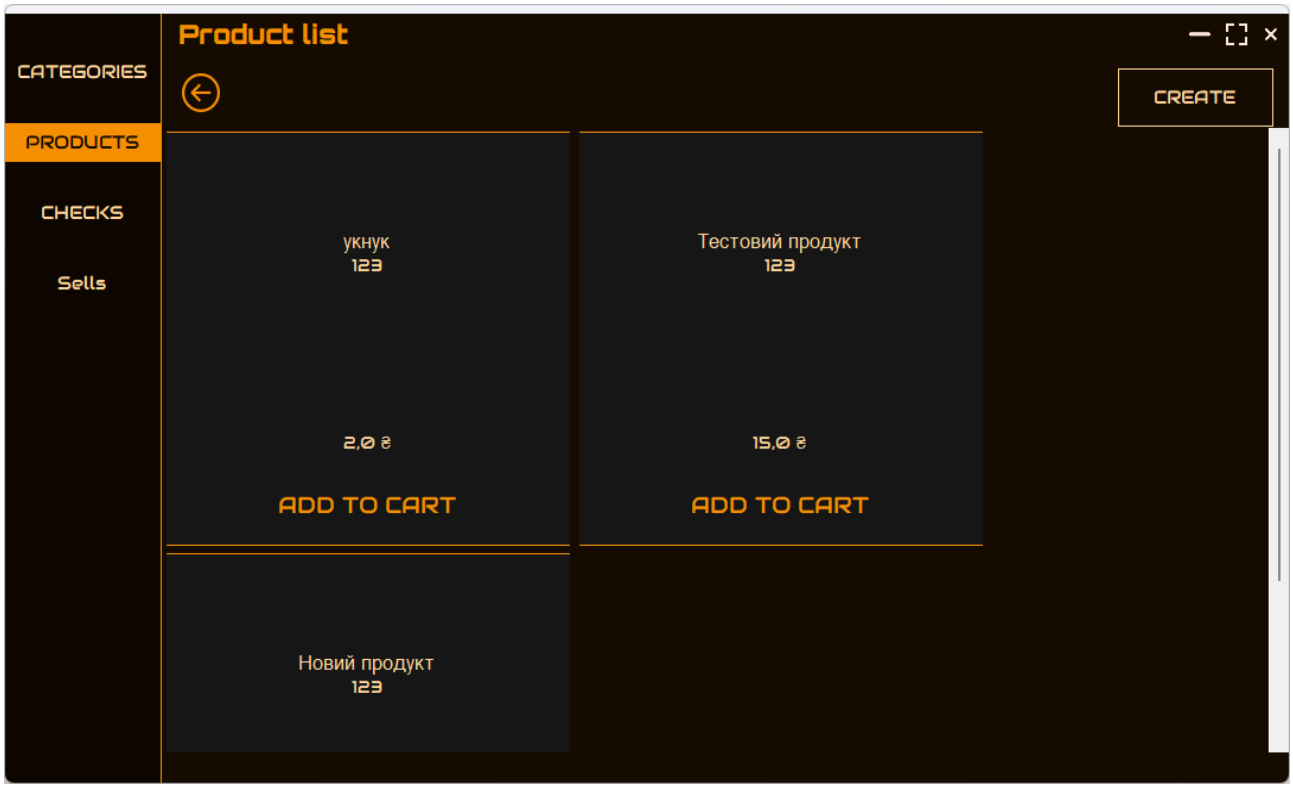


Рис. 3.18 Список товарів

Вікно прогнозування попиту є центральним елементом дослідницької частини системи. Його основна мета — надати користувачеві можливість оцінити очікувану динаміку продажів конкретного товару на основі історичних даних, зібраних із чеків. Це вікно поєднує аналітичну функцію з візуалізацією, дозволяючи побачити результат роботи алгоритму прогнозування у вигляді графіка.

Інтерфейс побудований таким чином, щоб процес був максимально прозорим: користувач обирає товар, система автоматично аналізує попередні покупки, і на основі цього будується графік, де по осі часу відкладається період,

а по вертикалі — прогнозована кількість продажів. Така візуальна форма дозволяє швидко виявити тренди, сезонні коливання або потенційні аномалії.

Це вікно є ключовим для дослідження, оскільки саме тут перевіряється ефективність обраного методу прогнозування. Воно дозволяє не лише побачити результат, а й оцінити його адекватність у контексті реальних даних.

У поточній версії параметри моделі не змінюються вручну — все працює за фіксованим сценарієм, що забезпечує стабільність і повторюваність результатів для порівняння. На рисунку 3.19 представлено UML-діаграму потоку даних.

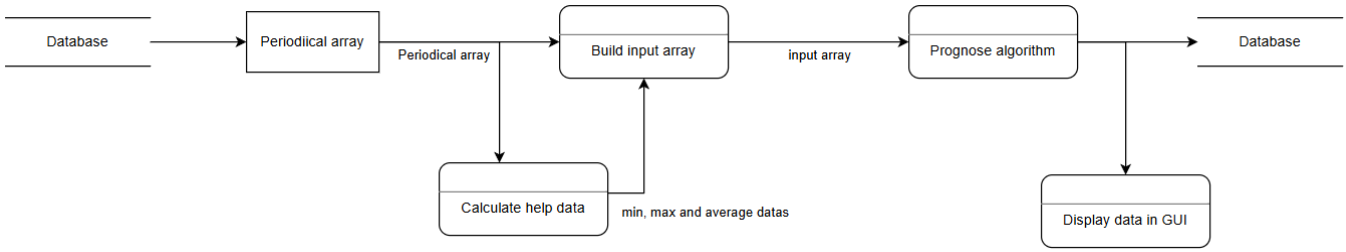


Рис. 3.19 Діаграма потоку даних

У майбутньому планується розширення функціоналу: додавання можливості вибору моделі прогнозування, налаштування параметрів, а також порівняння кількох варіантів графіків. Але вже зараз це вікно виконує свою дослідницьку функцію повністю — воно дозволяє побачити, як система реагує на реальні дані, і чи здатна вона передбачити майбутній попит.

На рисунку 3.20 представлено вікно з результатами прогнозування.

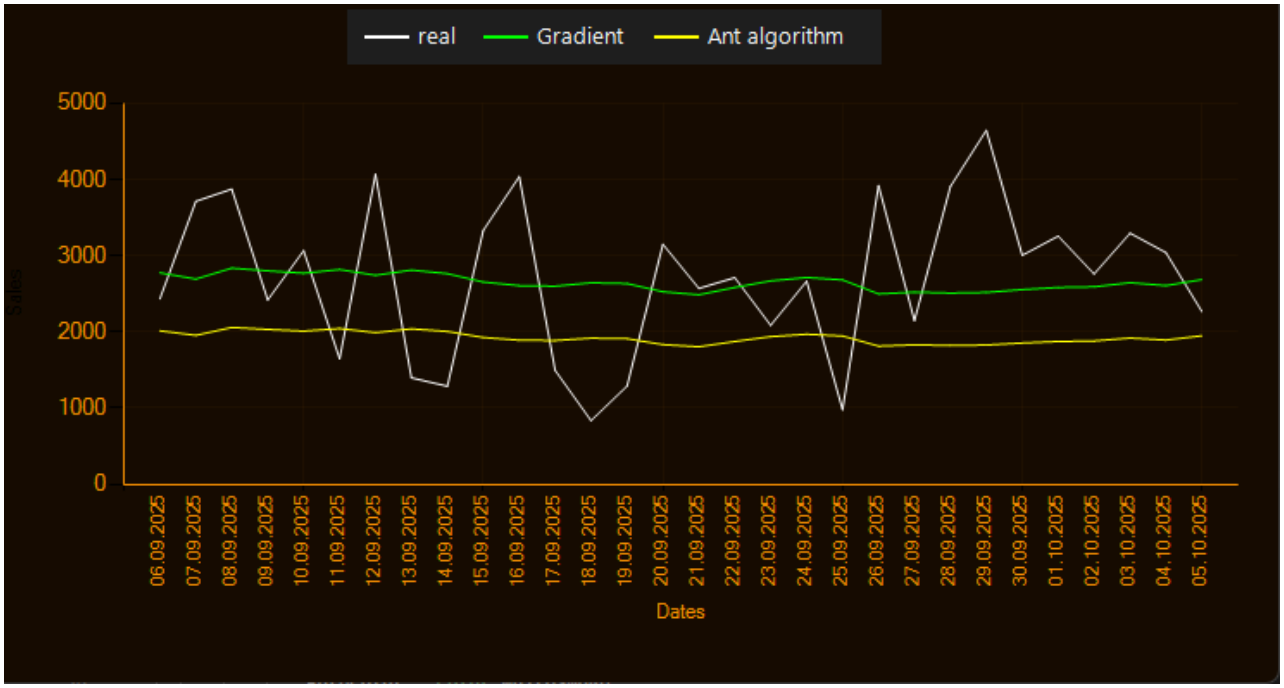


Рис. 3.20 Вікно прогнозування

3.5 Опис розроблених класів

В рамках даного проєкту виділено основні 3 класи: інтерфейс IArtificialIntelligence, AntAlgorithm та Perceptron.

Інтерфейс IArtificialIntelligence визначає базовий контракт для реалізації штучного інтелекту в системі. Він задає структуру, яку мають реалізовувати всі класи, що відповідають за прогнозування або навчання моделей.

Основні методи:

- double Calculate(double[] inputs)

Метод для обчислення результату на основі вхідних даних. Приймає масив чисел як вхід, повертає одне числове значення — прогноз або оцінку.

Призначення: використовується для генерації результату після навчання моделі.

- void Train(double[][] inputs, double[] outputs)

Метод для навчання моделі. Приймає набір вхідних даних (двовимірний масив) та відповідні очікувані результати.

Призначення: дозволяє моделі адаптувати свої параметри (ваги, структуру) для точнішого прогнозування.

- інтерфейс дозволяє легко змінювати або розширювати логіку ШІ, не змінюючи загальну структуру програми.

- підтримує реалізацію різних підходів — від простих алгоритмів до нейронних мереж або генетичних моделей.

AntAlgorithm реалізує мурашиний алгоритм для навчання моделі прогнозування. Виступає як модуль штучного інтелекту, що оптимізує ваги на основі імітації поведінки мурах.

Методи:

- Calculate(double[] inputs)

Обчислює прогнозоване значення на основі вхідних даних та збережених ваг.

- Train(double[][] inputs, double[] outputs)

Запускає процес навчання моделі: нормалізує дані та передає їх у навчальний цикл.

- TrainBatch(double[][] inputs, double[] outputs)

Основний цикл мурашиного алгоритму: генерує ваги, обчислює помилки, оновлює феромони.

- CalculateError(double[] weights, double[][] inputs, double[] outputs)

Обчислює середньоквадратичну помилку між прогнозом і очікуваними значеннями з регуляризацією.

- SaveParameters(string filePath)

Зберігає найкращі знайдені ваги та феромони у JSON-файл.

- LoadParameters(string filePath)

Завантажує збережені параметри моделі з файлу, якщо він існує.

- ToJaggedArray(double[,])

Перетворює двовимірну матрицю у зубчастий масив для серіалізації.

- ToMatrix(double[][])

Перетворює зубчастий масив назад у двовимірну матрицю.

- Dispose()

Автоматично викликається при завершенні роботи об’єкта — зберігає

параметри.

Призначення:

Percepton реалізує просту нейронну мережу типу персептрону з одним прихованим шаром. Використовується для прогнозування на основі навчання з прикладами. Підходить для демонстраційних задач, де важлива зрозуміла структура та стабільна поведінка.

Методи:

- Calculate(double[] inputs)

Обчислює прогнозоване значення на основі вхідного масиву. Виконує пряме проходження через мережу з нормалізацією та активацією.

- Train(double[][] inputs, double[] outputs)

Навчає модель на основі заданих входів і очікуваних результатів.

- Dispose()

Зберігає ваги моделі у файл при завершенні роботи. Дозволяє зберігати стан між сесіями.

- Sigmoid(double x)

Активаційна функція, що використовується для обмеження значень у межах [0,1].

- ToJaggedArray(double[,])

Перетворює двовимірну матрицю у зубчастий масив для серіалізації.

- ToMatrix(double[][])

Перетворює зубчастий масив назад у двовимірну матрицю.

3.6 Опис отриманих результатів

Під час побудови алгоритмів було створено 2 алгоритми: на базі градієнтного спуску та на базі мурашиного алгоритму.

У результаті виконання роботи було отримано список тестових даних, який відображає результат виконання роботи. Результат представлено на рисунку 3.21.

Id	Date	Summary	AntPrediction	PerceptronPrediction
56584	2023-06-27 00:00:00	4590	2140.0	2950.0
56585	2023-06-28 00:00:00	3655	2240.0	3078.0
56586	2023-06-29 00:00:00	1566	2265.0	3121.0
56587	2023-06-30 00:00:00	2660	2155.0	2970.0
56588	2023-07-01 00:00:00	4267	2092.0	2884.0
56589	2023-07-02 00:00:00	4345	2098.0	2891.0
56590	2023-07-03 00:00:00	4952	2082.0	2869.0
56591	2023-07-04 00:00:00	1951	2140.0	2949.0
56592	2023-07-05 00:00:00	4061	2044.0	2816.0
56593	2023-07-06 00:00:00	4287	2085.0	2873.0
56594	2023-07-07 00:00:00	4026	2119.0	2921.0
56595	2023-07-08 00:00:00	4492	2139.0	2948.0
56596	2023-07-09 00:00:00	4873	2238.0	3084.0
56597	2023-07-10 00:00:00	2432	2290.0	3156.0
56598	2023-07-11 00:00:00	4904	2338.0	3222.0
56599	2023-07-12 00:00:00	3788	2393.0	3298.0
56600	2023-07-13 00:00:00	4326	2501.0	3447.0
56601	2023-07-14 00:00:00	3927	2519.0	3471.0
56602	2023-07-15 00:00:00	4380	2499.0	3445.0
56603	2023-07-16 00:00:00	728	2609.0	3596.0
56604	2023-07-17 00:00:00	4874	2603.0	3588.0
56605	2023-07-18 00:00:00	1126	2732.0	3766.0
56606	2023-07-19 00:00:00	3222	2613.0	3601.0
56607	2023-07-20 00:00:00	2497	2598.0	3580.0
56608	2023-07-21 00:00:00	1458	2630.0	3625.0
56609	2023-07-22 00:00:00	4944	2588.0	3567.0
56610	2023-07-23 00:00:00	2954	2648.0	3600.0
56611	2023-07-24 00:00:00	3923	2564.0	3533.0
56612	2023-07-25 00:00:00	4953	2564.0	3484.0
56613	2023-07-26 00:00:00	1062	2632.0	3627.0
56614	2023-07-27 00:00:00	3828	2528.0	3485.0
56615	2023-07-28 00:00:00	614	2512.0	3463.0
56616	2023-07-29 00:00:00	2029	2394.0	1290.0
56617	2023-07-30 00:00:00	2141	2309.0	3183.0

Рис. 3.21 Представлення результатів даних

Як можна побачити з результатів, прогнозування за допомогою штучного інтелекту на базі мурашиного алгоритму відображає більш реальні дані та більш гнучкі результати. Таким чином, результат не є сильно завищеним, а його стабільні зміни відображають, наскільки точнішим є алгоритм.

Під час навчання двох мереж було прийнято рішення виконати задачі

паралельно, щоб зрозуміти, який саме алгоритм буду швидше виконувати свою задачу. Результат порівняння задач представлено у таблиці 3.1.

Таблиця 3.1

Порівняльний аналіз реалізованих алгоритмів

	Мурашиний алгоритм	Градієнтний спуск
Час навчання, хвилин	0:37	1:16
Навантаження процесору, %	11%	14%
Навантаження ОЗП, МБ	30	67
Кількість ітерацій	100	500
Середня абсолютна похибка	1327.12	1211.53

В результаті оцінки можна зрозуміти, що мурашиний алгоритм набагато легше навчається та більш чутливий до змін, завдяки чому більше відображає зміну графіку, хоча точність здається трохи меншою. Завдяки цьому, алгоритм буде набагато краще демонструвати необхідні дані для закупівлі у довгій перспективі.

ВИСНОВКИ

В рамках магістерської кваліфікаційної роботи було проаналізовано тему, оцінено всі ризики розроблюваного алгоритму, його сильні та слабкі сторони, переглянуто та оцінено існуючі алгоритми.

1. Досліджено проблему прогнозування попиту, проаналізовано існуючі підходи та виявлено їх ключові особливості.
2. Запропоновано алгоритм на базі мурашиного алгоритму для прогнозування попиту на товар.
3. Розроблено та впроваджено алгоритм навчання нейронної мережі на базі мурашиного алгоритму для оптимізації вагів нейронної мережі.
4. Проведено тестування алгоритму та проведено експериментальне порівняння з алгоритмом прогнозування на базі глибокого навчання за такими метриками, як час навчання, навантаження процесору, навантаження ОЗП, кількість ітерацій навчання, середня абсолютна похибка.

- час навчання зменшився до на 39 секунд (≈48,7%);
- навантаження процесору знизилось на 3% (≈22,4%);
- навантаження ОЗП знизилось на 37 МБ (≈55.3%);
- середня абсолютна похибка збільшилась на 115,59 (≈9,54%).

5. Алгоритм може бути використано для інтеграції прогнозування попиту у малих магазинах роздрібної торгівлі та як додатковий алгоритм прогнозування для великих компаній.

Робота пройшла апробацію на конференціях та опублікована в:

1. Коденцев М.І., Замрій І.В. Використання мурашиного алгоритму, як ефективної альтернативи вирішення задач оптимізації прогнозування попиту на товар харчової промисловості. Всеукраїнська науково-технічна конференція «Сучасний стан та перспективи розвитку IoT». Збірник тез. – К.: ДУІКТ, 2025. С.149-151.

2. Коденцев М.І., Замрій І.В. Перспективи мурашиного алгоритму, як ефективної альтернативи вирішення задач оптимізації прогнозування попиту на товар харчової промисловості. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». Збірник тез. – К.: ДУІКТ, 2025. С.548-553.

ПЕРЕЛІК ПОСИЛАНЬ

1. Прогнозування попиту. URL: https://uk.wikipedia.org/wiki/Прогнозування_попиту

2. Прогноз попиту як інструмент прибуткового бізнесу. URL: https://sdh.smart-it.com/news-post/prognoz-popytu-yak-instrument-prybutkovogo-biznesu/

3. Порадюк Т. Інтелектуальні системи для прогнозу споживання альтернативної енергії. IX Всеукраїнська студентська науково-технічна конференція "ПРИРОДНИЧІ ТА ГУМАНІТАРНІ НАУКИ. АКТУАЛЬНІ ПИТАННЯ", 2016.

4. Яновський Д., Граф М. Аналіз існуючих методів прогнозування попиту та способів оцінки їх якості. Information Technology: Computer Science, Software Engineering and Cyber Security, 3, 2018.

5. Фінажин М. Інтелектуальна система прогнозування на основі індексів криптовалют, 2022.

6. Задача оптимізації. URL: https://uk.wikipedia.org/wiki/Задача_оптимізації

7. Генетичний алгоритм алгоритм. URL: https://uk.wikipedia.org/wiki/Генетичний_алгоритм

8. Мурашиний алгоритм. URL: https://uk.wikipedia.org/wiki/Мурашиний_алгоритм

9. Тимчук О., Проценко Я., Парамонов А. Застосування алгоритму мурашиної колонії до вирішення задачі декількох комівояжерів без депо, 2019.

10. Горносталь. О., Дорогий Я. Мурашиний алгоритм кластеризації, 2018.

11. Мальченко П. Інтелектуальна система прогнозування вартості комерційних компаній на основі методів машинного навчання, 2023.

12. Бідюк П., Савченко С., Савченко А. Методи інтелектуального аналізу даних в прогнозуванні конкурентоспроможності підприємств. Теоретичні та емпіричні дослідження, 2018.

13. Сізова Н. Інтелектуальні управляючі системи і технології, 2024.

14. Добра Н., Корнілова Є., Самохіна Ж. Інтелектуальні інформаційні технології та системи, 2016.

15. Заруба В. Методи прогнозування рівня попиту, 2009.

16. Овсієнко О. Прогнозування попиту та пропозиції послуг підприємства, 2016.

17. .NET. URL: https://uk.wikipedia.org/wiki/.NET

18. Кормен Т. Алгоритми. Побудова та аналіз, 2020.

19. Булгакова О., Зосімов В., Поздєєв В. Методи та системи штучного інтелекту: теорія та практика, 2025.

20. Величко О., Гордієнко Т. Основи системного аналізу і прийняття оптимальних рішень, 2022.

21. Приймак В. Математичні методи економічного аналізу. Навчальний посібник, 2019.

22. Троцько В. Теорія алгоритмів, 2023.

23. Штовба С., Рудий О. Мурашині алгоритми оптимізації, 2010.

24. Аналітика. URL: https://uk.wikipedia.org/wiki/Аналітика

25. Прогнозування. URL: https://uk.wikipedia.org/wiki/Прогнозування

26. Що таке CRM. URL: https://www.creatio.com/ua/crm/what-is-crm

27. SmartCRM. URL: https://smartcrm.binotel.ua

28. Odoo. URL: https://www.odoo.com/uk_UA

29. KeyCRM. URL: https://ua.keycrm.app/

30. 5 найкращих програм для планування попиту на основі ШІ. URL: https://gmdhsoftware.com/ua/demand-planning-software/

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Магістерська робота

«Адаптивний алгоритм прогнозування попиту на товари харчової промисловості для оптимізації запасів у торговельній мережі»

Виконав: студент групи ПДМ-63 Микита КОДЕНЦЕВ

Керівник: д-р техн.наук, проф., завідувач кафедри ІПЗ Ірина ЗАМРІЙ

Київ - 2025

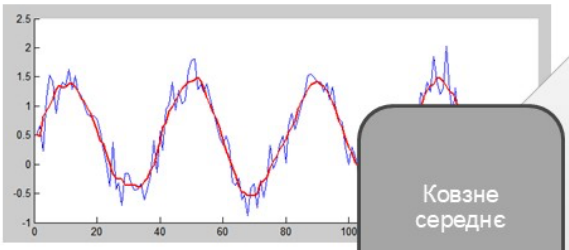
МЕТА, ОБ'ЄКТА ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

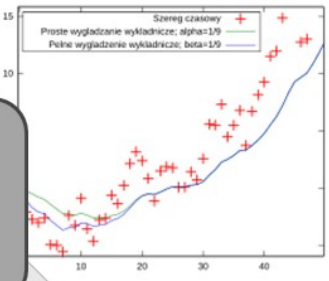
Мета роботи: підвищення ефективності управління запасами торговельної мережі шляхом розробки адаптивного алгоритму прогнозування попиту на основі нейронної мережі та мурашиного алгоритму.

Об'єкт дослідження: процес прогнозування попиту та управління запасами товарів харчової промисловості у торговельній мережі.

Предмет дослідження: адаптивний алгоритм прогнозування попиту з використанням нейронної мережі та мурашиного алгоритму.

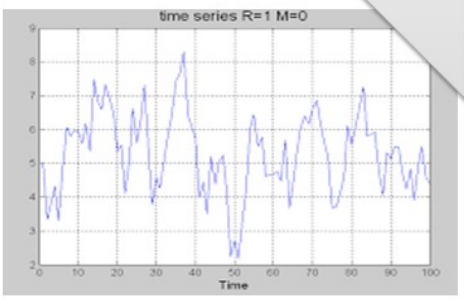
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧІ МЕТОДИ





Ковзне середнє

Експонційне згладження

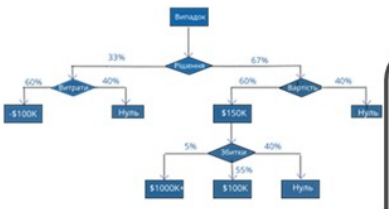


АРКС

Класичні методи (Simple)

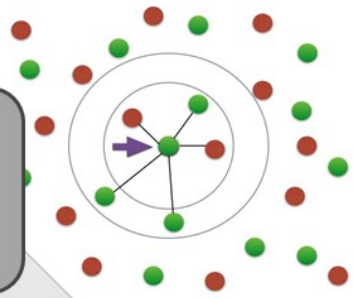
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧІ МЕТОДИ

Методи ШІ (Complex)

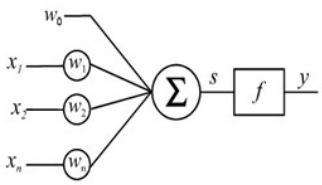


Дерево рішень

KNN



Нейронні мережі



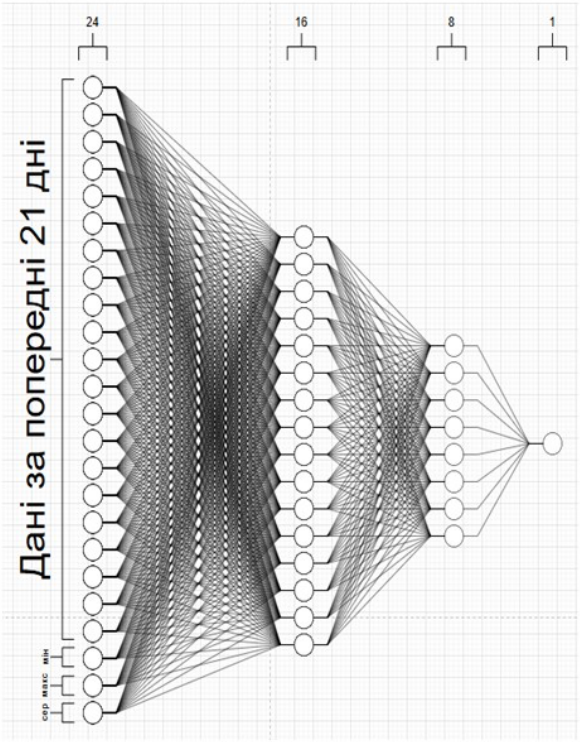
КОМПАРАТИВНИЙ АНАЛІЗ МЕТОДІВ

Для оцінки обрано 5 основних параметрів з оцінкою за п'ятибальною шкалою, а саме

Назва Хар-ка	Ковзне середнє	Експоненційне згладжування	АРКС	Дерево рішень	Метод KNN	Нейронні мережі
Точність (1 – найменш, 5 – найбільш точні)	3	2	1	3	3	5
Складність реалізації (1 – дуже складно, 5 – легко)	5	3	2	3	4	3
Вимоги до даних (1 – дуже залежать, 5 – майже не залежать)	2	2	3	2	2	2
Здатність до узагальнення (1 – майже не узагальнює, 5 – гарна можливість узагальнення)	2	4	4	3	3	4
Стійкість до шуму (1 – висока чутливість, 5 - практично повна стійкість)	3	5	3	2	3	3

5

МАТЕМАТИЧНА МОДЕЛЬ СИСТЕМИ



$$x \in \mathbb{R}^{24}$$

R^{24} – вхідний вектор із 24 даних

$$h_1 = \sigma(W_1 x + b_1)$$

$$h_2 = \sigma(W_2 h_1 + b_2)$$

$$h_3 = \sigma(W_3 h_2 + b_3)$$

$$h_4 = \sigma(W_4 h_3 + b_4)$$

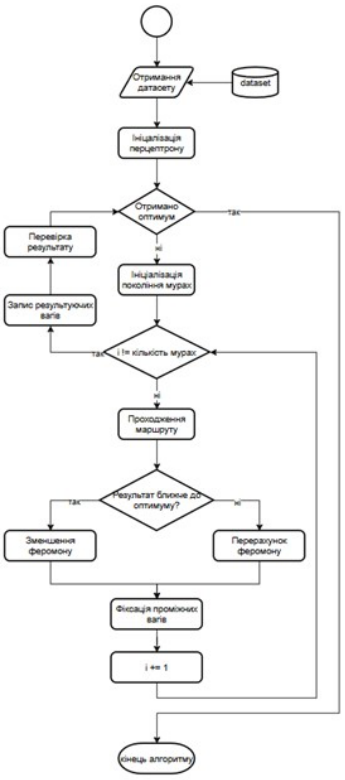
σ – функція активації,
x – вхідні дані,
Wi – ваги, що проходять навчан
bi – параметри зміщення.

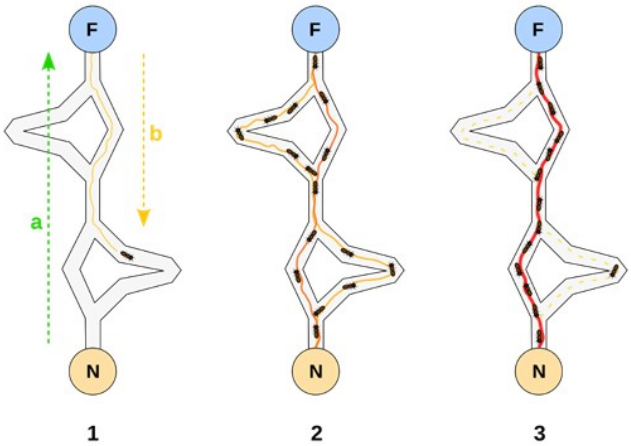
$$\mathrm{ReLU}(x) = \begin{cases} 0, x \le 0 \\ x, x > 0 \end{cases}$$

x – результат розрахунку нейрону.

6

ПРИНЦИП РОБОТИ МУРАШИНОГО АЛГОРИТМУ (ACA)

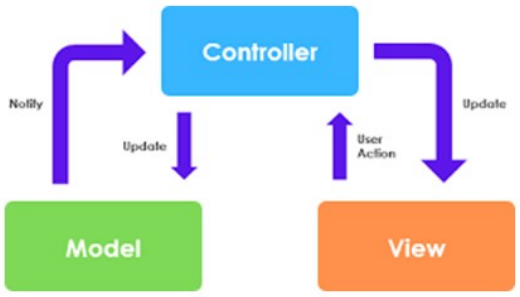


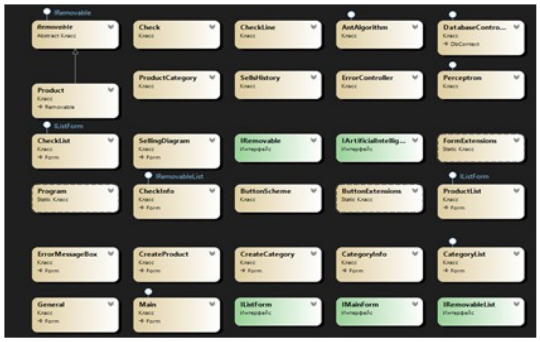


$$P_i = \frac{l_i^q * f_i^p}{\sum_{k=0}^{N} l_k^q * f_k^p}$$

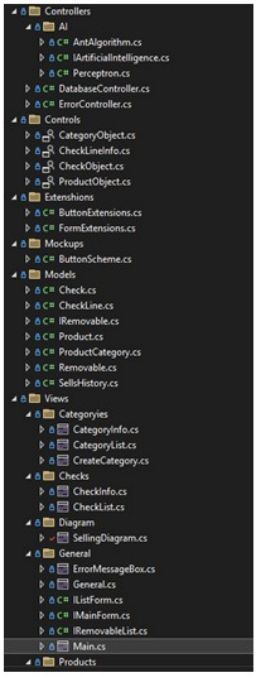
де P_i-ймовірність переходу до шляху i,
l_i-величина, зворотня вагу шляху i,
f_i-кількість феромону на шляху i,
q-величина жадності алгоритму,
p-величина стайності алгоритму.

АРХІТЕКТУРА ТА ІНСТРУМЕНТАРІЙ



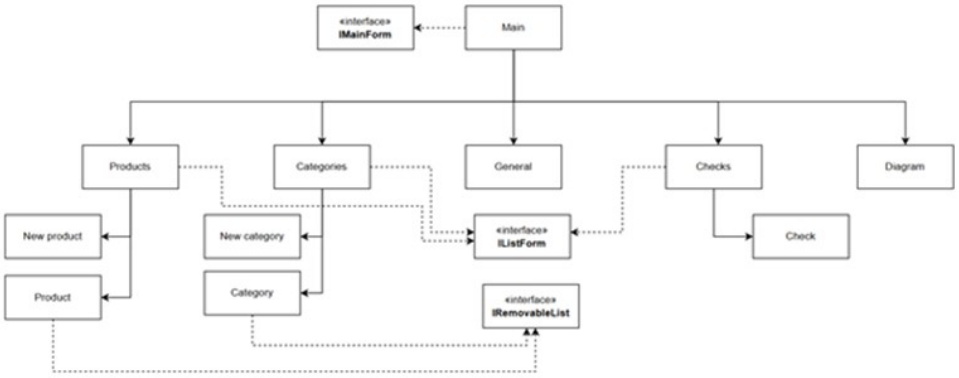


Діаграма класів проєкту

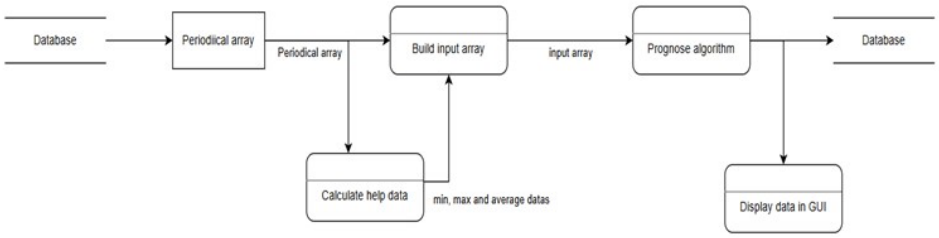


Структура проєкту

ДІАГРАМИ РОЗРОБКИ СИСТЕМИ

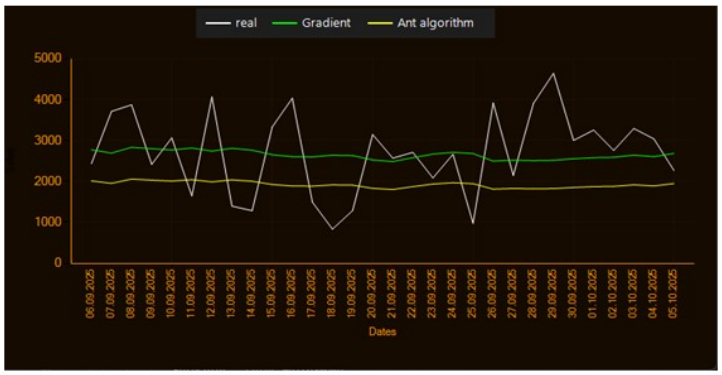


Діаграму зв'язків вікон додатків



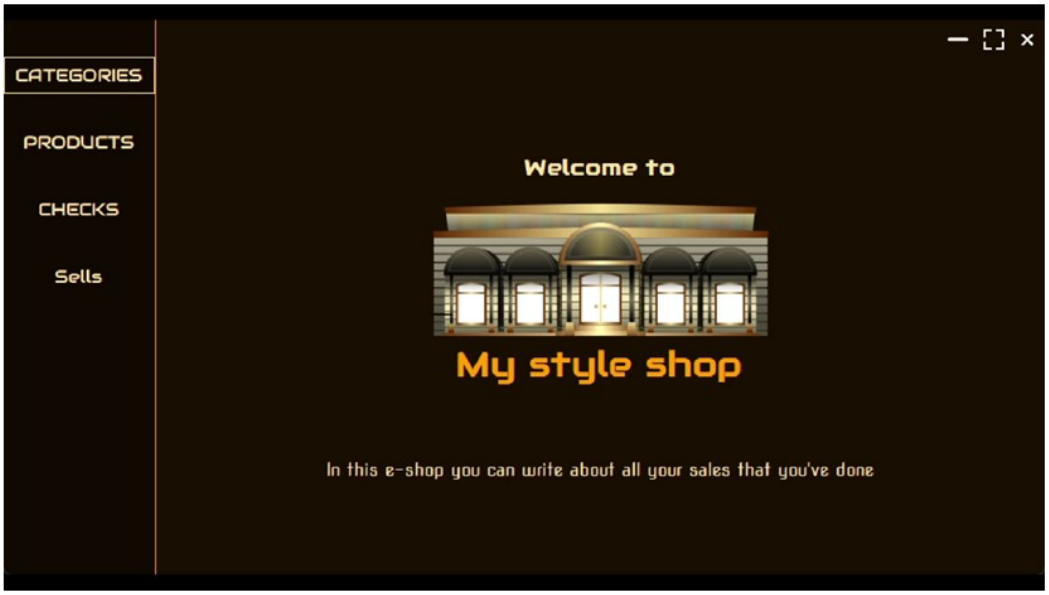
Діаграма потоку даних

ПРАКТИЧНИЙ РЕЗУЛЬТАТ



Id	Date	Summary	AntPrediction	PerceptronPrediction
56584	2023-06-27 00:00:00	4590	2140.0	2950.0
56585	2023-06-28 00:00:00	3655	2240.0	3078.0
56586	2023-06-29 00:00:00	1566	2265.0	3121.0
56587	2023-06-30 00:00:00	2660	2155.0	2970.0
56588	2023-07-01 00:00:00	4267	2092.0	2884.0
56589	2023-07-02 00:00:00	4345	2098.0	2891.0
56590	2023-07-03 00:00:00	4952	2082.0	2869.0
56591	2023-07-04 00:00:00	1951	2140.0	2949.0
56592	2023-07-05 00:00:00	4061	2044.0	2816.0
56593	2023-07-06 00:00:00	4287	2085.0	2873.0
56594	2023-07-07 00:00:00	4026	2119.0	2921.0
56595	2023-07-08 00:00:00	4492	2139.0	2948.0
56596	2023-07-09 00:00:00	4873	2238.0	3084.0
56597	2023-07-10 00:00:00	2432	2290.0	3156.0
56598	2023-07-11 00:00:00	4904	2338.0	3222.0
56599	2023-07-12 00:00:00	3788	2393.0	3298.0
56600	2023-07-13 00:00:00	4326	2501.0	3447.0
56601	2023-07-14 00:00:00	3927	2519.0	3471.0
56602	2023-07-15 00:00:00	4380	2499.0	3445.0
56603	2023-07-16 00:00:00	728	2609.0	3596.0
56604	2023-07-17 00:00:00	4874	2603.0	3588.0
56605	2023-07-18 00:00:00	1126	2732.0	3766.0
56606	2023-07-19 00:00:00	3222	2613.0	3601.0
56607	2023-07-20 00:00:00	2497	2598.0	3580.0
56608	2023-07-21 00:00:00	1458	2630.0	3625.0
56609	2023-07-22 00:00:00	4944	2588.0	3567.0
56610	2023-07-23 00:00:00	2954	2648.0	3600.0
56611	2023-07-24 00:00:00	3923	2564.0	3533.0
56612	2023-07-25 00:00:00	4953	2564.0	3484.0
56613	2023-07-26 00:00:00	1062	2632.0	3627.0
56614	2023-07-27 00:00:00	3828	2528.0	3485.0
56615	2023-07-28 00:00:00	614	2512.0	3463.0
56616	2023-07-29 00:00:00	2029	2394.0	1290.0
56617	2023-07-30 00:00:00	2141	2309.0	3183.0

ВІДЕО РОБОТИ СИСТЕМИ



11

ПОРІВНЯЛЬНИЙ АНАЛІЗ АЛГОРИТМУ

	Мурашиний алгоритм	Градієнтний спуск
Час навчання, хвилин	0:37	1:16
Навантаження процесору, %	11%	14%
Навантаження ОЗП, МБ	30	67
Кількість ітерацій	100	500
Середня абсолютна помилка	1327.12	1211.53

12

ВИСНОВКИ

1. Досліджено проблему прогнозування попиту, проаналізовано існуючі підходи та виявлено їх ключові особливості.
2. Запропоновано алгоритм на базі мурашиного алгоритму для прогнозування попиту на товар.
3. Розроблено та впроваджено алгоритм навчання нейронної мережі на базі мурашиного алгоритму для оптимізації вагів нейронної мережі.
4. Проведено тестування алгоритму та проведено експериментальне порівняння з алгоритмом прогнозування на базі глибокого навчання за такими метриками, як час навчання, навантаження процесору, навантаження ОЗП, кількість ітерацій навчання, середня абсолютно похибка.

– час навчання зменшився до на 39 секунд (≈48,7%);
– навантаження процесору знизилось на 3% (≈22,4%);
– навантаження ОЗП знизилось на 37 МБ (≈55.3%);
– середня абсолютна похибка збільшилась на 115,59 (≈9,54%).

5. Алгоритм може бути використано для інтеграції прогнозування попиту у малих магазинах роздрібної торгівлі та як додатковий алгоритм прогнозування для великих компаній.

ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ

Тези доповідей:

1. Коденцев М.І., Замрій І.В. Використання мурашиного алгоритму, як ефективної альтернативи вирішення задач оптимізації прогнозування попиту на товар харчової промисловості. Всеукраїнська науково-технічна конференція «Сучасний стан та перспективи розвитку IoT». Збірник тез. – К.: ДУІКТ, 2025. С.149-151.
2. Коденцев М.І., Замрій І.В. Перспективи мурашиного алгоритму, як ефективної альтернативи вирішення задач оптимізації прогнозування попиту на товар харчової промисловості. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». Збірник тез. – К.: ДУІКТ, 2025. С.548-553.

ДОДАТОК Б. ЛІСТИНГИ КОДУ АЛГОРИТМІВ

```
namespace DiplomaAIShop.Controllers.AI;

public interface IArtificialIntelligence
{
    public double Calculate(double[] inputs);
    public void Train(double[][] inputs, double[] outputs);
}

using System.Text.Json;

namespace DiplomaAIShop.Controllers.AI;

public class AntAlgorithm : IArtificialIntelligence,
IDisposable
{
    private readonly int _inputCount;
    private readonly int _antCount;
    private readonly int _iterations;
    private double[,] _pheromones;
    private readonly Random _random;
    private readonly double _evaporation;
    private readonly double _pheromoneBoost;

    public double[] BestWeights { get; private set; }
    public double BestError { get; private set; }

    public AntAlgorithm(int inputCount, int antCount = 20,
int iterations = 1000, double evaporation = 0.5, double
pheromoneBoost = 1.0)
    {
        _inputCount = inputCount;
        _antCount = antCount;
        _iterations = iterations;
        _evaporation = evaporation;
        _pheromoneBoost = pheromoneBoost;
        _pheromones = new double[_inputCount, 2];
        _random = new Random();

        for (int i = 0; i < _inputCount; i++)
        {
            _pheromones[i, 0] = 1.0;
            _pheromones[i, 1] = 1.0;
        }

        LoadParameters();
    }

    public double Calculate(double[] inputs)
    {
        var max = inputs.Max();
        inputs = inputs.Select(x => x / max).ToArray();

        double result = 0;
        for (int i = 0; i < _inputCount; i++)
            result += inputs[i] * BestWeights[i];
        return result * max;
    }

    public void Train(double[][] inputs, double[] outputs)
    {
        double maxInput = inputs.SelectMany(x => x).Max();
        double maxOutput = outputs.Max();
        double max = Math.Max(maxInput, maxOutput);

        double[][] normInputs = inputs
            .Select(arr => arr.Select(x => x / max).ToArray())
            .ToArray();
        double[] normOutputs = outputs
            .Select(x => x / max)
            .ToArray();

        TrainBatch(normInputs, normOutputs);
    }

    private void TrainBatch(double[][] inputs, double[]
outputs)
    {
        BestError = double.MaxValue;
        BestWeights = new double[_inputCount];

        for (int iter = 0; iter < _iterations; iter++)
        {
            double[][] antsWeights = new
double[_antCount][];
            double[] antsErrors = new double[_antCount];

            for (int ant = 0; ant < _antCount; ant++)
            {
                antsWeights[ant] = new double[_inputCount];
                for (int i = 0; i < _inputCount; i++)
                {
                    double pMinus = _pheromones[i, 0];
                    double pPlus = _pheromones[i, 1];
                    double direction = (_random.NextDouble() <
pPlus / (pPlus + pMinus)) ? 1 : -1;
                    antsWeights[ant][i] = Math.Max(-1,
Math.Min(1, direction * _random.NextDouble()));
                }
                antsErrors[ant] =
CalculateError(antsWeights[ant], inputs, outputs);
            }

            int bestAnt = Array.IndexOf(antsErrors,
antsErrors.Min());
            if (antsErrors[bestAnt] < BestError)
            {
                BestError = antsErrors[bestAnt];
                Array.Copy(antsWeights[bestAnt], BestWeights,
_inputCount);
            }

            for (int i = 0; i < _inputCount; i++)
```

```
        {
            _pheromones[i, 0] *= (1 - _evaporation);
            _pheromones[i, 1] *= (1 - _evaporation);
        }

        for (int i = 0; i < _inputCount; i++)
        {
            if (BestWeights[i] > 0)
                _pheromones[i, 1] += _pheromoneBoost;
            else
                _pheromones[i, 0] += _pheromoneBoost;
        }
    }
  }

  private double CalculateError(double[] weights,
double[][] inputs, double[] outputs)
  {
    double error = 0;
    for (int i = 0; i < inputs.Length; i++)
    {
      double predicted = 0;
      for (int j = 0; j < _inputCount; j++)
        predicted += inputs[i][j] * weights[j];
      error += Math.Pow(predicted - outputs[i], 2) + 0.01
* weights.Sum(w => w * w);
    }
    return error / inputs.Length;
  }

  public void SaveParameters(string filePath =
"ant_parameters.json")
  {
    var parameters = new AntParameters
    {
      BestWeights = BestWeights,
      Pheromones = ToJaggedArray(_pheromones)
    };

    var json = JsonSerializer.Serialize(parameters);
    File.WriteAllText(filePath, json);
  }

  public void LoadParameters(string filePath =
"ant_parameters.json")
  {
    if (!File.Exists(filePath))
      return;

    var json = File.ReadAllText(filePath);
    var parameters =
JsonSerializer.Deserialize<AntParameters>(json);
    if (parameters != null)
    {
      BestWeights = parameters.BestWeights;
      _pheromones =
ToMatrix(parameters.Pheromones);
    }
  }

  private double[][] ToJaggedArray(double[,] matrix)
  {
    int rows = matrix.GetLength(0);
    int cols = matrix.GetLength(1);
    var result = new double[rows][];
    for (int i = 0; i < rows; i++)
    {
      result[i] = new double[cols];
      for (int j = 0; j < cols; j++)
        result[i][j] = matrix[i, j];
    }
    return result;
  }

  private double[,] ToMatrix(double[][] jagged)
  {
    int rows = jagged.Length;
    int cols = jagged[0].Length;
    var result = new double[rows, cols];
    for (int i = 0; i < rows; i++)
      for (int j = 0; j < cols; j++)
        result[i, j] = jagged[i][j];
    return result;
  }

  public void Dispose()
  {
    SaveParameters();
  }

  private class AntParameters
  {
    public double[] BestWeights { get; set; }
    public double[][] Pheromones { get; set; }
  }
}

using System.Text.Json;

namespace DiplomaAIShop.Controllers.AI;

public class Perceptron : IArtificialIntelligence, IDisposable
{
  private readonly int _inputCount;
  private readonly int _hiddenCount;
  private readonly int _outputCount;

  private double[,] _weightsInputHidden;
  private double[,] _weightsHiddenOutput;
  private readonly Random _random;
  private readonly double _learningRate;
  private readonly int _epoch;

  public Perceptron()
  {
    _inputCount = 7;
    _hiddenCount = 3;
    _outputCount = 1;

    _weightsInputHidden = new double[_inputCount,
_hiddenCount];
    _weightsHiddenOutput = new double[_hiddenCount,
_outputCount];
    _random = new Random();
    _learningRate = 0.05;
    _epoch = 1000;
```

```
        GetParameters();
    }

    private void GetParameters()
    {
        if (File.Exists("perceptron_parameters.json"))
        {
            var json =
File.ReadAllText("perceptron_parameters.json");
            var parameters =
JsonSerializer.Deserialize<Dictionary<string,
double[][]>>(json);
            if (parameters != null)
            {
                _weightsInputHidden =
ToMatrix(parameters["WeightsInputHidden"]);
                _weightsHiddenOutput =
ToMatrix(parameters["WeightsHiddenOutput"]);
            }
        }

        else
        {
            for (int i = 0; i <
_weightsInputHidden.GetLength(0); i++)
                for (int j = 0; j <
_weightsInputHidden.GetLength(1); j++)
                    _weightsInputHidden[i, j] =
_random.NextDouble() * 2 - 1;

            for (int i = 0; i <
_weightsHiddenOutput.GetLength(0); i++)
                for (int j = 0; j <
_weightsHiddenOutput.GetLength(1); j++)
                    _weightsHiddenOutput[i, j] =
_random.NextDouble() * 2 - 1;
        }
    }

    public double Calculate(double[] inputs)
    {
        double max = inputs.Max();

        for (int i = 0; i < inputs.Length; i++)
            inputs[i] /= max;

        // Forward pass
        double[] firstHidden = new double[_hiddenCount];
        for (int j = 0; j < _hiddenCount; j++)
        {
            double sum = 0;
            for (int i = 0; i < _inputCount; i++)
                sum += inputs[i] * _weightsInputHidden[i, j];
            firstHidden[j] = Sigmoid(sum);
        }

        double output = 0;
        for (int i = 0; i < _hiddenCount; i++)
            output += firstHidden[i] * _weightsHiddenOutput[i,
0];
        output = Sigmoid(output);
        return output * max;
    }

    public void Train(double[][] inputs, double[] outputs)
    {
        double maxInput = inputs.SelectMany(x => x).Max();
        double maxOutput = outputs.Max();
        double max = Math.Max(maxInput, maxOutput);

        double[][] normInputs = inputs
            .Select(arr => arr.Select(x => x / max).ToArray())
            .ToArray();

        double[] normOutputs = outputs
            .Select(x => x / max)
            .ToArray();

        for (int epoch = 0; epoch < _epoch; epoch++)
        {
            for (int sample = 0; sample < normInputs.Length;
sample++)
            {
                double[] input = normInputs[sample];
                double expected = normOutputs[sample];

                double[] firstHidden = new
double[_hiddenCount];
                for (int j = 0; j < _hiddenCount; j++)
                {
                    double sum = 0;
                    for (int i = 0; i < _inputCount; i++)
                        sum += input[i] * _weightsInputHidden[i, j];
                    firstHidden[j] = Sigmoid(sum);
                }

                double output = 0;
                for (int i = 0; i < _hiddenCount; i++)
                    output += firstHidden[i] *
_weightsHiddenOutput[i, 0];
                output = Sigmoid(output);

                double deltaOutput = (expected - output) *
output * (1 - output);

                double[] deltaFirstHidden = new
double[_hiddenCount];
                for (int i = 0; i < _hiddenCount; i++)
                {
                    double sum = deltaOutput *
_weightsHiddenOutput[i, 0];
                    deltaFirstHidden[i] = sum * firstHidden[i] * (1 -
firstHidden[i]);
                }

                for (int i = 0; i < _hiddenCount; i++)
                    _weightsHiddenOutput[i, 0] += _learningRate
* deltaOutput * firstHidden[i];

                for (int i = 0; i < _inputCount; i++)
                    for (int j = 0; j < _hiddenCount; j++)
                        _weightsInputHidden[i, j] += _learningRate
* deltaFirstHidden[j] * input[i];
            }
```

```
        }
    }

    public void Dispose()
    {
        var parameters = new
        {
            WeightsInputHidden =
ToJaggedArray(_weightsInputHidden),
            WeightsHiddenOutput =
ToJaggedArray(_weightsHiddenOutput),
        };

        var json = JsonSerializer.Serialize(parameters);
        File.WriteAllText("perceptron_parameters.json",
json);
    }

    private double Sigmoid(double x) => 1 / (1 + Math.Exp(-
x));

    private double[][] ToJaggedArray(double[,] matrix)
    {
        int rows = matrix.GetLength(0);
        int cols = matrix.GetLength(1);
        var result = new double[rows][];
        for (int i = 0; i < rows; i++)
        {
            result[i] = new double[cols];
            for (int j = 0; j < cols; j++)
                result[i][j] = matrix[i, j];
        }
        return result;
    }

    private double[,] ToMatrix(double[][] jagged)
    {
        int rows = jagged.Length;
        int cols = jagged[0].Length;
        var result = new double[rows, cols];
        for (int i = 0; i < rows; i++)
            for (int j = 0; j < cols; j++)
                result[i, j] = jagged[i][j];

        return result;
    }
}
```