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

# КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Метод прогнозування вартості продуктової корзини за допомогою нейронних мереж»

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*

_____ Максим КУТНЯК
*(підпис)*

Виконав: здобувач вищої освіти групи ПДМ-63
Максим КУТНЯК

Керівник: Владислав ЯСКЕВИЧ
к.т.н.

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

**Київ 2025**

**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
«_____» _____ 2024 р.

# ЗАВДАННЯ
# НА КВАЛІФІКАЦІЙНУ РОБОТУ

Кутняку Максиму Юрійовичу

1. Тема кваліфікаційної роботи: «Метод прогнозування вартості продуктової корзини за допомогою нейронних мереж»

керівник кваліфікаційної роботи Владислав ЯСКЕВИЧ, к.т.н.,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «15» жовтня 2024 р. № 320.

2. Строк подання кваліфікаційної роботи «17» грудня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, параметри навчання нейронної мережі, метод прогнозування вартості продуктової корзини, вимоги до точності прогнозування.

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

1. Дослідження методів прогнозування продуктової корзини.
2. Аналіз технологій прогнозування за допомогою нейронних мереж.
3. Розробка та тестування методу прогнозування продуктової корзини за допомогою нейронних мереж.

5. Перелік ілюстративного матеріалу: *презентація*

1. Мета, об’єкт та предмет дослідження
2. Порівняльний аналіз
3. Послідовність Дій Роботи методу
4. Математична модель
5. Математична модель
6. Програмні засоби реалізації
7. Результат прогнозу
8. Оцінка точності
9. Порівняння точності
10. Висновки
11. Публікації та апробація роботи

6. Дата видачі завдання «16» Жовтня 2024 Р.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Аналіз наявної науково-технічної літератури | 16.10-23.10.2024 | |
| 2 | Вивчення матеріалів для аналізу вартості продуктової корзини | 24.10-31.10.2024 | |
| 3 | Дослідження методів навчання нейронних мереж | 1.11-5.11.2024 | |
| 4 | Аналіз особливостей впливу різних факторів на вартість продуктової корзини | 6.11-10.11.2024 | |
| 5 | Дослідження технологій навчання нейронних мереж | 11.11-16.11.2024 | |
| 6 | Застосування нейронних мереж для прогнозування вартості продуктової корзини | 17.11-25.11.2024 | |
| 7 | Оформлення роботи: вступ, висновки, реферат | 25.11-26.11.2024 | |
| 8 | Розробка демонстраційних матеріалів | 27.11-28.11.2024 | |
| 9 | Попередній захист роботи | 29.11.2024 | |

Здобувач вищої освіти _____ Максим КУТНЯК
*(підпис)*

Керівник
кваліфікаційної роботи _____ Владислав ЯСКЕВИЧ
*(підпис)*

# РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 102 стор., 1 табл., 25 рис., 31 джерел.

***Мета роботи**: підвищення точності прогнозування вартості продуктової корзини внаслідок використання нейронних мереж.*

***Об'єкт дослідження:** процес прогнозування вартості продуктової корзини.*

***Предмет дослідження**: методи прогнозування вартості продуктової корзини.*

У роботі використано різноманітні методи, такі як методи прогнозування, методи навчання нейронних мереж, технології об'єктно-орієнтованого програмування. Головний акцент зроблено на методах прогнозування з використанням нейронних мереж для покращення точності прогнозування.

Проведено аналіз сучасних методів прогнозування та навчання нейронних мереж. Глибоко вивчено принципи побудови нейронних мереж та їх застосовність у завданні прогнозування вартості продуктової корзини.

Розроблено метод для підвищення точності прогнозування продуктової корзини.

Реалізовано програмну систему, яка використовує розроблений метод для точного прогнозування вартості продуктової корзини. Застосовано технології об'єктно-орієнтованого програмування для розробки програмного забезпечення.

Проведено експерименти для валідації розробленого методу та порівняння його ефективності з існуючими підходами. Оцінено продуктивність та точність системи прогнозування на реальних даних.

Результати дослідження підтверджують успішність та перспективність використання нейронних мереж для задач прогнозування. Розроблений метод виправдовує очікувані результати та може знайти широке застосування в галузі прогнозування вартості нейронних мереж.

КЛЮЧОВІ СЛОВА: НЕЙРОННІ МЕРЕЖІ, ПРОДУКТОВА КОРЗИНА, МЕТОДИ ПРОГНОЗУВАННЯ, МАШИННЕ НАВЧАННЯ.

## ABSTRACT

The text part of the qualification work for the master's degree: 102 pages, 1 tables, 25 figures, 31 sources.

Purpose: to improve the accuracy of predicting the cost of a grocery basket by using neural networks.

Object of research: the process of forecasting the cost of a grocery basket.

Subject of research: methods of forecasting the cost of a grocery basket.

Various methods were used in the study, such as forecasting methods, neural network training methods, and object-oriented programming technologies. The main emphasis is placed on forecasting methods using neural networks to improve forecasting accuracy.

The analysis of modern methods of forecasting and training of neural networks is carried out. The principles of building neural networks and their applicability in the task of forecasting the cost of a grocery basket are studied in depth.

A method for improving the accuracy of grocery basket forecasting is developed.

A software system has been implemented that uses the developed method to accurately predict the cost of a grocery basket. The technologies used are object-oriented programming technologies for software development.

Experiments were conducted to validate the developed method and compare its effectiveness with existing approaches. The performance and accuracy of the forecasting system on real data are evaluated.

The results of the study confirm the success and prospects of using neural networks for forecasting tasks. The developed method justifies the expected results and can be widely used in the field of neural network value prediction.



KEYWORDS: NEURAL NETWORKS, GROCERY BASKET, FORECASTING METHODS, MACHINE LEARNING.

**ЗМІСТ**

ВСТУП ..... 11
1 АНАЛІЗ ФАКТОРІВ ЩО ВПЛИВАЮТЬ НА ЦІНУ ПРОДУКТОВОГО КОШИКА ..... 16
1.1. Вибір основних факторів для аналізу ..... 16
1.2. Економічні фактори ..... 17
1.2.1. Курс валют ..... 17
1.2.2. Індекс споживчих цін ..... 18
1.2.3. Витрати на паливо та енергію ..... 18
1.3. Екологічні фактори ..... 19
1.4. Політичні фактори ..... 21
1.5. Глобальні фактори ..... 22
1.6. Логістичні фактори ..... 24
2 АНАЛІЗ МОДЕЛЕЙ НАВЧАННЯ НЕЙРОННИХ МЕРЕЖ ..... 25
2.1. Багатошаровий перцептрон (MLP) ..... 25
2.2. Лінійна регресія ..... 29
2.3. Дерево рішень ..... 33
2.4. Рандомні ліси ..... 37
2.5. Згорткові нейронні мережі ..... 39
2.5.1. LeNet ..... 43
2.5.2. AlexNet ..... 44
2.5.3. Resnet ..... 44
2.5.4. GoogleNet ..... 45
2.5.5. MobileNet ..... 46
2.5.6. VGG ..... 46
2.6. Рекурентна нейронна мережа ..... 48
3 РОЗРОБКА МЕТОДУ ДЛЯ ПРОГНОЗУВАННЯ ВАРТОСТІ ПРОДУКТОВОГО КОШИКУ ЗА ДОПОМОГОЮ НЕЙРОННИХ МЕРЕЖ ..... 56
3.1. Опис використаних програмних засобів ..... 56
3.1.1. Python ..... 56
3.1.2. Jupyter lab ..... 59
3.1.3. Jupyter Notebook ..... 60

3.1.4. Pandas ..... 62
3.1.5. NumPy ..... 65
3.1.6. Matplotlib ..... 69
3.1.7. Scikit-learn ..... 72
3.2. Опис методу прогнозування вартості продуктової корзини ..... 74
3.3. Розробка програмного застосунку ..... 75
3.4. Результат роботи ..... 81
ВИСНОВКИ ..... 90
ПЕРЕЛІК ПОСИЛАНЬ ..... 92
ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ ..... 95
ДОДАТОК Б. ЛІСТИНГ ОСНОВНОГО МОДУЛЮ ..... 101

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

FAO – Індекс цін на їжу (Food Price Index)

MLP – Багатошаровий перцептрон

ReLU – зрізаний лінійний вузол

MSE - середньоквадратична помилка (Mean Squared Error)

CNN – згорткові нейроні мережі

# ВСТУП

*Актуальність дослідження.* В наші часи тема прогнозування вартості продуктової корзини є надзвичайно актуальною темою, адже в сучасному світі існують умови дуже динамічної зміни цін на продукти харчування, інфляція та глобальні економічні зміни. Прогнозування вартості продуктової корзини дасть змогу ефективніше планувати свій сімейний бюджет користувачам, для багатьох домогосподарств витрати на продукти є однією з найбільших витрат, прогнозування вартості продуктів також стане для них дуже корисним інструментом для планування бюджету, Для рітейлерів, дистриб'юторів і виробників продуктів прогнозування цін дозволяє оптимізувати закупівлі, управляти складськими запасами, а також краще адаптуватися до змін у споживчому попиті. Технології стають більш доступними, та методи штучного інтелекту стають більш поширеними що дає змогу навчати складні системи штучного інтелекту за адекватний час та розробляти подібні системи за адекватний час. Зараз є великий обсяг даних про споживчі ціни, що може бути достатнім для побудови більш-менш точної системи нейронної мережі. Вибір штучного інтелекту для подібної задачі зумовлений можливістю враховувати велику кількість факторів які впливають на ціни продуктів та можливість обробляти велику кількість даних. Отже, використання нейронних мереж для прогнозування вартості продуктової корзини може принести користь як споживачам, так і бізнесу, дозволяючи більш точно прогнозувати ціни та планувати витрати.

***Мета роботи**: підвищення точності прогнозування вартості продуктової корзини внаслідок використання нейронних мереж.*

***Об'єкт дослідження:** процес прогнозування вартості продуктової корзини*.

***Предмет дослідження**: методи прогнозування вартості продуктової корзини.*

Для досягнення поставленої мети необхідно вирішити наступні завдання:

1. Провести детальний огляд існуючих наукових праць та методів у галузі прогнозування економічних показників, зокрема вартості продуктового

кошика. Проаналізувати підходи, що використовуються для обробки та аналізу цінових даних, зокрема методи машинного навчання, регресійного аналізу, часових рядів та їхніх гібридних модифікацій. Виявлено переваги та недоліки кожного з методів.

2. Розробити математичну модель для передбачення вартості продуктового кошика. У моделі враховано специфіку роботи з реальними економічними даними,.
3. Оцінити точність моделі.
4. На основі аналізу недоліків існуючих підходів було запропонувати вдосконалений метод передбачення та протестувати метод на реальних даних.

Вирішення цих задач сприяє аналізу, тестуванню та розвитку різних підходів у дослідженні та розробці методів прогнозування вартості продуктової корзини за допомогою нейронних мереж.

Для досягнення поставленої мети використовувались наступні методи дослідження:

Аналіз науково-технічної літератури, пов'язаної з методами прогнозування та навчання нейронних мереж.

Огляд сучасних методів прогнозування вартості продуктового кошика, включаючи аналіз економічних, екологічних, політичних та інших факторів.

Оцінка точності моделі та порівняння з іншими методами прогнозування.

Використання сучасних інструментів програмування для розробки та тестування програмного забезпечення.

Дослідження та використання різних типів нейронних мереж, таких як багатошаровий перцептрон (MLP), згорткові та рекурентні нейронні мережі.

Навчання та валідація нейронної мережі на реальних даних.

Аналіз продуктивності та точності прогнозування розробленої системи на основі реальних даних.

Наукова новизна:

У роботі запропоновано метод підвищення точності прогнозування вартості продуктового кошика на основі нейронних мереж, який враховує багатофакторний вплив економічних, еколгічних, політичних та логістичних чинників.

Використано сучасні підходи машинного навчання, включаючи багатошаровий перцептрон (MLP) і згорткові нейронні мережі, що дозволило створити ефективний алгоритм для прогнозування цін.

Деталізовано вплив різних факторів на вартість продуктової корзини, включаючи екологічні, економічні та політичні. Проведено їх інтеграцію в навчальну модель, що є важливим для підвищення точності прогнозів.

Виконано порівняльний аналіз запропонованого підходу з іншими існуючими методами прогнозування, що демонструє переваги використання розробленого алгоритму.

Теоретична значущість:

У роботі представлено детальний аналіз сучасних методів прогнозування вартості продуктового кошика із застосуванням нейронних мереж. Це сприяє розвитку теорії прогнозування економічних показників.

Запропоновано нову математичну модель, що враховує багатофакторний вплив на вартість продуктового кошика. Ця модель може бути використана як основа для подальших наукових досліджень.

Визначено основні фактори, що впливають на вартість продуктової корзини (економічні, екологічні, політичні, логістичні тощо), що є основою для формування ефективних моделей прогнозування.

Практична значущість:

Результати роботи мають значення для домогосподарств, допомагаючи ефективно планувати бюджет, а також для підприємств, які можуть використовувати розроблену систему для оптимізації своїх бізнес-процесів (планування закупівель, складських запасів, ціноутворення).

Розроблене програмне забезпечення на основі запропонованого методу може бути впроваджено в корпоративні інформаційні системи та державні структури для аналізу цінової динаміки.

Інструмент прогнозування може допомогти державним органам та компаніям приймати більш обґрунтовані рішення в умовах змін ринку.

Методика та результати дослідження можуть бути використані у навчальному процесі для підготовки студентів за напрямами, пов’язаними з програмною інженерією, економікою та аналітикою даних.

Робота пройшла апробацію на наступних конференціях:

Тези доповідей:

1. Кутняк М. Ю., Яскевич В. О. Аналіз алгоритму «Зворотного поширення помилки» для навчання нейронних мереж // «Всеукраїнська науково-технічна конференція "Виклики та рішення в програмній інженерії» . – Київ: ДУІКТ, 2024. – С.101-104

2. Кутняк М.Ю., Яскевич В. О. Порівняльний аналіз популярних фреймворків Java // Всеукраїнська науково-технічна конференція «Виклики та рішення в програмній інженерії» . – Київ: ДУІКТ, 2024. – С.243-245.

# 1 АНАЛІЗ ФАКТОРІВ ЩО ВПЛИВАЮТЬ НА ЦІНУ ПРОДУКТОВОГО КОШИКА

## 1.1. Вибір основних факторів для аналізу

Загалом фактори що впливають на вартість цін продуктового кошика можна розділити на наступні категорії:

- Економічні фактори – це фактори, що пов’язані з обігом грошей, товарів, інформації та енергії [2]. До цих факторів можна віднести курс валют, індекс споживчих цін, витрати на паливо та енергію, ціни на добрива та аграрну техніку;
- Екологічні фактори – це фактори, що мають вплив на функціонування живих організмів [3]. В даному контексті це також дуже важлива категорія факторів, адже ціна на продукти безпосередньо залежить від врожайності та кількості продуктів на ринку на що впливають погодні умови, екологічний стан, сезонність для певних видів продуктів і тд. До цієї категорії можна віднести наступні фактори що впливають на ціну продуктового кошика: сезонність, погодні умови, врожайність сільськогосподарських культур.
- Політичні фактори – це фактори на які уряд має, або може мати вплив. Сюди можна віднести державну політику, політичну стабільність чи нестабільність, корупцію, зовнішньоторговельну політику, податкову політику, трудове законодавство, екологічне законодавство [4]. В контексті впливу на ціни продуктового кошика можна виділити наступні фактори: Регуляція цін державою на базові продукти, митні тарифи та дані про експорт та імпорт продуктів;

- Глобальні фактори – це фактори що не прив’язані до конкретного ринку. До них можна віднести Ціни на сировинні товари на міжнародних ринках та світові економічні тренди;
- Логістичні фактори – це фактори що впливають на доставку та логістичні шляхи. До них можна віднести ціни на транспорті послуги та доступність логістичних шляхів

## 1.2. Економічні фактори

### 1.2.1. Курс валют

Курс валют є одним з ключових факторів, що впливає на вартість продуктового кошика, особливо в країні з відкритою економікою в яку як імпортуються деякі товари, так і експортуються. Багато продуктів, особливо фрукти, морепродукти, риба, екзотичні продукти, а також сировина в Україні імпортуються. При зростанні курсу валют (девальвація гривні) вартість імпортованих товарі зростає, оскільки закупівля відбувається в основному в або доларах США, або в Євро, а ціни в гривні формуються шляхом перерахунку, що безпосередньо впливає на ціни для кінцевого споживача на ринку. Та при зміцненні Гривні відбувається зворотна реакція, та ціна на імпортовані товари падає. Але в Україну не тільки імпортують товари, а ще й експортують. Україна є величезним експортером аграрної продукція такої як зерно, м’ясо, олія молочна продукція і т.д. При девальвації гривні експорт стає вигіднішим, оскільки товари за кордоном як купляються у міжнародній валюті, так і продаються, через що виручка у валюті стає більшою при конвертації в гривні. Через це ціна на внутрішньому ринку буде росте, тому що більше товару буде йти на експорт і менше залишатися на локальному ринку.

Також курс валют впливає на ціну сировини та витрати на виробництво. Багато складових залежать від імпорту. Наприклад: імпортні добрива, аграрна техніка та запчастини для її ремонту, імпортний корм для тварин і так далі. Також останню роль в фінальній ціні на продукти займає ціна на енергетичні ресурси, такі

як газ, пальне, електроенергію, які також залежать від курсі, що впливає на вартість транспортування , зберігання та переробки продуктів.

Для врахування цього фактору у моделі нейронної мережі можна взяти історичні дані курсу та використовувати їх для прогнозування вартості продуктової корзини, також необхідно виявити залежність на які саме товари та як впливає курс.

Зміна курсу валют має як прямий, так і опосередкований вплив, тому врахування його динаміки є критично важливим для точного прогнозування цін.

### 1.2.2. Індекс споживчих цін

Індекс інфляції, або, що теж саме, індекс споживчих цін (ІСЦ) — показник, що характеризує зміни загального рівня цін на товари та послуги, які купує населення для невиробничого споживання [5]. ІСЦ є важливим фактором при аналізі вартості продуктового кошика, оскільки: підвищення ІСЦ свідчить про загальне подорожання товарів та послуг. ІСЦ враховує вартість основних продуктів харчування, енергетичних ресурсів та інших товарів та послуг. ІСЦ є корисною змінною для моделювання, оскільки: Відображає загальні інфляційні тенденції, що впливають на ціни продуктів. Може слугувати індикатором зміни витрат домогосподарств. Допомагає виявити кореляцію між зростанням інфляції та цінами на продукти харчування. Для нейронної мережі ІСЦ можна включити як часову змінну, наприклад, у розрізі місяців або кварталів. Актуальні дані про ІСЦ є на офіційному сайті Державної служби статистики України [6].

### 1.2.3. Витрати на паливо та енергію

Витрати на паливо та енергію суттєво впливають на кінцеву вартість продуктів у продуктовому кошику через їх значення у виробництві, транспортуванні та зберіганні.

Зростання цін на пальне (бензин, дизель):

Призводить до збільшення транспортних витрат. Це впливає як на доставку сировини на виробничі підприємства, так і на перевезення готових продуктів до торговельних мереж. Наприклад, у 2022–2023 роках в Україні через дефіцит

пального та зростання цін на світових ринках вартість логістики суттєво зросла: Руйнування нафтобаз, скорочення постачання пального та зростання залежності від імпорту створюють додаткові труднощі для аграріїв і транспортних компаній, що впливає на кінцеву ціну продуктів.

Ціни на електроенергію:

Виробництво багатьох продуктів харчування залежить від енергоємного обладнання. Зростання тарифів на електроенергію прямо збільшує собівартість продукції. Наприклад, підвищення цін на електроенергію для підприємств у 2023 році стало однією з причин подорожчання хлібобулочних виробів . Ціни на газ є ключовим енергоресурсом для тепличного господарства, виробництва хліба та багатьох інших продуктів. Підвищення цін на газ збільшує ціни на овочі, вирощені в теплицях, та випічку. Багато продуктів потребують спеціальних умов зберігання, що залежить від електроенергії. Зростання тарифів на електроенергію автоматично підвищує ціни на такі продукти. Вирощування овочів у теплицях залежить від газу, електрики або твердого палива. Підвищення цін на ці ресурси збільшує витрати виробників. Дані для аналізу можна взяти на офіційному ресурсі Міністерства енергетики України [7], Державна служба статистики України [6] також збирає інформацію про ціни на паливо та енергоресурси.

## 1.3. Екологічні фактори

Екологічні фактори мають вагомий вплив на формування вартості продуктового кошика, оскільки ми маємо справу з аграрною промисловістю. Ці фактори визначають рівень врожайності та сезонні коливання цін. Ці фактори можна розділити на 2 категорії: природні та сезонні.

Природні фактори відіграють ключову роль у формуванні вартості продуктового кошика, оскільки вони впливають на рівень врожайності, собівартість виробництва та доступність продукції. Ці фактори включають погодні умови, кліматичні зміни, екологічні катастрофи та ґрунтово-географічні умови. До них можна віднести наступні фактори:

- Температура. Екстремальні температури негативно впливають на врожайність сільськогосподарських культур. Наприклад, заморозки в травні можуть знищити фруктові дерева, а спека у липні знижує врожай зернових.
- Опади. Дефіцит опадів або надмірна вологість впливають на якість і кількість врожаю. Посуха: Зменшує врожайність зернових, соняшнику, овочів. Надлишок опадів: Може призводити до гниття врожаю або розвитку хвороб рослин. Як приклад можна навести сезон 2019/2020, коли фермери вкотре гостро відчули нестачу продуктивної вологи в ґрунті, чекали дощів і сходів озимих значно довше, ніж завжди. Через те, що восени в деяких регіонах України сильно бракувало вологи і дуже багато культур сіяли в сухий грунт, то сільгосппідприємства недосіяли півмільйона гектарів по Україні. Реальний «недосів» тільки озимої пшениці склав близько 10% [8].
- Родючість ґрунтів. Чорноземи України забезпечують високий потенціал урожайності, але їх виснаження через інтенсивне використання або неправильне удобрення може знижувати ефективність сільськогосподарського виробництва.
- Сезонність. У період збору врожаю (літо–осінь) спостерігається зниження цін на овочі, фрукти, ягоди завдяки їхній надлишковій пропозиції. У зимово-весняний період ціни зростають через витрати на зберігання, імпорт або вирощування у теплицях. Перед релігійними чи національними святами ціни на певні категорії товарів (м'ясо, яйця, цукор) можуть зростати через підвищений попит. Деякі продукти навіть тваринного походження також мають сезонність. Наприклад молоко також має сезонність та влітку зазвичай коштує дешевше.
- Зміна клімату. Підвищення температури призводить до необхідності адаптації агротехнологій, впровадження посухостійких сортів рослин або збільшення площі зрошення. Тривалі зміни клімату можуть

скоротити періоди традиційного вирощування культур і знизити врожайність.

## 1.4. Політичні фактори

Політична ситуація в країні відіграє значну роль у формуванні цін на продукти харчування. Вона може впливати як безпосередньо через закони, субсидії або санкції, так і опосередковано через економічну та соціальну стабільність. Політичні фактори впливають на формування вартості продуктового кошика через регулювання аграрного сектору, підтримку або обмеження виробників, а також через міжнародну політику. Наприклад, субсидії для сільського господарства дозволяють знизити собівартість продукції. Це можливо завдяки підтримці фермерів у вигляді фінансування добрив, техніки чи пального.

Якщо така підтримка скорочується, витрати виробників збільшуються, що сприяє зростанню цін.

Додатково, експортні або імпортні мита можуть значно вплинути на ціни. Обмеження експорту зернових, наприклад, допомагає зберігати продовольчу безпеку, але може вплинути на рентабельність виробництва. У свою чергу, високі мита на імпортовані продукти (такі як цукор чи м'ясо) підвищують їхню вартість для споживачів.

Політична стабільність також має ключове значення. Військові конфлікти, як показав досвід війни в Україні, можуть зруйнувати логістичні ланцюги, обмежити транспортування продуктів, підвищити витрати на страхування вантажів і призвести до дефіциту певних товарів. Політичні кризи, часті зміни уряду або парламенту створюють невизначеність для бізнесу, що також впливає на ціни.

Економічна політика уряду, зокрема монетарна та фіскальна, має значний вплив. Наприклад, підвищення облікової ставки ускладнює кредитування аграрного сектору, збільшуючи виробничі витрати. З іншого боку, зниження ПДВ на базові продукти може зменшити їхню вартість.

Зовнішня політика також грає важливу роль. Міжнародні санкції або торговельні обмеження можуть вплинути на доступність аграрної техніки, добрив або продовольства, підвищуючи собівартість виробництва. Угода про зону вільної торгівлі або зміна митного регулювання може вплинути на кінцеву ціну продукції для споживачів.

Окрім того, законодавство щодо використання добрив, пестицидів і екологічних вимог додає змін у витрати виробників. Наприклад, обмеження на використання певних хімікатів змушує фермерів шукати альтернативи, що може бути дорожчим.

Нарешті, формування стратегічних запасів урядом може суттєво змінити ціноутворення. Закупівля великих обсягів продукції для резервів зменшує доступність товарів на ринку, що може підвищити їхню ціну для споживачів.

Ці фактори слід враховувати як кількісні та категорійні змінні при навчанні нейронних мереж для прогнозування цін. Дані можна отримувати від Міністерства аграрної політики України [9].

## 1.5. Глобальні фактори

Глобальні фактори, що впливають на вартість продуктового кошика, відображають взаємодію міжнародних ринків, кліматичних змін, політичних відносин і демографічних змін. Для навчання нейронної мережі ці фактори можна подавати у вигляді числових, категорійних або часових змінних, що представляють глобальні тренди та події.

Міжнародні ціни на сировину та енергоресурси є одним із ключових чинників. Світові ціни на нафту, газ і електроенергію безпосередньо впливають на витрати виробництва та транспортування продуктів. Наприклад, зростання вартості нафти підвищує витрати на паливо для сільськогосподарської техніки й транспортування, що збільшує кінцеву ціну продуктів. Для нейромережі можна використовувати такі індикатори, як Brent Crude Oil Index або спотові ціни на газ.

Зміни на світовому ринку продовольства також суттєво впливають. Наприклад, індекс FAO (Food Price Index) відображає глобальні тенденції у цінах на основні продукти харчування (зернові, цукор, олії). Коливання цього індексу свідчать про доступність продовольства у світі та впливають на локальні ціни в імпортозалежних країнах.

Курс валют і міжнародна торгівля мають вирішальне значення для країн, залежних від імпорту продуктів або аграрної техніки. Наприклад, послаблення національної валюти збільшує вартість імпортованих товарів, таких як пестициди чи обладнання. Для моделі можна використовувати показники валютних пар, як-от USD/UAH, EUR/UAH.

Глобальні кліматичні зміни також значно впливають на сільське господарство. Зміна температур, частота посух, паводків або ураганів порушують глобальні виробничі ланцюги. Для аналізу можна використовувати кліматичні індекси, такі як ENSO (El Niño Southern Oscillation) або глобальні звіти про посухи.

Пандемії та глобальні кризи відіграють роль у порушенні поставок і зміні попиту. Наприклад, пандемія COVID-19 вплинула на вартість логістики, закриття ринків і зростання витрат на продовольство. Ці події можуть бути представлені через бінарні змінні (криза/немає кризи).

Демографічні зміни у світі також впливають на попит на продукти. Зростання населення в регіонах із високим рівнем споживання зернових чи олії створює додатковий тиск на світові ринки. Дані про демографічні зміни можна використовувати з відкритих баз ООН (United Nations Population Division).

Глобальна політика і торгові санкції впливають на доступність продуктів. Наприклад, санкції проти великих експортерів зерна можуть спричинити дефіцит на ринку й підвищення цін.

## 1.6. Логістичні фактори

Логістика охоплює транспортування, зберігання, управління постачанням та розподілом продукції. Ці процеси мають прямий вплив на кінцеву вартість продуктів. У нейронній мережі логістичні фактори можна врахувати у вигляді числових показників (витрати, час транспортування) або категорійних змінних (тип транспорту, наявність перешкод).

Ціна на транспортування безпосередньо впливає на вартість продуктів, особливо якщо вони імпортуються або доставляються на великі відстані. Основними змінними можна виокремити ціни на паливо та тип транспорту яким доставляються продукти. Якість доріг, доступність залізничних вузлів, портів та аеропортів визначають швидкість та витрати на транспортування. У певні періоди року логістика ускладняються через погодні умови. Наприклад, взимку зростають витрати на транспортування через ускладнення руху, а в дощовий сезон можуть виникати перебої з доставкою. Продукти харчування часто потребують спеціальних умов зберігання: холодильники, склади, контроль вологості. Протяжність і складність ланцюгів постачання впливають на ціни. Переривання ланцюгів (збої в логістиці, закриття кордонів, дефіцит транспорту) можуть спричиняти подорожчання. Логістика може бути порушена через санкції, закриття торговельних шляхів, військові конфлікти. Продукти, які експортуються чи імпортуються, потребують додаткових витрат на митне оформлення, тарифи, страхування.

# 2 АНАЛІЗ МОДЕЛЕЙ НАВЧАННЯ НЕЙРОННИХ МЕРЕЖ

## 2.1. Багатошаровий перцептрон (MLP)

Багатошаровий перцептрон (MLP) - це штучна нейронна мережа з прямим зв'язком, що має щонайменше три рівні вузлів: вхідний шар, один або кілька прихованих шарів і вихідний шар [11].

MLP в машинному навчанні є одним з найпоширеніших видів нейронних мереж. Його можна застосовувати для виконання різноманітних завдань. Наприклад завдання класифікації, регресії або прогнозування часових рядів.

Набір зважених зав’язків кожен вузол попереднього шару з кожним вузлом наступного шару в нейронній мережі. Вузли вхідного шару отримують вхідні дані, і кожен наступний прихований шар не лінійно перетворює дані, застосовуючи функції активації. Це можуть бути функції такі як сигмоїд або ReLU (зрізаний лінійний вузол). Вихідний шар генерує остаточний прогноз моделі, який може бути як вектором значень так і скалярним значенням.

MLP активно використовується в різноманітних сферах. Як приклад за допомогою MLP можна розпізнавати зображення, звук, обробляти природню мову або прогнозувати часові ряди.

MLP виконує серію математичних операцій над вхідними даними для створення результату чи прогнозу. MLP складається з численних шарів вузлів, кожен з яких виконує нелінійну модифікацію вхідних даних. Принцип роботи MLP можна побачити на рисунку 2.1.

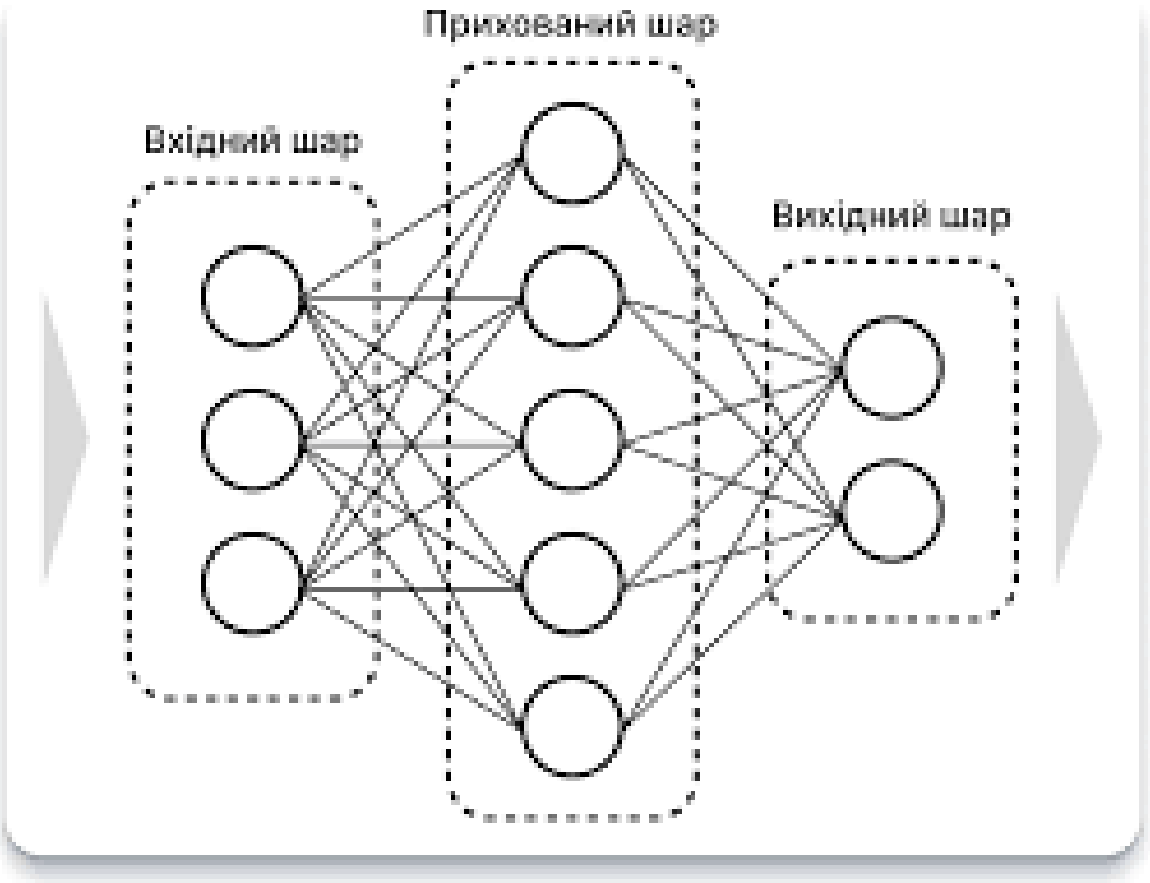



Рис. 2.1 Принцип роботи MLP

MLP складається з наступних складових:

- Вхідний шар. Він складається з одного або декількох вузлів, кожен з яких відповідає певній характеристиці або вхідній змінній у вхідних даних. Вхідні дані подаються у вхідний шар, і кожен вузол обчислює зважену суму вхідних значень;
- Приховані шари. Після вхідного шару дані подаються у прихований шар, де кожен вузол отримує дані від усіх вузлів попереднього шару та обчислює зважену суму, яка потім обробляється за допомогою функції активації для подальшого створення вихідного вузла. Прихованих шарів може бути декілька;
- Вихідний шар. Всі вихідні дані з останнього прихованого шару подаються до вихідного шару, де кожен вузол обчислює зважену суму

входів та пропускає їх через функцію активації та на виході ми отримуємо готовий результат чи прогноз;

- Закріплення Ваги. Часто навчаються за допомогою закріплення, в якому різниця між очікуваним і фактичним виходом передається назад через мережу, а ваги змінюються для мінімізації похибки. Таке навчання часто здійснюється за допомогою стохастичного градієнтного спуску або однієї з його варіацій.

Головна задача MLP – це зрозуміти зв'язок між вхідними даними та вихідними даними на навчальних даних , щоб цей метод міг робити точні прогнози на нових даних, які він ще не бачив. MLP можна навчити представляти складні нелінійні взаємодії між вхідними даними та вихідними змінними, змінюючи ваги мережі.

Математична складова MLP складається з наступних формул:

- Формула виходу першого прихованого шару (2.1) :

$$Z_1 = f(w_1 * x + b1), \quad (2.1)$$

де $x$ – вхідний вектор;

$w$ – вагова матриця;

$b$ – вектор зсуву;

$f$ – функція активації.

- Формула виходу другого прихованого шару (2.2) :

$$Z_2 = f(w_2 * z_1 + b2), \quad (2.2)$$

де $x$ – вхідний вектор;

$w$ – вагова матриця;

$b$ – вектор зсуву;

$f$ – функція активації.

- Формула виходу вихідного шару (2.3) :

$$y = f(w_3 * z_2 + b_3), \quad (2.3)$$

де $x$ – вхідний вектор;

$w$ – вагова матриця;

$b$ – вектор зсуву;

$f$ – функція активації.

Формула MLP передбачає послідовність множення матриць і застосування функцій активації на всіх рівнях мережі, причому кожен рівень забезпечує вихід, який стає входом для наступного рівня, поки вихідний рівень не створить остаточний результат.

До переваг MLP можна віднести:

- Універсальність. MLP відзначається своєю гнучкістю, оскільки може працювати з різними типами вхідних даних, такими як безперервні або категоріальні змінні, і здатні обробляти дані з пропусками;
- Узагальнення. MLP здатні добре узагальнювати нові й раніше невідомі дані за умови належного навчання, що робить їх корисними для практичних застосувань.
- Масштабованість: Моделі можна розширювати, додаючи додаткові приховані шари або вузли, що підвищує їхню ефективність у складних завданнях;
- Нелінійне моделювання. MLP ефективно моделюють складні нелінійні взаємодії між входами та виходами, завдяки чому вони підходять для широкого спектра застосувань.

До недоліків MLP відносять:

- Модель "чорної скриньки". Часто MLP вважають "чорними скриньками", оскільки процес ухвалення рішень або формування прогнозів може бути важко зрозуміти через приховані шари;
- Перенавчання. MLP можуть легко перенавчатися, якщо модель надмірно складна або обсяг навчальних даних обмежений;
- Повільне навчання. Навчання MLP може бути тривалим, особливо у випадках великих наборів даних або глибоких мереж;

- Оптимізація гіперпараметрів. Для досягнення максимальної продуктивності потрібно налаштовувати параметри моделі, такі як кількість вузлів, шари, функції активації та швидкість навчання.

Отже за рахунок своєї гнучкості та універсальності даний метод можливо використовувати для рішення задачі прогнозування вартості продуктової корзини.

## 2.2. **Лінійна регресія**

Лінійна регресія – це один з методів навчання нейронної мережі. Він використовується для моделювання взаємозв'язку між однією або кількома незалежними змінними, їх ще називають предикторами та залежною змінної або цільовою змінної. Основна ідея методу лінійної регресії полягає в пошуку лінійної функції, яка найкраще описує ці дані.

Наглядно приклад роботи лінійної регресії зображено на рисунку (2.2):

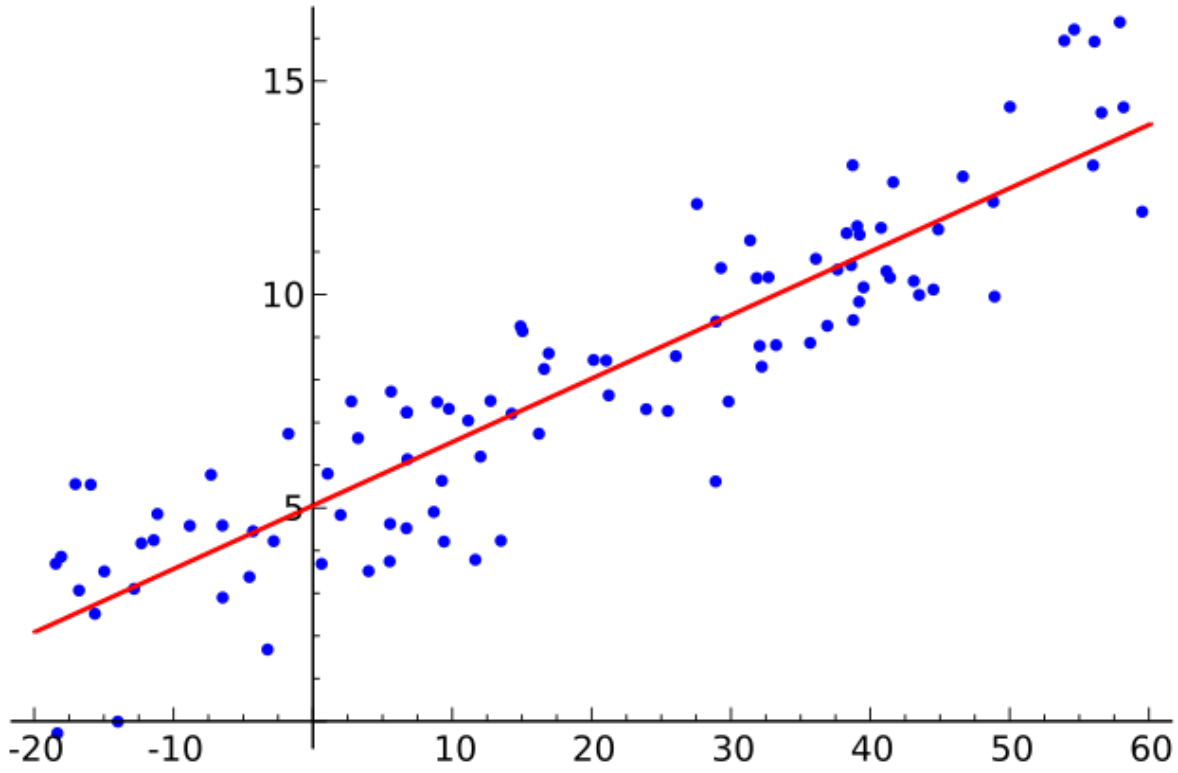



Рис.2.2 Лінійна регресія

Лінійну регресію використовують досить часто. В основному її використовують для рішення завдання в яких необхідно вгадати якесь число, або спрогнозувати продажі товару або вказати приблизну вартість житла за заданими параметрами-критеріями [12].

В загальному вигляді формула лінійної регресії має вигляд об'єкта (2.4) :

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k + \varepsilon , \quad (2.4)$$

де $Y$ – це залежна змінна, яку ми намагаємося передбачити;

$X_1, X_2, \dots, X_k$- це предиктори;

$\beta_0, \beta_1, \beta_2, \dots, \beta_k$ – це коефіцієнти регресії (ваги), які множаться на відповідні предиктори. Ваги предикторів визначають, як кожен предиктор впливає на залежну змінну;

$\varepsilon$ – це розбіжність між передбаченими значеннями й фактичними, нормально розподілена випадкова величина.

У загальному вигляді лінійна модель має вигляд функції (2.5):

$$a(X) = \omega_0 + \sum_{j=1}^{d} \omega_J x_j, \quad (2.5)$$

де $\omega_0$ – це вільний коефіцієнт;

$\omega_J$ – це ваги або коефіцієнти;

$x_j$ – це предиктори.

Перед початком роботи з лінійною регресією необхідно підготувати дані. Для лінійної регресії не рекомендовано брати довільну вибірку даних, через те що результат скоріше всього ми отримаємо не дуже задовільний. Дані необхідно попередньо підготувати для того щоб отримати ефективний та точний результат. Найкраще брати вибірку з різноманітними усередненими даними, де не буде велика кількість вибірки займати велике процентне відношення.

Також в даних не повинно міститись шуму та непотрібної інформації, тому що через них модель може давати неправильний результат.

Наступними кроком при роботі з Лінійною регресією може бути шкалювання.

Шкалювання – це процес приведення предикторів до одного масштабу. Якщо предиктори будуть мати різний масштаб, це вплине на процес навчання та інтерпретацію коефіцієнтів. Для цього використовують методи стандартизації або нормалізації.

Припустимо в нас є набір даних проданих квартир в конкретному регіоні з двома змінними: Перша змінна – це ціна квартири яка варіюється від кількох тисяч, до сотен тисяч доларів, а інша – це кількість проданих квартир, яка не може бути більше кількох тисяч. У даному випадку нам необхідно шкалювати ці змінні, оскільки модель лінійної регресії буде більше приділяти уваги ціні квартири, а не кількості. Це відбудется через те що ціна квартири має більший діапазон значень. Якщо ми наприклад, розділимо ціну квартири на 1000, а кількість проданих автомобілів на 100, то після масштабування матимуть діапазон значень від 0 до 1. Через це модель буде більш стійкою до шумів та аномалій, та буде поліпшено її передбачувальну точність.

Також для підготовки даних можна використовувати бінаризацію даних. Це метод перетворення кількісних ознак на двійкові, на основі деякого порогового значення. У результаті роботи цього методу всі значення які менше за певний поріг ми отримаємо 0, а ті що більше або дорівнює порогу стануть одиницею.

Також якщо ми маємо справу з задачами у яких ми маємо вхідні дані якийсь текст, чи якесь дискретне значення , то нам необхідно представити його в числовому форматі. Для цього використовується метод  Label Encoding за допомогою якого можна кожній унікальній категорії можна присвоїти унікальне число [13].

Також якщо нам необхідно перетворити категоріальні змінні на числові, можна застосовувати метод One-Hot encoding. Принцип його роботи такий: для кожної категоріальної змінної створюється вектор, після чого для кожного компонента вектора присвоюється значення 1, якщо значення категоріальної змінної відповідає цьому компоненту, або 0 якщо не відповідає і фінальним кроком всі вектори об’єднуються в один єдиний набір даних.

Після підготовки даних можна застосовувати алгоритм лінійної регресії який видасть нам результат.

Також важливою частиною після навчання нейронної мережі є оцінення її помилки.

Класичним методом для вимірювання помилки в задачах лінійної регресії є середньоквадратична помилка MSE Mean Squared Error (MSE). SE вимірює середньоквадратичне відхилення між фактичними значеннями та передбачуваними значеннями моделі. Чим більше помилка, тим більше внесок вона унесе в цю метрику.

Також існує модифікований варіант цього методу - середньоквадратична помилка Root Mean Square Error (RMSE). RMSE рахує середнє абсолютне відхилення між фактичними даними та передбаченими значеннями моделі. Вона надає значення яке простіше інтерпретувати.

Переваги методу Лінійної регресії:

- Швидке навчання;
- Результати роботи цього методу легко інтерпретувати;
- Досить стійкі до шуму
- Він досить не вимогливий до обчислювальних ресурсів.

Недоліки методу Лінійної регресії:

- Низька ефективність в задачах з нелінійною залежністю;
- Потребує досить довгої підготовки даних;
- Вразливий до викидів даних.

Отже метод Лінійної регресії не дуже добре підійде для розв’язування завдання прогнозу вартості продуктової корзини, оскільки в цій задача ми маємо роботу по більшій часті з нелінійним даними, через що цей метод буде не ефективним. Але метод оцінки точності можна спробувати використати з іншими методами, що може бути досить ефективно.

## 2.3. **Дерево рішень**

Дерево рішень – це метод навчання нейроних мереж. Зазвичай цей алгоритм використовується для завдань лінійної регресії або класифікації [14]. Цей алгоритм ділить набір даних на дедалі менші частини, доки дані не будуть розділені на окремі результати, які потім будуть класифікуватись. Якщо візуалізувати результати роботи алгоритму, то спосіб розподілу категорій нагадував би дерево та багато листя.

Дерево рішення дуже схоже на блок-схему. Коли ви використовуєте блок-схему, ви починаєте з початкової точки або кореня діаграми, а потім, залежно від того як дані відповідають критеріям фільтрації початкового вузла, а потім переходите до одного з можливих вузлів. Цей процес циклічно повторюється, поки не буде досягнуто фінального результата.

Дерева рішень працюють аналогічно, причому кожен внутрішній вузол у дереві є деяким критерієм перевірки чи фільтрації. Кінцеві точки дерева є мітками для відповідної точки даних, зазвичай їх називають листям. Гілки, які ведуть від внутрішніх вузлів до наступного вузла, є об’єктами або об’єднаннями об’єктів. Правила, які використовуються для класифікації точок даних, — це шляхи, що проходять від кореня до листя.

На рисунку (2.3) зображено приклад дерева рішень.

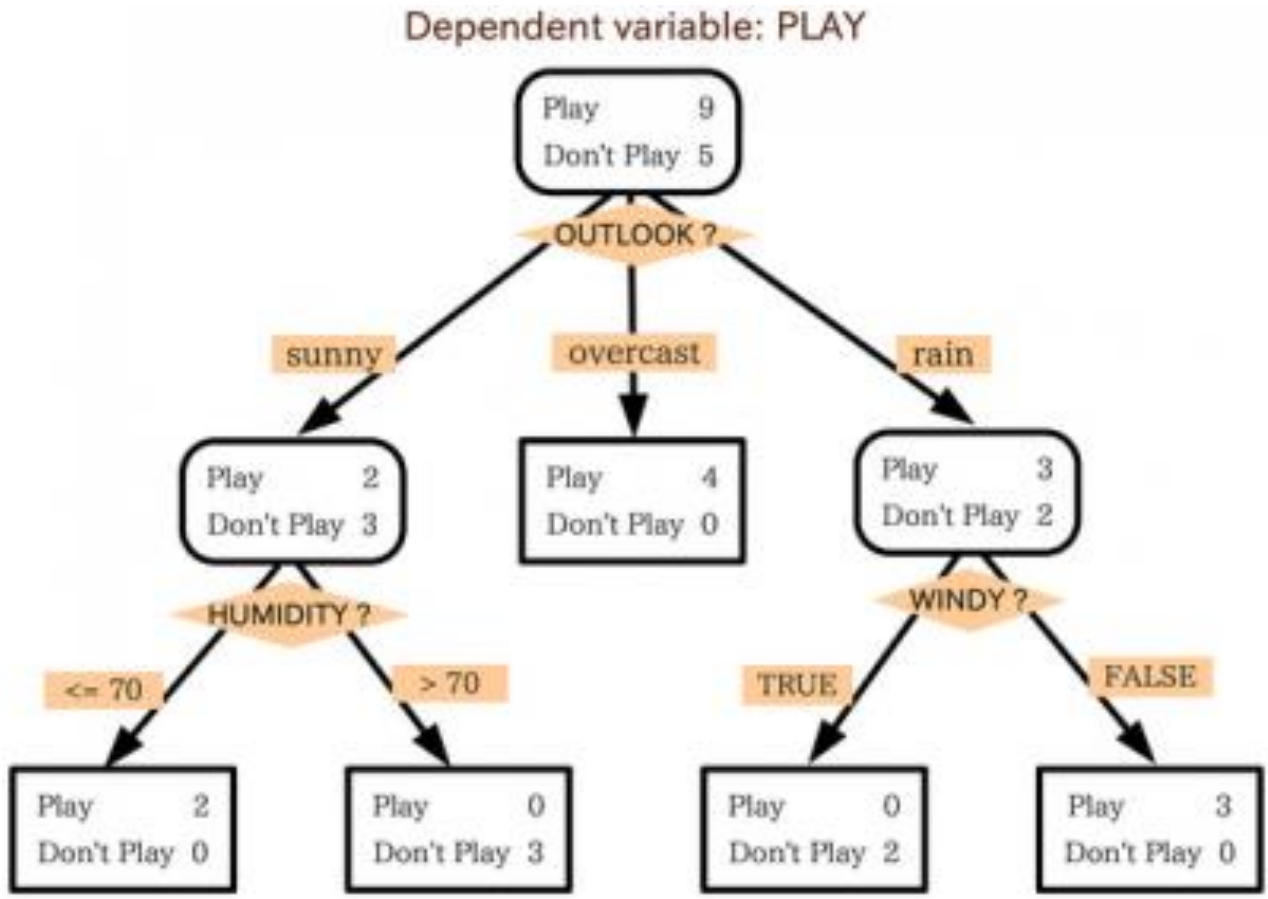



Рис. 2.3 Приклад дерева рішень

Дерева рішень в своїй основі містять алгоритмічний підхід, вони розбивають набір даних на окремі точки даних на основі різних критеріїв. Ці розбиття виконуються за допомогою різних змінних або різних функцій набору даних.

Існує багато методів за допомогою яких можна розбити дерево на гілки та листи. Одним з найпоширеніших способів розбиття є техніка під назвою рекурсивне двійкове розбиття. Процес розподілу під час виконання цього методу починається з кореня, а кількість ознак у наборі даних представляє можливу кількість можливих поділів. Функція використовується для того щоб визначити, скільки точності буде коштувати кожне можливе розбиття та здійснює розбиття за критерієм, який буде мати найбільшу точність. Цей процес відбувається рекурсивно та підгрупи за аналогічною стратегією.

При визначені вартості розколу використовується функція втрат. Для завдань регресії та завдань класифікації буде використовується інша функція вартості.

Метою обох функцій витрат є визначення того, які гілки мають найбільш схожі значення відповіді або найбільш однорідні гілки.

З точки зору регресійної функції вартості для рекурсивного двійкового розбиття, алгоритм, який використовується для обчислення вартості можна описати наступною формулою (2.6):

$$\sum(y - \text{передбачення})^2, \quad (2.6)$$

де $y -$ це реальні значення з навчалбної вибірки.

Прогнозом для групи точок даних є середнє значення цільової змінної навчальної вибірки в цій групі. Усі точки даних аналізуються за допомогою функції вартості для оцінки всіх можливих розподілів, і обирається той розподіл, що забезпечує найменшу вартість.

Для класифікаційних завдань функція вартості визначається за формулою (2.7):

$$G = \sum\big(p_k * (1 - p_k)\big), \quad (2.7)$$

де $p_k$- це частка точок які належать до певного класу.

Це так звана оцінка Джині, яка характеризує ефективність розбиття на основі пропорції точок кожного класу в утворених групах. Вона показує рівень "змішаності" груп: ідеальне розбиття створює групи, в яких усі дані належать до одного класу. У такому випадку $p_k$ дорівнює 0 або 1, а $G = 0.$ Найгірший сценарій — це рівний розподіл класів, наприклад, $50/50$ у разі бінарної класифікації. Тоді $p_k = 0.5$, а $G = 0.5$.

Процес розбиття завершується, коли всі точки даних перетворюються на листові вузли дерева. Проте ріст дерева можна зупинити раніше, щоб уникнути перенавчання. Для цього використовуються різні стратегії, зокрема: можна встановити мінімальну кількість точок, необхідні для утворення листа або обмежити максимальну довжину шляху від кореня до листа.

Ще однією важливою складовою побудови дерев рішень є обрізка (pruning). Цей процес дозволяє зменшити складність дерева, видаляючи гілки з низькою прогностичною здатністю. Це сприяє підвищенню узагальнюваності моделі та зниженню ризику перенавчання.

Обрізка може виконуватись зверху вниз або знизу вгору. Найпоширеніший метод обрізки — скорочення помилок: видаляються листи або вузли, що містять найпоширеніший клас, якщо це не знижує точність моделі. Інші методи обрізки також застосовуються, але описаний підхід є найбільш популярним.

Дерева рішень часто використовуються для задач класифікації завдяки своїй здатності виділяти ключові ознаки з набору даних, які мають найбільшу прогностичну силу. Однією з головних переваг є зрозумілість: на відміну від багатьох інших алгоритмів машинного навчання, їхні правила легко інтерпретувати. Крім того, дерева рішень можуть працювати як з категоричними, так і з безперервними змінними, що зменшує обсяг попередньої обробки даних порівняно з методами, які підтримують лише один тип змінних.

Однак деревам рішень властиві певні недоліки. Вони не завжди ефективні для прогнозування значень безперервних змінних. Також точність дерева рішень може суттєво знизитися у випадках, коли навчальні дані містять мало прикладів, але мають велику кількість класів. Ще одне обмеження — час обчислень, який може бути значним, особливо для великих і складних наборів даних.

До переваг цього методу можна віднести:

- Проста інтерпретація результату;
- Можливість працювати як з числовими даними, так і з категоріальними даними.
- Висока швидкість навчання;
- Не вимагає нормалізації або масштабування.

До недоліків цього методу можна віднести:

- Алгоритм дуже схильний до перенавчання, особливо на великих наборах даних;
- Навіть невеликі зміни у даних можуть суттєво вплинути на структуру дерева;
- Необхідність обрізання.

Для вирішення завдання прогнохування вартості продуктової корзини цей метод не дуже доречно використовувати оскільки ми маємо справу з досить великими наборами даних.

## 2.4. **Рандомні ліси**

Рандомні ліси – це ансамблева модель навчання нейронної мережі, яка об’єднує велику кількість дерев рішень. Цей підхід використовується для задач класифікації та регресії [16].

Ансамблевий метод навчання – це метод, який використовує багато класифікаторів для вирішення складних завдань.

Для того щоб навчити ліс сформований методом випадкового лісу зазвичай використовують мішки або бутстрап- агрегацію. Агрегація – це алгортм, який шляхом групування методів алгоритмів машинного навчання підвищує їх точність.

Алгоритм визначає результат основі прогнозів дерева рішень. Він видає результат методом узагальнення результатів різних дерев. Чим більше буде кількості дерев, тим точніше буде результат.

Деяких недоліків алгоритму дерев рішень можна уникнути використовуючи метод випадкового лісу. Метод рандомного лісу підвищує точність, зменшуючи при цьому надмірне використання набору даних. Він не вимагає великої кількості налаштувань пакета.

Різниця між деревом рішень та алгоритмами рандомних лісів полягає в тому, що рандомні ліси випадковим чином визначають кореневі вузли та відокремлюють вузли. Метод мішків використовується випадковим лісом для генерації необхідного прогнозу.

Цей метод передбачає використовування багатьох вибірок, замість того щоб використовувати одну. Навчальний набір даних - це набір спостережень і атрибутів, які використовуються для прогнозування. В залежності від навчальних даних, які надаються алгоритму випадкового лісу, дерева рішень будуть мати різні

результати. Після цього ці результати пройдуть оцінку, в результаті якої той що отримає найвищу оцінку буде обрано як кінцевий продукт.

Модель рандомних лісів передбачає як значення залежних ознак, так і незалежних змінних.

Регресію випадкових лісів можна в різних системах, будь-то Python, R чи SAS. Кожне дерево в регресії випадкового лісу робить свій унікальний прогноз. Результатом регресії є середній прогноз окремих дерев, в той час коли класифікація випадкового лісу визначає результат на основі класу дерева рішень.

Регресія випадкового лісу та лінійна регресія хоч і досить схожу поняття, але їх цілі. Формула лінійної регресії (2.8):

$$y = bx + c, \tag{2.8}$$

де $y$ – це залежні змінна;

$x$ – незалежна змінна;

$b$ – оцінюваний параметр;

$c$ – константа.

Функція складної регресії є функцією чорного ящику. Це означає що ми не можемо передбачити та інтерпретувати її кроки.

З іншого боку, класифікація у випадкових лісах базується на ансамблевому підході для досягнення необхідного результату. Навчання здійснюється через побудову кількох дерев рішень, які використовують навчальний набір даних. Цей набір складається зі спостережень і характеристик, які вибираються випадковим чином для поділу у вузлах.

У системі рандомних лісів використовуються сукупність дерев рішень. Дерево рішень складається з трьох типів вузлів: кореневих вузлів, внутрішніх вузлів і листових вузлів. Листовий вузол кожного дерева визначає кінцевий результат, отриманий цим деревом рішень. Остаточний результат системи визначається шляхом голосування більшістю: вибирається той варіант, який підтримала більшість дерев. У підсумку, результат випадкового лісу є рішенням, яке має найбільшу підтримку серед дерев рішень.

До переваг цього методу можна віднести:

- Досить висока точність, особливо якщо порівнювати з одним деревом рішень;
- Модель досить стійка до перенавчання, вона досить добре узагальнює дані;
- Можливість моделі працювати з чиловими, категоріальними або змішаними даними;
- Досить ефективна при роботі з даним, в яких є велика кількість характеристик;
- Модель може оцінювати важливість тих чи інших характеристик.

Недоліки ця модель має наступні:

- Ця модель є чорним ящиком, через що неможливо пояснити той чи інший результат;
- Потребує досить викосі обчислювальні витрати;
- Потребує налаштувань кількості дерев, їх максимальної глибини, кількості характеристик та ще деяких переметрів, що ускладнює процес навчання.

## 2.5. **Згорткові нейронні мережі**

Згорткові нейронні мережі – це потужний інструмент машинного навчання, особливо в задачах, пов'язаних з комп'ютерним зором. Згорткові нейронні мережі, або CNN, - це спеціалізований клас нейронних мереж, призначений для ефективної обробки сіткоподібних даних, таких як зображення.

Згорткова нейронна мережа (CNN) - це тип алгоритму глибокого навчання, який особливо добре підходить для задач розпізнавання та обробки зображень. Вона складається з декількох шарів, включаючи згорткові шари, об'єднані шари та повністю з'єднані шари.

Ключові компоненти згорткової нейронної мережі включають в себе наступні:

- Згорткові шари: Ці шари застосовують операції згортки до вхідних зображень, використовуючи фільтри (також відомі як ядра) для виявлення таких особливостей, як краї, текстури та більш складні патерни. Згорткові операції допомагають зберегти просторові відношення між пікселями;
- Об'єднання шарів: Об'єднання шарів зменшує просторові розміри вхідних даних, зменшуючи обчислювальну складність і кількість параметрів у мережі. Максимальне об'єднання - це звичайна операція об'єднання, яка вибирає максимальне значення з групи сусідніх пікселів;
- Функції активації: Нелінійні функції активації, такі як Rectified Linear Unit (ReLU), вносять нелінійність в модель, дозволяючи їй вивчати більш складні взаємозв'язки в даних;
- Повністю з'єднані шари: Ці шари відповідають за прогнозування на основі високорівневих характеристик, вивчених попередніми шарами. Вони з'єднують кожен нейрон одного шару з кожним нейроном наступного шару.

CNN навчаються на великому наборі даних мічених зображень, де мережа вчиться розпізнавати патерни та ознаки, які асоціюються з певними об'єктами або класами. Довели свою високу ефективність у завданнях, пов'язаних із зображеннями, досягаючи найсучаснішої продуктивності в різних програмах комп'ютерного зору. Їх здатність автоматично вивчати ієрархічні представлення ознак робить їх добре придатними для завдань, де просторові зв'язки і закономірності в даних мають вирішальне значення для точних прогнозів. CNN широко використовуються в таких областях, як класифікація зображень, виявлення об'єктів, розпізнавання облич і аналіз медичних зображень.

Згорткові шари є ключовим компонентом CNN, де до вхідного зображення застосовуються фільтри для виокремлення таких елементів, як краї, текстури та форми.

Вихідні дані згорткових шарів проходять через об'єднувальні шари, які використовуються для зменшення вибірки карт об'єктів, зменшуючи просторові розміри, зберігаючи при цьому найважливішу інформацію. Вихідні дані об'єднаних шарів пропускаються через один або кілька повністю з'єднаних шарів, які використовуються для прогнозування або класифікації зображення.

Побудова згорткової нейронної мережі

- Згорткова нейронна мережа - це багатошарова нейронна мережа прямого поширення, створена шляхом складання багатьох невидимих шарів один над одним у певному порядку;
- Саме послідовна конструкція дозволяє CNN вивчати ієрархічні атрибути;
- У CNN деякі з них супроводжуються шарами групування, а приховані шари, як правило, є шарами згортки, за якими слідують шари активації;
- Попередня обробка, необхідна в ConvNet, споріднена з відповідною структурою нейронів у людському мозку і була мотивована організацією зорової кори.

CNN навчаються з використанням підходу керованого навчання. Це означає, що CNN отримують набір навчальних зображень з мітками. Потім CNN навчається зіставляти вхідні зображення з правильними мітками.

Процес навчання CNN складається з наступних кроків:

- Підготовка даних: Навчальні зображення попередньо обробляються, щоб переконатися, що всі вони мають однаковий формат і розмір;
- Функція втрат: Функція втрат використовується для вимірювання того, наскільки добре CNN працює на навчальних даних. Функція втрат зазвичай обчислюється як різниця між передбаченими мітками та фактичними мітками навчальних зображень;

- Оптимізатор: Оптимізатор використовується для оновлення ваг CNN з метою мінімізації функції втрат;
- Зворотне розповсюдження: Зворотне поширення - це метод, який використовується для обчислення градієнтів функції втрат відносно ваг CNN. Градієнти потім використовуються для оновлення ваг CNN за допомогою оптимізатора.

Після навчання CNN можна оцінити на тестовому наборі, який зберігається. Набір зображень, які CNN не бачив під час навчання, складає тестовий набір. Те, наскільки добре CNN працює на тестовому наборі, є гарним предиктором того, наскільки добре він працюватиме на реальних даних.

Ефективність CNN на завданнях категоризації зображень можна оцінити за різними критеріями. Серед найпопулярніших метрик є такі:

- Точність: Точність - це відсоток тестових зображень, які CNN прогнозує як певний клас і які дійсно належать до цього класу.
- Відтворення: Впізнаваність - це відсоток тестових зображень, які належать до певного класу і які CNN прогнозує як зображення цього класу.
- Оцінка F1: Показник F1 - це середнє гармонійне значення точності та впізнаваності. Це хороша метрика для оцінки продуктивності CNN на класах, які є незбалансованими.

Існують наступні типи моделей CNN:

- LeNet
- AlexNet
- ResNet
- GoogleNet
- MobileNet
- VGG

### 2.5.1. LeNet

LeNet - це новаторська архітектура згорткової нейронної мережі (CNN), розроблена Яном Лекуном та його колегами наприкінці 1990-х років [18]. Вона була спеціально розроблена для розпізнавання рукописних цифр і стала однією з перших успішних CNN для розпізнавання зображень.

LeNet складається з декількох шарів згорткових та об'єднаних шарів, за якими слідують повністю з'єднані шари. Архітектура включає два набори шарів згортки та об'єднання, за кожним з яких слідує шар субдискретизації, а потім три повністю з'єднані шари.

Перший згортковий шар використовує ядро розміром 5×5 і застосовує 6 фільтрів до вхідного зображення. Вихід цього шару потім пропускається через шар об'єднання, який зменшує просторові розміри карт об'єктів. Другий згортковий шар використовує ядро розміром 5×5 і застосовує 16 фільтрів до виходу першого об'єднуючого шару. За ним слідує ще один об'єднувальний шар і шар субдискретизації.

Вихід шару субдискретизації пропускається через три повністю з'єднані шари з 120, 84 та 10 нейронами відповідно. Останній повністю з'єднаний шар використовується для класифікації і виробляє розподіл ймовірностей по 10 цифрам (0-9).

LeNet був навчений на наборі даних MNIST, який складається з 70 000 зображень рукописних цифр, і зміг досягти високої точності розпізнавання. Архітектура LeNet, хоч і відносно проста порівняно з сучасними архітектурами, послужила основою для багатьох пізніших CNN, і її вважають класичною і простою архітектурою для задач розпізнавання зображень.

### 2.5.2. AlexNet

AlexNet - це архітектура згорткової нейронної мережі (CNN), яку розробили Алекс Крижевський, Ілля Суцкевер та Джеффрі Хінтон у 2012 році [19]. Це була перша CNN, яка перемогла в ImageNet Large Scale Visual Recognition Challenge (ILSVRC), великому конкурсі з розпізнавання зображень, і це допомогло зарекомендувати CNN як потужний інструмент для розпізнавання зображень.

AlexNet складається з декількох шарів згорткових та об'єднаних шарів, за якими слідують повністю з'єднані шари. Архітектура включає п'ять згорткових шарів, три об'єднувальні шари і три повністю з'єднані шари.

Перші два згорткові шари використовують ядро розміром 11×11 і застосовують 96 фільтрів до вхідного зображення. Третій і четвертий згорткові шари використовують ядро розміром 5×5 і застосовують 256 фільтрів. П'ятий згортковий шар використовує ядро розміром 3×3 і застосовує 384 фільтри. Вихідні дані цих згорточних шарів потім пропускаються через шари максимального об'єднання, які зменшують просторові розміри карт об'єктів.

Вихідні дані об'єднаних шарів пропускаються через три повністю зв'язані шари з 4096, 4096 і 1000 нейронів відповідно. Останній повністю з'єднаний шар використовується для класифікації і виробляє розподіл ймовірностей для 1000 класів ImageNet.

AlexNet був навчений на наборі даних ImageNet, який складається з 1,2 мільйона зображень з 1000 класів, і зміг досягти високої точності розпізнавання. Архітектура AlexNet вперше показала, що CNN можуть значно перевершити традиційні методи машинного навчання в задачах розпізнавання зображень, і стала важливим кроком у розвитку більш глибоких архітектур, таких як VGGNet, GoogleNet і ResNet.

### 2.5.3. Resnet

ResNets (залишкові мережі) - це тип алгоритму глибокого навчання, який особливо добре підходить для задач розпізнавання та обробки зображень [20]. ResNets відомі своєю здатністю навчати дуже глибокі мережі без перенавчання.

ResNets часто використовують для задач виявлення ключових точок. Виявлення ключових точок - це завдання визначення місцезнаходження певних точок на об'єкті на зображенні. Наприклад, виявлення ключових точок можна використовувати для визначення очей, носа і рота на людському обличчі.

ResNets добре підходять для задач виявлення ключових точок, тому що вони можуть навчитися витягувати ознаки із зображень різного масштабу.

ResNets досягли найкращих результатів у багатьох тестах на виявлення ключових точок, таких як COCO Keypoint Detection Challenge та MPII Human Pose Estimation Dataset.

### 2.5.4. GoogleNet

GoogleNet, також відомий як InceptionNet, - це тип алгоритму глибокого навчання, який особливо добре підходить для задач розпізнавання та обробки зображень [21]. GoogleNet відомий своєю здатністю досягати високої точності в задачах класифікації зображень, використовуючи при цьому менше параметрів і обчислювальних ресурсів, ніж інші сучасні CNN.

Початкові модулі є ключовим компонентом GoogleNet. Вони дозволяють мережі одночасно вивчати ознаки на різних масштабах, що покращує продуктивність мережі в задачах класифікації зображень.

GoogleNet використовує глобальне усереднене об'єднання для зменшення розміру карт об'єктів, перш ніж вони будуть передані повністю підключеним шарам. Це також допомагає покращити продуктивність мережі в задачах класифікації зображень.

GoogleNet використовує факторизовані згортки для зменшення кількості параметрів та обчислювальних ресурсів, необхідних для навчання мережі.

GoogleNet є потужним інструментом для класифікації зображень і використовується в широкому спектрі додатків, наприклад, GoogleNet можна використовувати для класифікації зображень на різні категорії, такі як коти і собаки, легкові і вантажні автомобілі, квіти і тварини.

### 2.5.5. MobileNet

MobileNets - це тип CNN, які особливо добре підходять для задач розпізнавання та обробки зображень на мобільних та вбудованих пристроях [22].

MobileNets відомі своєю здатністю досягати високої точності в задачах класифікації зображень, використовуючи при цьому менше параметрів і обчислювальних ресурсів, ніж інші сучасні CNN.

MobileNets також використовуються для задач виявлення ключових точок.

MobileNets досягли найкращих результатів у багатьох тестах на виявлення ключових точок.

### 2.5.6. VGG

VGG - це тип згорткової нейронної мережі (CNN), яка відома своєю простотою та ефективністю [23]. CNN зазвичай складаються з низки згорткових та об'єднувальних шарів, за якими слідують кілька повністю з'єднаних шарів.

VGG можуть використовуватися самокерованими автомобілями для виявлення і класифікації об'єктів на дорозі, таких як інші транспортні засоби, пішоходи і дорожні знаки. Ця інформація може бути використана для безпечної навігації автомобіля.

VGG - це потужний та універсальний інструмент для задач розпізнавання зображень.

Застосування CNN

Класифікація зображень: CNN - це найсучасніші моделі для класифікації зображень. Їх можна використовувати для класифікації зображень за різними категоріями, наприклад, коти і собаки, легкові та вантажні автомобілі, квіти і тварини.

Виявлення об'єктів: CNN можна використовувати для виявлення об'єктів на зображеннях, таких як люди, автомобілі та будівлі. Їх також можна використовувати для локалізації об'єктів на зображеннях, тобто для визначення місця розташування об'єкта на зображенні.

Сегментація зображень: CNN можна використовувати для сегментації зображень, що означає, що вони можуть ідентифікувати та позначати різні об'єкти на зображенні. Це корисно для таких застосувань, як медична візуалізація та робототехніка.

Аналіз відео: CNN можна використовувати для аналізу відео, наприклад, для відстеження об'єктів у відео або виявлення подій у відео. Це корисно для таких застосувань, як відеоспостереження та моніторинг дорожнього руху.

Переваги CNN:

- CNN можуть досягти найсучаснішої точності в різноманітних задачах розпізнавання зображень, таких як класифікація зображень, виявлення об'єктів та сегментація зображень;
- CNN можуть бути дуже ефективними, особливо якщо вони реалізовані на спеціалізованому обладнанні, такому як графічні процесори;
- CNN відносно стійкі до шуму та варіацій вхідних даних;
- CNN можуть бути адаптовані до різноманітних завдань шляхом простої зміни архітектури мережі.

Недоліки CNN:

- CNN можуть бути складними і важкими для навчання, особливо для великих наборів даних;
- Для навчання та розгортання CNN може знадобитися багато обчислювальних ресурсів;
- Для навчання CNN потрібна велика кількість мічених даних;
- CNN можуть бути складними для інтерпретації, що ускладнює розуміння того, чому вони роблять ті чи інші прогнози.

## 2.6. Рекурентна нейронна мережа

Рекурентна нейронна мережа (RNN) - це глибока нейронна мережа, яка навчається на послідовних даних або часових рядах для створення моделі машинного навчання (ML), яка може робити послідовні прогнози або висновки на основі послідовних вхідних даних [24].

RNN можна використовувати для прогнозування щоденних рівнів паводків на основі минулих щоденних паводків, припливів і метеорологічних даних. Але RNN також можна використовувати для вирішення порядкових або часових завдань, таких як переклад мови, обробка природної мови (NLP), аналіз настроїв, розпізнавання мови і підписи до зображень.

Як працюють RNN

Як і традиційні нейронні мережі, такі як нейронні мережі прямого поширення та згорткові нейронні мережі (CNN), рекурентні нейронні мережі використовують навчальні дані для навчання. Вони відрізняються своєю «пам'яттю», оскільки беруть інформацію з попередніх входів, щоб впливати на поточні вхідні та вихідні дані.

У той час як традиційні мережі глибокого навчання припускають, що входи і виходи є незалежними один від одного, вихід рекурентних нейронних мереж залежить від попередніх елементів у послідовності. Хоча майбутні події також можуть бути корисними при визначенні виходу певної послідовності, односпрямовані рекурентні нейронні мережі не можуть врахувати ці події у своїх прогнозах.

Візьмемо таку ідіому, як «погане самопочуття», яка зазвичай використовується, коли хтось хворіє, щоб пояснити роботу RNN. Щоб ідіома мала сенс, вона повинна бути виражена в цьому конкретному порядку. Як наслідок, рекурентні мережі повинні враховувати позицію кожного слова в ідіомі, і вони використовують цю інформацію, щоб передбачити наступне слово в послідовності.

Кожне слово у фразі «погане самопочуття» є частиною послідовності, де порядок має значення. RNN відстежує контекст, зберігаючи прихований стан на кожному часовому кроці. Цикл зворотного зв'язку створюється шляхом передачі прихованого стану від одного часового кроку до іншого. Прихований стан діє як пам'ять, що зберігає інформацію про попередні вхідні дані. На кожному часовому кроці RNN обробляє поточний вхід (наприклад, слово в реченні) разом з прихованим станом з попереднього часового кроку. Це дозволяє RNN «пам'ятати» попередні точки даних і використовувати цю інформацію для впливу на поточний вихід.

Ще однією відмінною характеристикою рекурентних мереж є те, що вони мають спільні параметри в кожному шарі мережі. У той час як мережі прямого поширення мають різні ваги в кожному вузлі, рекурентні нейронні мережі мають однакові вагові параметри в кожному шарі мережі. Однак ці ваги все одно коригуються за допомогою процесів зворотного поширення і градієнтного спуску, щоб полегшити навчання з підкріпленням.

Рекурентні нейронні мережі використовують алгоритми прямого поширення і зворотного поширення в часі (BPTT) для визначення градієнтів (або похідних), які дещо відрізняються від традиційного зворотного поширення, оскільки вони специфічні для даних послідовності [30]. Принципи BPTT такі ж, як і в традиційному зворотному поширенні, де модель навчається сама, обчислюючи помилки від вихідного шару до вхідного. Ці розрахунки дозволяють нам налаштувати і підібрати параметри моделі належним чином. BPTT відрізняється від традиційного підходу тим, що BPTT підсумовує помилки на кожному часовому кроці, тоді як мережі прямого поширення не потребують підсумовування помилок, оскільки вони не мають спільних параметрів для кожного шару. Порівняння прямих та рекурентних нейронних мереж наведено на рисунку (2.4).

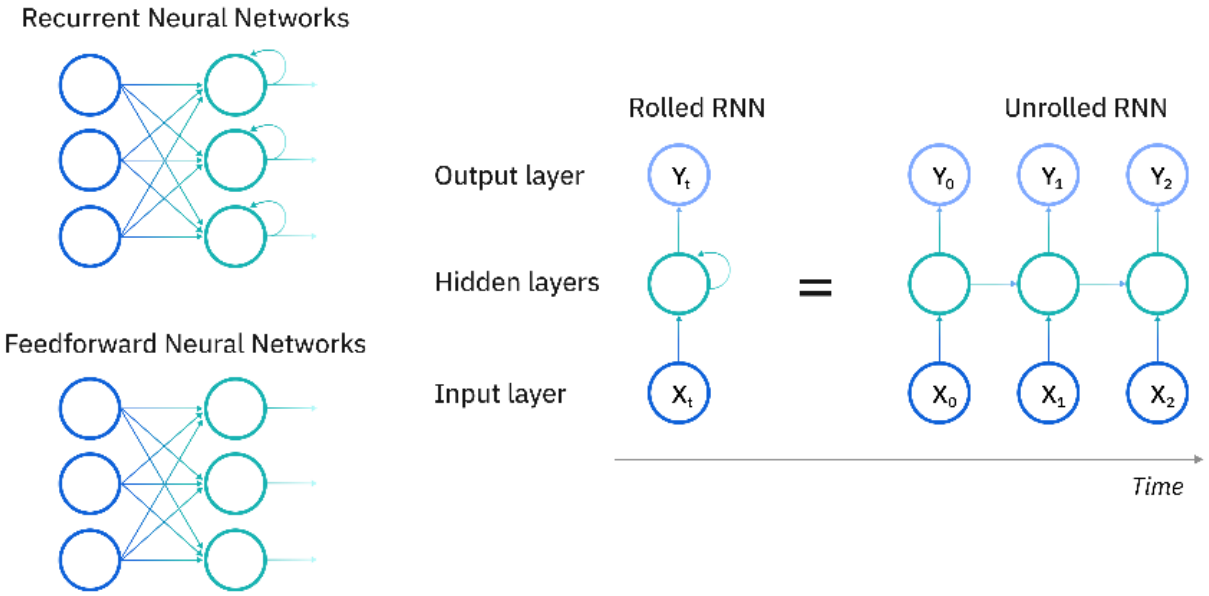



Рис.2.4 Порівняння прямих та рекурентних нейронних мереж

Загальні функції активації

Функція активації - це математична функція, яка застосовується до виходу кожного шару нейронів у мережі, щоб ввести нелінійність і дозволити мережі вивчати більш складні закономірності в даних. Без функцій активації RNN просто обчислював би лінійні перетворення вхідних даних, що робило б його нездатним вирішувати нелінійні задачі. Нелінійність має вирішальне значення для навчання і моделювання складних закономірностей, особливо в таких завданнях, як НЛП, аналіз часових рядів і послідовне прогнозування даних.

Функція активації контролює величину виходу нейрона, утримуючи значення в заданому діапазоні (наприклад, між 0 і 1 або -1 і 1), що допомагає запобігти занадто великому або занадто малому зростанню значень під час прямих і зворотних проходів. У RNN функції активації застосовуються на кожному часовому кроці до прихованих станів, контролюючи, як мережа оновлює свою внутрішню пам'ять (прихований стан) на основі поточних вхідних даних і минулих прихованих станів.

До найпоширеніших функцій активації зображені на рисунку (2.5) нижче відносяться:

Сигмоїдна функція інтерпретує вихідні дані як ймовірності або керує воротами, які вирішують, скільки інформації зберігати або забувати. Однак сигмоїдна функція схильна до проблеми зникаючого градієнта (пояснюється далі), що робить її менш ідеальною для більш глибоких мереж.

Функція Tanh (гіперболічний тангенс), яку часто використовують, оскільки вона виводить значення, зосереджені навколо нуля, що допомагає краще відстежувати градієнтний потік і легше вивчати довгострокові залежності.

ReLU (випрямлена лінійна одиниця) може викликати проблеми з вибуховими градієнтами через свою необмежену природу. Однак, такі варіанти, як Leaky ReLU та Parametric ReLU були використані для пом'якшення деяких з цих проблем.

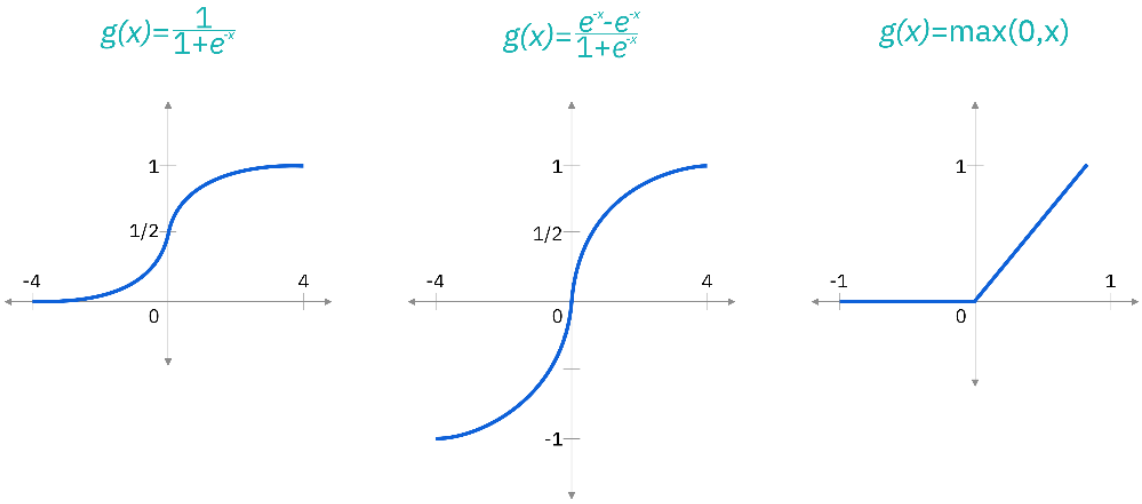



Рис.2.5 Сигмоїдальна функція, Tanh та ReLU

Типи RNN

Мережі прямого поширення відображають входи і виходи один до одного, і хоча ми візуалізували рекурентні нейронні мережі таким чином на діаграмах до цього, вони не мають цього обмеження. Натомість їхні входи та виходи можуть

мати різну довжину, і різні типи RNN використовуються для різних випадків використання, таких як створення музики, класифікація настроїв та машинний переклад. Популярні варіанти архітектури рекурентних нейронних мереж включають:

- Стандартні RNN
- Двонаправлені рекурентні нейронні мережі (BRRN)
- Довга короткочасна пам'ять (LSTM)
- Закриті рекурентні блоки (GNU)
- Кодер-декодер RNN

Популярні варіанти архітектури рекурентних нейронних мереж зображені на рисунку (2.6).

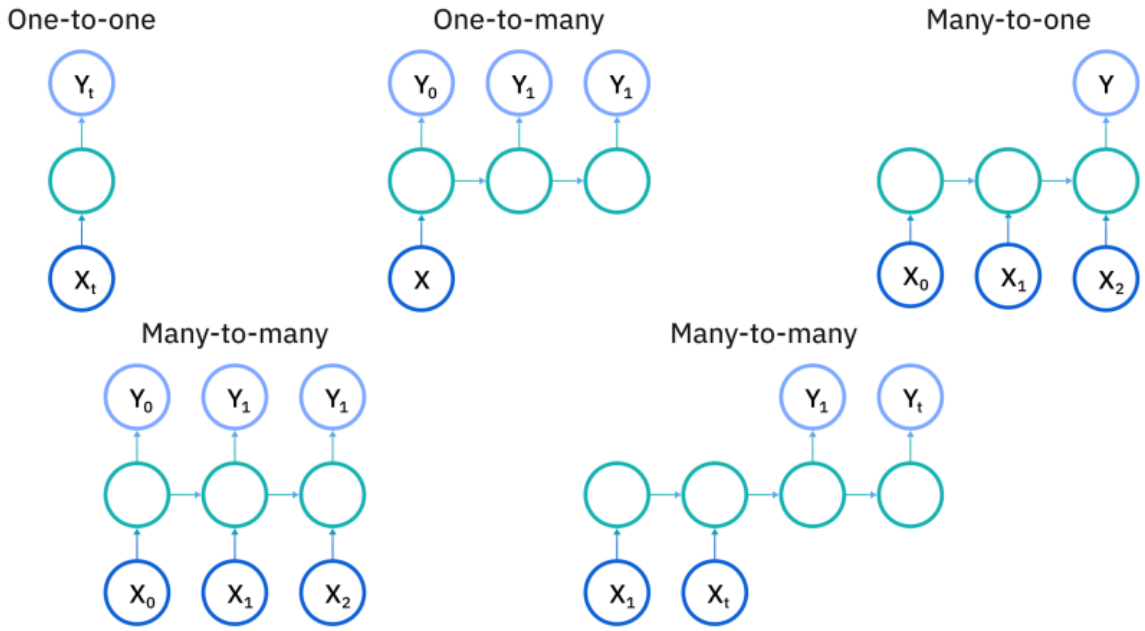



Рис.2.6 Різні типи RNN

Стандартні RNN

Найпростіша версія RNN, де вихід на кожному часовому кроці залежить як від поточного входу, так і від прихованого стану на попередньому часовому кроці, страждає від таких проблем, як зникаючі градієнти, що ускладнює для них вивчення довгострокових залежностей. Вони відмінно справляються з простими завданнями з короткостроковими залежностями, такими як передбачення наступного слова в реченні (для коротких, простих речень) або наступного значення в простому часовому ряді.

RNN добре підходять для завдань, які обробляють дані послідовно в реальному часі, наприклад, обробка даних з датчиків для виявлення аномалій в короткі проміжки часу, коли вхідні дані надходять по одному, а передбачення потрібно робити негайно на основі останніх вхідних даних.

Двонаправлені рекурентні нейронні мережі (BRNN)

У той час як односпрямовані RNN можуть робити прогнози про поточний стан лише на основі попередніх даних, двоспрямовані RNN, або BRNN, залучають майбутні дані, щоб підвищити точність прогнозу. Повертаючись до прикладу з «поганою погодою», модель на основі BRNN може краще передбачити, що другим словом у цій фразі є «під», якщо вона знає, що останнім словом у послідовності є «погода».

Довга короткочасна пам'ять (LSTM)

LSTM - це популярна архітектура RNN, яку представили Сепп Хохрайтер і Юрген Шмідхубер як рішення проблеми зникаючого градієнта. У цій роботі розглядалася проблема довготривалих залежностей. Тобто, якщо попередній стан, який впливає на поточний прогноз, не був у недалекому минулому, RNN-модель може бути не в змозі точно передбачити поточний стан.

Для прикладу, скажімо, ми хочемо передбачити слова, виділені курсивом: «У Аліси алергія на горіхи. Вона не може їсти арахісове масло». Контекст алергії на

горіхи може допомогти нам передбачити, що їжа, яку не можна їсти, містить горіхи. Однак, якби цей контекст був кількома реченнями раніше, то це ускладнило б або навіть унеможливило б зв'язок між інформацією для RNN.

Щоб виправити це, LSTM-мережі мають «комірки» у прихованих шарах штучної нейронної мережі, які мають 3 ворота: вхідні ворота, вихідні ворота та ворота забування. Ці ворота контролюють потік інформації, яка необхідна для прогнозування вихідних даних у мережі. Наприклад, якщо гендерні займенники, такі як «вона», неодноразово повторювалися в попередніх реченнях, ви можете виключити їх зі стану комірки.

Закриті рекурентні блоки (Gated recurrent units, GRU)

GRU схожий на LSTM, оскільки він також працює для вирішення проблеми короткочасної пам'яті моделей RNN. Замість того, щоб використовувати «стан комірки» для регулювання інформації, він використовує приховані стани, і замість 3 вентилів, він має 2: вентиль скидання і вентиль оновлення. Подібно до вентилів у LSTM, вентилі скидання та оновлення контролюють, скільки і якої інформації зберігати.

Завдяки своїй простішій архітектурі GRU є обчислювально ефективнішими і потребують менше параметрів порівняно з LSTM. Це робить їх швидшими в навчанні і часто більш придатними для певних додатків, що працюють в реальному часі або з обмеженими ресурсами.

RNN кодер-декодер

RNN зазвичай використовуються для послідовних завдань, таких як машинний переклад. Кодер перетворює вхідну послідовність у вектор фіксованої довжини (контекст), а декодер використовує цей контекст для генерації вихідної послідовності. Однак вектор контексту фіксованої довжини може бути вузьким місцем, особливо для довгих вхідних послідовностей.

Обмеження RNN

Використання RNN у штучному інтелекті зменшилося, особливо на користь таких архітектур, як трансформантні моделі, але RNN не є застарілими. RNN традиційно були популярні для послідовної обробки даних (наприклад, часових рядів і мовного моделювання) через їхню здатність обробляти часові залежності.

Однак слабкість RNN до проблем зникаючих і вибухаючих градієнтів, а також поява трансформантних моделей, таких як BERT і GPT, призвели до цього занепаду. Трансформатори можуть набагато ефективніше вловлювати довгострокові залежності, їх легше розпаралелювати і вони краще справляються з такими завданнями, як НЛП, розпізнавання мови і прогнозування часових рядів.

Тим не менш, RNN все ще використовуються в специфічних контекстах, де їх послідовна природа і механізм пам'яті можуть бути корисними, особливо в невеликих середовищах з обмеженими ресурсами або для задач, де обробка даних виграє від покрокової повторюваності.

# 3 РОЗРОБКА МЕТОДУ ДЛЯ ПРОГНОЗУВАННЯ ВАРТОСТІ ПРОДУКТОВОГО КОШИКУ ЗА ДОПОМОГОЮ НЕЙРОННИХ МЕРЕЖ

## 3.1. Опис використаних програмних засобів

### 3.1.1. Python

Основною мовою програмування для розробки даного методу було обрано мову Python.

Python — це потужна мова програмування, яка поєднує в собі простоту, гнучкість і широкий спектр можливостей, що робить її однією з найпопулярніших у світі програмістів. Ось деякі деталі, які пояснюють, чому Python так широко використовується.

Однією з основних переваг Python є його інтуїтивно зрозумілий синтаксис. Мова розроблена таким чином, щоб код був максимально простим для читання та написання. Замість використання складних структур або ключових слів, Python використовує відступи для організації блоків коду, що дозволяє уникнути переповненості синтаксичних елементів. Наприклад, вирази й операції виглядають дуже наближеними до звичайної англійської мови, що значно знижує поріг для початківців.

Крім того, Python є інтерпретованою мовою програмування, що означає, що код виконується безпосередньо без необхідності попередньої компіляції. Це дає значну перевагу в швидкості розробки, оскільки програміст може одразу виконувати код, тестувати його й вносити зміни, не чекаючи компіляції, як це необхідно в компільованих мовах (C++, Java). Інтерпретованість також спрощує процес налагодження, оскільки помилки можна виявляти на етапі виконання.

Ще одна потужна характеристика Python — це його здатність працювати на різних операційних системах. Незалежно від того, чи працює програма на Windows, macOS чи Linux, код Python буде працювати однаково, що забезпечує гнучкість у розробці та застосуванні програм. Це робить Python ідеальним вибором для

створення програм, які мають працювати на різних платформах без необхідності адаптації коду для кожної з них.

Python використовує динамічну типізацію, що означає, що тип змінних визначається автоматично під час виконання програми. Це робить код більш гнучким і зручним у використанні, оскільки програмісту не потрібно вказувати тип змінних заздалегідь, що економить час і зменшує кількість коду. Проте це може бути як перевагою, так і недоліком. Відсутність статичної типізації може ускладнити відстеження помилок, особливо у великих проектах.

Python славиться своєю великою стандартною бібліотекою, яка надає готові рішення для численних завдань — від роботи з файлами, операціями з рядками до складних обчислень та веб-запитів. Наприклад, бібліотеки os та sys дозволяють взаємодіяти з операційною системою, а math забезпечує доступ до математичних функцій. Крім того, існують потужні бібліотеки для роботи з веб-програмами, аналізу даних, візуалізації та машинного навчання, такі як Django, Flask, Pandas, NumPy, Matplotlib і TensorFlow.

Ці бібліотеки дозволяють зекономити час і сили, оскільки вже готові рішення для різних завдань, таких як робота з великими даними, візуалізація, обчислення чи розробка веб-додатків. Наприклад, NumPy та Pandas стали стандартом для роботи з великими наборами даних, а Matplotlib дозволяє створювати високоякісні графіки та діаграми. Водночас Python активно підтримує створення глибоких нейронних мереж і машинного навчання завдяки таким бібліотекам, як PyTorch і TensorFlow.

Python є універсальною мовою програмування, яка може бути використана в широкому спектрі застосувань. У веб-розробці популярні фреймворки Django та Flask, які дозволяють швидко створювати потужні та ефективні веб-додатки. Для наукових обчислень Python використовують у таких галузях, як астрономія, фізика, статистика та інженерія. Завдяки своїй здатності до роботи з великими даними, Python став основною мовою в сфері науки про дані та машинного навчання.

Також Python часто використовується для автоматизації рутинних завдань, таких як обробка файлів, автоматичне тестування та розробка скриптів для

адміністрування. У поєднанні з бібліотеками для роботи з веб-протоколами, такими як Requests і BeautifulSoup, Python є популярним вибором для створення скриптів для парсингу веб-сайтів і збору інформації з Інтернету.

Для зручної роботи з Python існує безліч середовищ розробки та інструментів. Найбільш популярними є PyCharm та VS Code — це інтегровані середовища розробки (IDE), які мають всі необхідні інструменти для програмування, від дебагерів до вбудованих терміналів. Jupyter Notebook став стандартом для роботи з науковими обчисленнями, надаючи інтерактивне середовище, яке поєднує код, графіки та текстові пояснення в одному документі.

Для управління залежностями Python використовує pip, який дозволяє встановлювати та оновлювати бібліотеки з офіційного репозиторію. Однак для більш складних проєктів, особливо тих, які мають багато залежностей, зручно використовувати conda, яка забезпечує ефективніше керування середовищами та бібліотеками.

До основних переваг мови Python можна віднести:

- Легкість у навчанні;
- Величезна кількість бібліотек;
- Кросплатформеність;
- Можливість використовувати Python для широкого спектра завдань.

Проте існують і недоліки у цієї мови програмування:

- Велика витрата пам'яті;
- Повільна швидкість виконання;

Python продовжує залишатися однією з найпопулярніших мов програмування завдяки своїй простоті, гнучкості, потужним бібліотекам та активній спільноті. Він підходить як для початківців, так і для досвідчених розробників, і застосовується у найрізноманітніших галузях: від веб-розробки та автоматизації до науки про дані та штучного інтелекту.

### 3.1.2. Jupyter lab

Jupyter Lab — це інтегроване середовище розробки для створення, виконання та обміну кодом, текстами, математичними виразами, графіками та іншими мультимедійними елементами [25]. Jupyter Lab є розширеним варіантом класичного Jupyter Notebook і надає значно більше можливостей для роботи з кодом і даними. Він має зручний інтерфейс для роботи з інтерактивними документами, підтримує різні мови програмування, такі як Python, R, Julia, і дозволяє працювати з різними типами вмісту в одному документі.

Jupyter Lab має інтерфейс з кількома панелями та вікнами, що дозволяє працювати з різними типами вмісту в одному середовищі. Ви можете мати відкриті кілька ноутбуків, терміналів, текстових редакторів, а також інтерактивно переглядати графіки та дані.

Інтерфейс дуже зручний для багатозадачності, оскільки ви можете легко переміщувати панелі та комбінувати їх за потребою. Це особливо корисно, коли працюєте з великими проєктами, де одночасно потрібен доступ до коду, результатів обчислень і графіків.

Хоча Jupyter Lab найбільше використовують для роботи з Python, він підтримує багато інших мов програмування через так звані "ядер" (kernels). Для Python є ядро IPython, але також доступні ядра для таких мов, як R, Julia, Scala, Ruby, Bash та інші. Це дає змогу працювати з різними мовами в одному середовищі.

Jupyter Lab має модульну архітектуру, що дозволяє додавати розширення та плагіни для підтримки нових функціональностей. Це можуть бути інструменти для інтеграції з базами даних, додаткові можливості для візуалізації або навіть інтеграція з системами для обробки великих даних.

Jupyter Lab є ідеальним середовищем для роботи з даними. Підтримка таких бібліотек як Matplotlib, Plotly, Seaborn дозволяє створювати графіки й інтерактивні візуалізації безпосередньо в середовищі Jupyter. Крім того, ви можете переглядати та обробляти таблиці з даними за допомогою Pandas або інтегрувати дані з Excel чи CSV файлів.

Jupyter Lab дозволяє інтерактивно виконувати код і бачити результат прямо в документі. Це зручно для навчання, наукових досліджень і аналізу даних, оскільки можна поетапно виконувати код, відслідковувати його результати та коригувати помилки.

### 3.1.3. Jupyter Notebook

Jupyter Notebook — це один із найпоширеніших інструментів для виконання наукових обчислень, візуалізації даних та створення інтерактивних документів. Він є основним компонентом Jupyter Lab, хоча може використовуватися і окремо.

Jupyter Notebook — це веб-додаток, який дозволяє створювати й обмінюватися документами, що містять як живий код, так і текстові пояснення, математичні рівняння, графіки, таблиці та багато іншого. Цей інструмент широко використовується в аналізі даних, науці про дані, машинному навчанні, освіті та багатьох інших галузях.

Робоче вікно зображено на рисунку (3.1):

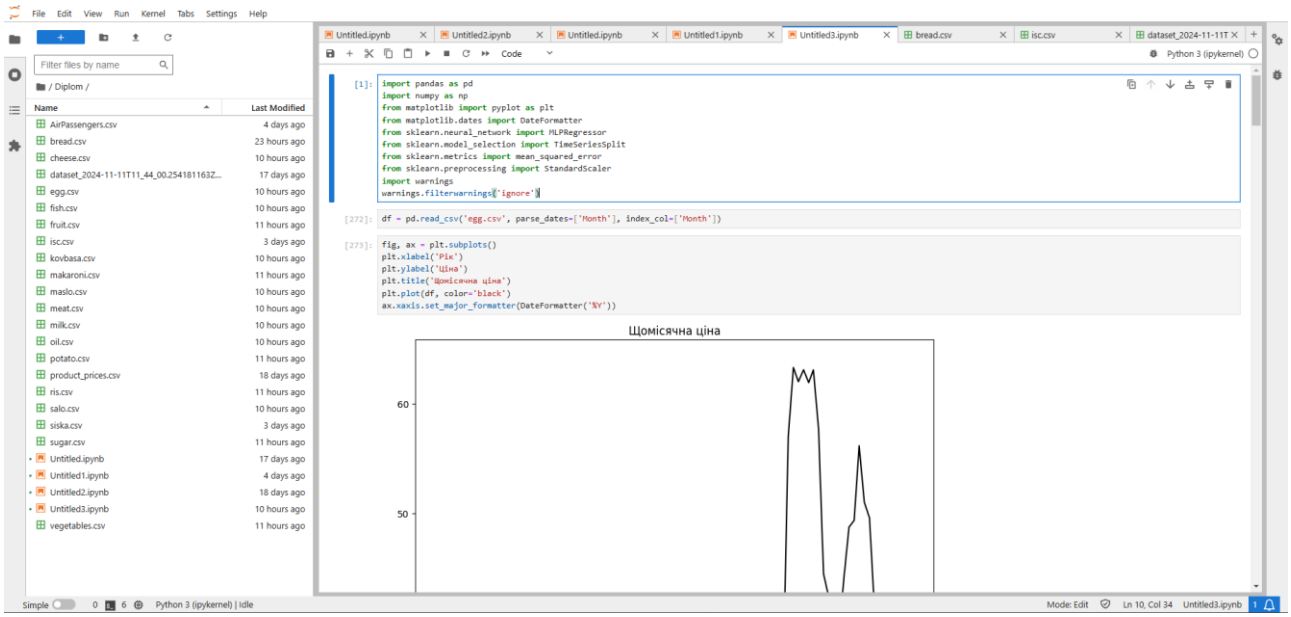



Рис.3.1 Робоче вікно Jupyter Notebook

Клітини коду: Основна частина ноутбука складається з клітин, в яких можна писати і виконувати код. Кожна клітина є окремим блоком коду, який можна

запускати незалежно. Це дозволяє гнучко працювати з кодом, виконуючи окремі частини програми.

Для пояснень, описів, коментарів і математичних формул використовуються клітини з текстом, які підтримують розмітку Markdown і LaTeX. Це дозволяє створювати документи, які не тільки містять код, але й зручні для читання і пояснень.

Jupyter підтримує використання LaTeX для написання математичних формул прямо в текстових клітинах, що робить його ідеальним для наукових робіт і досліджень.

Після виконання клітини з кодом результат (виведення) з'являється прямо під нею, що дозволяє миттєво бачити результат обчислень або візуалізації. Це значно полегшує відстеження помилок і покращує взаємодію з даними.

Jupyter Notebook підтримує інтерактивність завдяки можливості виконувати код поетапно. Це означає, що ви можете протестувати окремі частини програми, відразу отримувати результати і вносити зміни на ходу. Для візуалізації даних доступні бібліотеки, такі як Matplotlib, Seaborn, Plotly. Ці бібліотеки дозволяють створювати графіки та діаграми безпосередньо в ноутбуці, що забезпечує зручний інтерфейс для аналізу результатів.

Jupyter Notebook дозволяє не тільки працювати з кодом, але й імпортувати та обробляти різноманітні формати даних, такі як CSV, Excel, JSON, SQL, а також здійснювати інтеграцію з іншими платформами, базами даних та інструментами для обробки даних.

Один з основних переваг Jupyter Notebook — це можливість зберігати та обмінюватися готовими ноутбуками. Документи можна легко експортувати у різні формати, такі як HTML, PDF, LaTeX, або просто зберігати у форматі .ipynb, що є стандартом для Jupyter. Крім того, ноутбуки можна розміщувати на платформах, таких як GitHub або NBViewer, що дозволяє працювати в команді і ділитися результатами досліджень.

Jupyter Notebook інтегрується з іншими інструментами для наукових обчислень і обробки даних. Наприклад, можна використовувати SciPy для

математичних обчислень, Pandas для обробки таблиць, а також підключати різні бібліотеки для роботи з машинним навчанням, наприклад, TensorFlow або Keras.

### 3.1.4. Pandas

Pandas — це одна з найпопулярніших бібліотек для роботи з даними в Python. Вона надає потужні структури даних і функції для маніпулювання та аналізу великих обсягів даних [26]. Pandas ідеально підходить для роботи з табличними даними, такими як Excel-файли, CSV, SQL бази даних та інші формати. Завдяки Pandas можна ефективно обробляти, очищати, змінювати та візуалізувати дані. Бібліотека стала стандартом де-факто для аналізу даних у Python і широко використовується в науці про дані, статистиці, фінансах, машинному навчанні та інших областях.

Структури даних: Pandas надає дві основні структури даних:

- Series: Одновимірний масив, схожий на список або масив в Python, але з можливістю додавати індекси до кожного елемента. Це дозволяє ефективно працювати з даними, де кожен елемент має відповідний індекс, що може бути числовим або рядковим.
- DataFrame: Двовимірна структура даних, схожа на таблицю або базу даних, яка складається з рядків і стовпців. DataFrame є основною структурою для зберігання та маніпулювання табличними даними. Він підтримує індекси для рядків і стовпців, що дозволяє працювати з великими наборами даних, зручно доступними за допомогою індексів або імен.

Імпорт і експорт даних: Однією з найбільших переваг Pandas є зручність імпорту та експорту даних у різних форматах. Ви можете працювати з даними, що зберігаються у CSV, Excel, SQL, JSON, HDF5, Parquet і багатьох інших форматах. Завдяки методам, таким як read_csv(), read_excel(), to_sql(), to_json(), можна легко завантажувати дані з різних джерел або зберігати результати роботи у потрібному форматі.

Наприклад код для завантаження csv та зберігання його вмісту в Excel формат:

```
import pandas as pd
df = pd.read_csv('data.csv')  # Завантаження CSV файлу в DataFrame
df.to_excel('output.xlsx')  # Експорт в Excel
```

Pandas дозволяє здійснювати численні операції з даними:

- Фільтрація: Можна вибирати підмножини даних за умовами, використовуючи логічні операції.
- Операції з колонками: Можна додавати нові стовпці, змінювати або видаляти існуючі.
- Агрегування: Використовуються функції для підсумовування, обчислення середніх значень, максимальних та мінімальних значень, а також групування за певними ознаками.
- Заповнення пропусків: За допомогою методів fillna() або dropna() можна працювати з пропущеними даними.

Наприклад:

```
df['new_column'] = df['column1'] + df['column2']  # Додавання нового стовпця
df = df[df['column1'] > 10]  # Фільтрація даних
df.groupby('category').mean()  # Групування і обчислення середнього для кожної категорії
df.fillna(0)  # Заповнення пропущених значень нулями
```

Pandas забезпечує простий і потужний механізм індексації та доступу до даних. Ви можете використовувати числові індекси або індекси за іменем для доступу до рядків і стовпців:

- df['column'] для доступу до стовпця.
- df.loc[] для доступу до рядків за індексами або умовами.
- df.iloc[] для доступу до рядків за позицією.

Наприклад:

```
df.loc[0]  # Доступ до першого рядка за індексом
```

```
df.iloc[0]  # Доступ до першого рядка за позицією

df['column1']  # Доступ до стовпця 'column1'
```

Pandas має вбудовані інструменти для роботи з датами та часом, що дозволяє легко перетворювати рядки в об'єкти типу datetime, здійснювати операції з датами (наприклад, обчислення різниці між датами) та зручно працювати з часовими рядами.

Наприклад:

```
df['date'] = pd.to_datetime(df['date'])  # Перетворення рядка в datetime

df['day'] = df['date'].dt.day  # Витягування дня з дати

df['month'] = df['date'].dt.month  # Витягування місяця
```

Pandas інтегрується з бібліотеками для візуалізації, такими як Matplotlib і Seaborn, що дозволяє швидко створювати графіки безпосередньо з DataFrame. За допомогою методу plot() можна створювати різні типи графіків (лінійні, гістограми, діаграми розсіювання тощо).

Наприклад:

```
df['column1'].plot(kind='hist')  # Побудова гістограми

df.plot(x='date', y='value', kind='line')  # Лінійний графік
```

Pandas підтримує базові статистичні функції, такі як середнє, стандартне відхилення, кореляція, коваріація, а також роботу з більш складними метриками. Це дозволяє ефективно аналізувати дані та витягувати з них статистичну інформацію.

Наприклад:

```
df.mean()  # Середнє значення для кожного стовпця

df.corr()  # Кореляція між стовпцями
```

Злиття, об'єднання та з'єднання даних: Pandas надає потужні методи для об'єднання різних наборів даних, такі як merge(), concat() і join(). Це дозволяє зручно комбінувати дані з кількох таблиць за певними ключами або індексами.

Наприклад:

```
pd.merge(df1, df2, on='id')  # Злиття двох DataFrame за стовпцем 'id'
pd.concat([df1, df2])  # Об'єднання двох DataFrame
```

Переваги Pandas:

- Pandas дозволяє здійснювати складні операції з даними за допомогою простого і інтуїтивно зрозумілого синтаксису;
- Бібліотека оптимізована для роботи з великими обсягами даних, тому операції виконуються швидко і ефективно;
- Pandas підтримує різноманітні типи даних (числові, категоріальні, текстові, часо-залежні) і дає змогу працювати з ними в одному середовищі;
- Pandas можна легко інтегрувати з іншими бібліотеками Python для машинного навчання (наприклад, scikit-learn), візуалізації (наприклад, Matplotlib, Seaborn) та обробки великих даних (наприклад, Dask).

Pandas є основним інструментом для аналізу даних у Python, який дозволяє ефективно працювати з табличними даними, маніпулювати ними, здійснювати статистичні обчислення та візуалізацію. Завдяки потужним функціям і зручному синтаксису, Pandas значно спрощує процес обробки та аналізу даних, що робить її незамінним інструментом для всіх, хто працює з даними.

### 3.1.5. NumPy

NumPy (скорочення від Numerical Python) — це основна бібліотека для наукових обчислень у Python, яка надає підтримку для великих багатовимірних масивів і матриць, а також надає великий набір математичних функцій для роботи з ними [27]. Вона є основою для багатьох інших бібліотек, таких як Pandas, SciPy,

Matplotlib та інші, і використовується в багатьох областях, таких як машинне навчання, обробка зображень, статистика, обчислювальна фізика та інші.

Основна структура даних в NumPy — це ndarray (n-dimensional array), який дозволяє працювати з багатовимірними масивами. Це схоже на списки в Python, але з кількома важливими відмінностями:

- Всі елементи масиву мають однаковий тип (наприклад, всі можуть бути числами з плаваючою комою);
- Масиви NumPy значно швидші та ефективніші, оскільки вони оптимізовані для обробки числових даних;
- Вони підтримують векторизовані операції (операції над усіма елементами масиву одночасно), що значно прискорює обчислення.

Для створення масиву NumPy використовують функцію np.array(), а також є спеціальні функції для створення масивів певних розмірів і значень, наприклад np.zeros(), np.ones(), np.arange(), np.linspace().

Приклад:

```
import numpy as np
# Створення одновимірного масиву
arr = np.array([1, 2, 3, 4, 5])
# Створення двовимірного масиву
arr2 = np.array([[1, 2, 3], [4, 5, 6]])
```

Однією з головних переваг NumPy є підтримка векторизації — можливість виконувати математичні операції над цілими масивами без використання циклів Python. Це дозволяє значно скоротити час виконання та зробити код більш елегантним і читабельним.

Наприклад:

```
arr = np.array([1, 2, 3, 4])
arr2 = arr * 2  # Векторизована операція (множення кожного елемента на 2)
```

Це еквівалентно циклу:

python

Копировать код

```
arr2 = []
for i in arr:
    arr2.append(i * 2)
```

Векторизація дозволяє уникнути використання явних циклів, що робить код значно швидшим.

NumPy надає величезну кількість вбудованих математичних функцій, таких як:

- Арифметичні операції: додавання, віднімання, множення, ділення, піднесення до степеня;
- Лінійна алгебра: операції з матрицями, обчислення визначника, зворотної матриці, власних значень та векторів, множення матриць;
- Статистичні функції: середнє значення, стандартне відхилення, медіана, коваріація, кореляція, максимуми та мінімуми.
- Операції з числами: синус, косинус, логарифм, експонента, арифметичні функції тощо.

Приклад:

```
arr = np.array([1, 2, 3, 4])

mean_val = np.mean(arr)  # Середнє значення

sum_val = np.sum(arr)  # Сума елементів

sqrt_val = np.sqrt(arr)  # Квадратний корінь кожного елемента
```

NumPy підтримує зручну індексацію для доступу до елементів масиву, а також розширене слайсінг (нарізка) для вибору підмножини масиву. Ось як це працює:

- Одновимірний масив:

  arr = np.array([1, 2, 3, 4, 5])

  print(arr[0])  # Перший елемент

  print(arr[1:4])  # Слайс від другого до четвертого елемента
- Двовимірний масив (матриця):

  arr2 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

  print(arr2[0, 1])  # елемент в першому рядку і другому стовпці

  print(arr2[:, 1])  # Вибір другого стовпця

NumPy дозволяє змінювати форму масивів за допомогою функції reshape(), а також виконувати транспонування масивів для зміни їх розмірностей.

Приклад:

arr = np.array([1, 2, 3, 4, 5, 6])

arr_reshaped = arr.reshape(2, 3)  # Перетворення на 2x3 масив

arr_transposed = arr_reshaped.T  # Транспонування масиву

NumPy підтримує роботу з багатовимірними масивами (матрицями і тензорами). Це дозволяє ефективно працювати з багатовимірними даними, що особливо важливо для машинного навчання та обробки зображень.

Приклад:

arr = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])

print(arr.shape)  # Показує розмір масиву (2, 2, 2)

Оскільки NumPy масиви оптимізовані для числових обчислень, вони працюють значно швидше, ніж звичайні списки Python. Це важливо при обробці великих обсягів даних або складних математичних операцій.

NumPy є основою для багатьох інших бібліотек, таких як Pandas (для аналізу даних), SciPy (для наукових обчислень), Matplotlib (для візуалізації) і scikit-learn (для машинного навчання). Оскільки NumPy масиви є стандартом для числових даних у Python, вони забезпечують сумісність між різними бібліотеками.

Переваги NumPy:

- NumPy значно швидший за стандартні списки Python при роботі з великими даними;
- Надає велику кількість математичних та статистичних функцій для обчислень;
- Має інтуїтивно зрозумілий інтерфейс для роботи з масивами та матрицями;
- Підтримка багатовимірних масивів, змінних розмірів і типів даних.

NumPy — це основний інструмент для числових обчислень у Python, і він є незамінним у багатьох сферах, де важлива ефективність та масштабованість обчислень.

### 3.1.6. Matplotlib

Matplotlib — це одна з найпопулярніших бібліотек для візуалізації даних в Python, яка дозволяє створювати статичні, анімовані та інтерактивні графіки [28]. Бібліотека надає велику кількість можливостей для створення різноманітних візуалізацій, таких як графіки, гістограми, діаграми розсіювання, теплові карти, 3D графіки і багато іншого.

Matplotlib підтримує широкий спектр типів графіків, серед яких:

- Лінійні графіки (plot());
- Гістограми (hist());
- Діаграми розсіювання (scatter());
- Стовпчикові діаграми (bar());
- Кругові діаграми (pie());
- Теплові карти (imshow());
- 3D графіки.

Для створення графіків у Matplotlib зазвичай використовують об'єкт Figure (який представляє весь графік) і Axes (який є ділянкою на графіку, де безпосередньо малюється зображення).

Приклад:

```
import matplotlib.pyplot as plt
import numpy as np

# Створення даних
x = np.linspace(0, 10, 100)
y = np.sin(x)

# Створення лінійного графіка
plt.plot(x, y)
plt.title('Синусоїдальний графік')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

Matplotlib дозволяє налаштовувати вигляд графіка за допомогою численних параметрів:

- Легенди: додаються за допомогою plt.legend();
- Теми: бібліотека має вбудовані стилі оформлення, які можна застосовувати для зміни вигляду графіків;
- Титули та підписи: plt.title(), plt.xlabel(), plt.ylabel() дозволяють додавати заголовки та підписи до осей;
- Шкали осей: можна налаштувати масштаби, типи осей (лінійний або логарифмічний), а також обмеження осей;
- Колір, маркери та лінії: параметри для налаштування кольору, стилю ліній, маркерів точок на графіку.

Matplotlib дозволяє виводити кілька графіків в одному вікні, використовуючи subplot. Це особливо корисно для порівняння декількох графіків або результатів.

Графіки можна зберігати в різних форматах (наприклад, PNG, JPEG, PDF, SVG) за допомогою функції savefig(). Це дозволяє експортувати графіки для подальшого використання або публікацій.

Matplotlib підтримує візуалізацію багатовимірних масивів даних за допомогою функції imshow(), що дає можливість створювати теплові карти або відображати зображення.

Matplotlib підтримує інтерактивні графіки, які можна інтегрувати в Jupyter Notebooks або веб-застосунки. Для інтерактивних можливостей можна використовувати такі модулі, як mpl_toolkits.mplot3d (для 3D-графіки) або widgets для створення інтерактивних елементів в графіках.

Matplotlib також підтримує створення 3D графіків за допомогою модуля mplot3d. Це дозволяє створювати 3D лінійні графіки, діаграми розсіювання, поверхневі графіки та інші 3D-візуалізації.

Matplotlib добре інтегрується з іншими бібліотеками для обробки даних та аналізу, такими як Pandas і NumPy, для ефективної візуалізації даних, що зберігаються у DataFrame або масивах.

Переваги Matplotlib:

- Широкий функціонал: підтримує створення найрізноманітніших типів графіків;
- Гнучкість налаштувань: дозволяє детально налаштовувати вигляд графіків;
- Інтерактивність: підтримка інтерактивних можливостей у Jupyter Notebooks та інших середовищах;
- Широка підтримка: є стандартною бібліотекою для візуалізації в Python, яка має активну спільноту та велику кількість ресурсів для навчання.

Matplotlib є невід'ємною частиною екосистеми Python для аналізу даних і широко використовується в наукових дослідженнях, бізнес-аналітиці, інженерії та багатьох інших галузях.

### 3.1.7. Scikit-learn

Scikit-learn (скорочено sklearn) — це одна з найбільш популярних і широко використовуваних бібліотек Python для машинного навчання [28]. Вона надає простий і зручний інтерфейс для реалізації широкого спектра алгоритмів машинного навчання, статистичних моделей та інструментів для обробки даних. Бібліотека розроблена з урахуванням ефективності, масштабованості та зручності використання, що робить її чудовим вибором для швидкого прототипування та наукових досліджень.

Scikit-learn підтримує величезну кількість алгоритмів для різних завдань машинного навчання, таких як:

- Класифікація: Алгоритми, які допомагають відносити вхідні дані до певних категорій або класів (наприклад, метод опорних векторів (SVM), дерева рішень, наївний байєсів класифікатор, логістична регресія, K-Nearest Neighbors (KNN));
- Регресія: Алгоритми для прогнозування числових значень (лінійна регресія, регресія на основі дерев рішень, ласо- та рідж-регресія);
- Кластеризація: Алгоритми для групування даних в класи без попереднього позначення (наприклад, K-means, DBSCAN, агломеративна кластеризація);
- Зниження вимірності: Техніки для зменшення кількості змінних при збереженні важливої інформації (наприклад, метод головних компонент (PCA), t-SNE);
- Обробка аномалій: Алгоритми для виявлення аномалій у даних (наприклад, One-Class SVM).

Scikit-learn надає вбудовані інструменти для попередньої обробки даних, що включають:

- Нормалізація та стандартизація даних (наприклад, StandardScaler, MinMaxScaler);

- Перетворення категоріальних змінних в числові (за допомогою OneHotEncoder та LabelEncoder);
- Заповнення пропусків (імпутація пропущених значень через SimpleImputer);
- Розбиття наборів даних на тренувальні та тестові за допомогою функції train_test_split;
- Конвеєри (Pipelines): Вони дозволяють об’єднувати кілька етапів обробки та моделювання в один об'єкт, що спрощує створення складних робочих процесів.

Scikit-learn пропонує кілька інструментів для оцінки продуктивності моделей:

- Крос-валідація (cross_val_score, GridSearchCV): дозволяє оцінити стабільність моделі за допомогою різних підмножин тренувальних даних;
- Метрики продуктивності: надаються метрики для оцінки результатів моделей, такі як точність, повнота, F1-міра, коефіцієнт детермінації ($R^2$) для регресії;
- Покращення гіперпараметрів: Scikit-learn включає інструменти для налаштування гіперпараметрів, наприклад, GridSearchCV для систематичного перебору параметрів або RandomizedSearchCV для випадкового пошуку.

Scikit-learn також включає ансамблеві методи для покращення точності моделі, такі як: MLP, Random Forests, Gradient Boosting, AdaBoost, Bagging.

Scikit-learn має простий, інтуїтивно зрозумілий інтерфейс, який працює на основі об'єктно-орієнтованого підходу. Це означає, що всі моделі, трансформери та процеси обробки даних реалізовані як об'єкти, які можна зручно комбінувати для створення робочих процесів.

Вона також добре інтегрується з іншими популярними бібліотеками, такими як NumPy, Pandas і Matplotlib, що дозволяє легко маніпулювати даними, візуалізувати результати та здійснювати подальший аналіз.

Переваги Scikit-learn:

- Включає всі основні алгоритми машинного навчання;
- Легкий у освоєнні інтерфейс, що дозволяє швидко створювати моделі;
- Бібліотека оптимізована для роботи з великими об'ємами даних;
- Легко поєднується з Pandas, NumPy і Matplotlib для комплексної обробки та візуалізації даних;
- Має детальну документацію, що допомагає вивчати і працювати з бібліотекою.

Scikit-learn є однією з найбільш важливих бібліотек для машинного навчання в Python і використовується в науці, промисловості, фінансах, медицині та багатьох інших галузях.

## 3.2. Опис методу прогнозування вартості продуктової корзини

Метод прогнозування вартості продуктової корзини включає в себе декілька етапів.

Першим етапом є підготовка даних. Необхідно підготувати датасет для роботи програми. Для цього було підготовлено дані про історичну ціну продуктів, з якої також можна побачити сезонність продукту та тренд ціни, також інформацію про курс валют, інфляцію, індекс споживчих цін, витрати на паливо та інші фактори що впливають на ціну продукту.

Другим етапом необхідно обробити датасет для подання до програмного середовища. Для цього необхідно стандартизувати вхідні дані, а саме видалити помилки та пропуски в даних, нормалізувати дані. Також необхідно розділити початкові дані на дві частини: 80% - це дані на яких буде навчатися нейронна мережа, та 20% - це дані які будуть використанні для тестування прогнозу.

Третім етапом необхідно виділити сезонність та тренд з початкових даних.

Четвертим етапом необхідно нормалізувати дані. Цей етап допоможе пришвидшити час навчання нейронної мережі та значно підвищить її точність.

П’ятим етапом йде підбір параметрів для навчання нейронної мережі. В якості моделі нейронної мережі було обрано модель MLP. Їй необхідно вказати кількість скритих шарів. Для цього методом прямого перебору від одного до ста було обрано найоптимальнішу кількість шарів для кожних даних. Оскільки якщо буде замало шарів – це призведе до зниження точності та недонавчання, якщо сильно багато – це призведе то перенавчанню, що також значно знизить точність. Це дозволить отримати максимально точний прогноз вартості.

Шостим етапом є навчання моделі на тестових даних за допомогою моделі MLP.Для цього її передається оптимальна кількість скритих шарів що була визначена на минулому етапі, максимальну кількість ітерацій, який виконає алгоритм оптимізації під час навчання, для підвищення точності рекомендується збільшувати цей параметр, оскільки якщо залишити його за замовчуванням, то модель може впертися в цей ліміт та недонавчитися, та передати моделі параметр повторюваності, щоб мати модель при зберігання решти факторів могла відтворювати результат.

Сьомим етапом необхідно порівняти початкові дані з результатом прогнозування. Для цього використовується коефіцієнт детермінації $R^2$ за допомогою якого можна оцінити точність та порівняти її з іншими методами.

## 3.3. Розробка програмного застосунку

В якості вхідний даних було використано офіційні дані з офіційного сайту Державної статистики України, а саме середні споживчі ціни на товари та послуги. В датасеті міститься інформація з початку 2017 року та жовтня 2024.

Було проведено передбачення на 16 основних продуктах продуктової корзини, а саме: яйця, хліб, твердий сир, риба, фрукти, ковбаса, картопля, рис, соняшникова олія, молоко, вершкове масло, макаронні вироби, м’ясо, цукор, сало та овочі.

Для кожного з цих продуктів було створено свій власний датасет у форматі csv, оскільки в датасеті від Держстату є дуже багато інформації про продукти та послуги які не входять до продуктової корзини, як ми не оброляємо. Дані в нових датасетах представлені у форматі показаний приклад одного з них.

Всього в кожному наборі 94 записи. Ці дані ми можемо представити у вигляді графіку. На рисунку (3.3) показаний приклад цих даних у вигляді графіку.

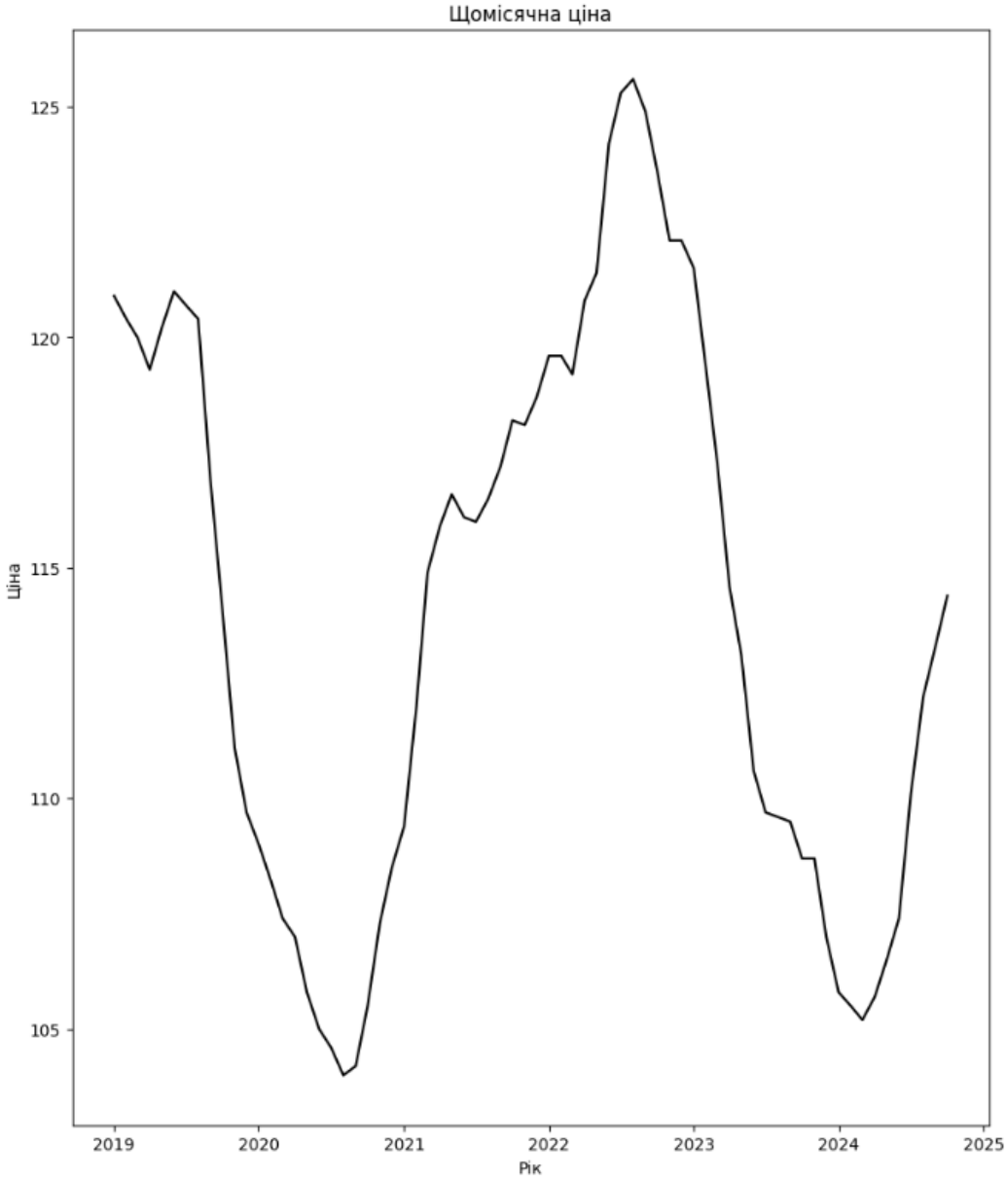
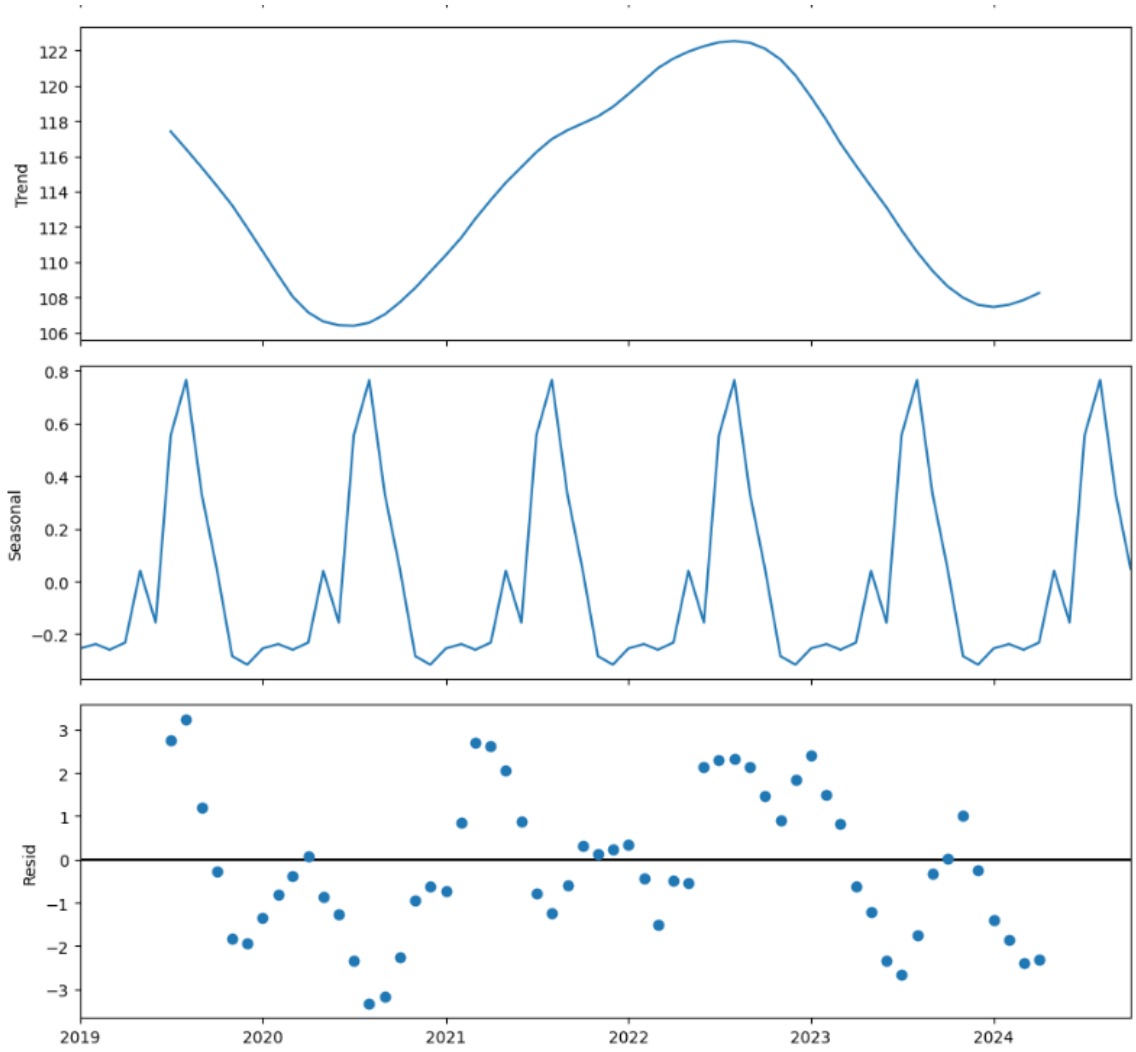



Рис 3.3 Графік цін на твердий сир

Також розроблений програмний застосунок розбиває початкові дані на 3 часових ряди для того щоб проаналізувати тренд, сезонність та залишки. Приклад цих графіків наведено на рисунку (3.4).




Риз 3.4 Графіки тренду, сезонності та залишків цін на твердий сир

Ці 3 графіки дають нам повну картину представлених даних. На них ми бачимо чітко виражену сезонність кожного року. Залишки даних які не будуть використовуватись в навчанні та зростаючий тренд.

Для того щоб почати обробляти вхідні дані нам необхідно їх спочатку перетворити на масиву формату n x (k + 1), де n - кількість вибірок, а k - кількість ознак. З минулого етапу ми бачимо що повний цикл в сезонності в нас це рівно рік, тому за k візьмемо 12.

В коді це буде виглядати наступним чином:

```
k = 12
Z = []
for i in range(k + 1, df.shape[0] + 1):Z.append(df.iloc[(i - k - 1): i, 0])
Z = np.array(Z)
Z.shape
```

Далі для проведення порівняння з завчасно відомим результатом необхідно розділити дані на 80% на яких ми будемо навчати нейронну мережу та 20% з якими ми буде порівнювати результат. В коді це виглядає наступним чином:

```
split = int(0.8 * Z.shape[0])
print(split)
Z_train, Z_test = Z[:split, :], Z[split:, :]
```

Та необхідно прибрати даних ознаки та цільове значення.

```
X_train, y_train = Z_train[:, :-1], Z_train[:, -1]
X_test, y_test = Z_test[:, :-1], Z_test[:, -1]
```

Після цього необхідно обчислити оптимальну кількість скритих шарів нейронної мережі для того щоб отримати максимальну точність та уникнути перенавчання. Зробити це можна застосовуючи RandomizedSearchCV який перебере всі можливі варіанти в діапазоні та видасть оптимальний результат.

```
param_distributions = {
   'hidden_layer_sizes': [(np.random.randint(1, 100),) for _ in range(10)]
}
model = MLPRegressor(max_iter=1000, random_state=1)
random_search = RandomizedSearchCV(estimator=model, param_distributions=param_distributions, n_iter=10, cv=5, scoring='neg_mean_squared_error')
random_search.fit(X_train, y_train)
print("Best hidden_layer_sizes:", random_search.best_params_)
```

Тепер коли маємо оптимальну кількість шарів можна приступити до навчання самої нейронної мережі методом MLP.

mlp=MLPRegressor(hidden_layer_sizes=random_search.best_params_['hidden_layer_sizes'], max_iter=1000, random_state=1)

mlp.fit(X_train, y_train)

Де кількість шарів – це результат обчислень з попереднього кроку, максимальна кількість ітерацій – 1000, та фіксуємо генератор випадкових чисел таким чином, що при кожному запуску коду можна було отримати ті ж самі початкові ваги, той же порядок вибору даних, і, відповідно, той же результат навчання, за інших однакових умов. Оскільки ми явно не вказуємо метод активації, то за замовчуванням буде використано метод ReLU.

Та обчислюємо точність фінального результата за методом R-квадрат.

print(mlp.score(X_train, y_train))

Формула обчислення R-квадрат (3.1) представлена нижче.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}, \qquad (3.1)$$

де $y_i$ – справжнє значення;

$\hat{y}_i$ – передбачене значення;

$\bar{y}$ – середнє значення $y$.

Результат може варіюватись 0 до 1, де 0 - модель не пояснює варіацію даних краще, ніж середнє значення, 1 - модель ідеально передбачає дані.

Результат точності представлено в таблиці (3.1).

Таблиця 3.1

Аналіз точності прогнозування продуктів

| Назва продукту | Точність прогнозування |
|---|---|
| яйця | 0,947513086 |
| хліб | 0,939278868 |
| твердий сир | 0,968023505 |
| риба | 0,9874993 |
| фрукти | 0,906683275 |
| ковбаса | 0,993955902 |
| картопля | 0,984491024 |

Продовження таблиці 3.1

Аналіз точності прогнозування продуктів

| Назва продукту | Точність прогнозування |
|---|---|
| вершкове масло | 0,931611505 |
| макаронні вироби | 0,95910891 |
| м’ясо | 0,932001145 |
| цукор | 0,990220459 |
| сало | 0,992326033 |
| овочі | 0,97009841 |
| рис | 0,987714653 |
| соняшникова олія | 0,986037923 |
| молоко | 0,972650709 |
| середнє | 0,965575919 |

Та для наочної демонстрації створюємо графік порівняння прогнозу та очікування.

```
fig, ax = plt.subplots()
plt.xlabel('Рік')
plt.ylabel('Ціна')
plt.title('Щомісячна ціна')

plt.plot(df, color='black')
plt.plot(pd.Series(y_pred, index=df.index[-len(y_test):]), color='red', label='Прогноз')
plt.legend(bbox_to_anchor=(1.05, 1))
ax.xaxis.set_major_formatter(DateFormatter('%Y'))
```

### 3.4. Результат роботи

Результатом роботи є графіки на яких чорним кольором зображено тренувальні дані та червоним зображено прогноз який був отриманий за допомогою використання розробленого методу.

Графіки порівняння прогнозу та початкових даних зображені на рисунках (3.5-3.20).

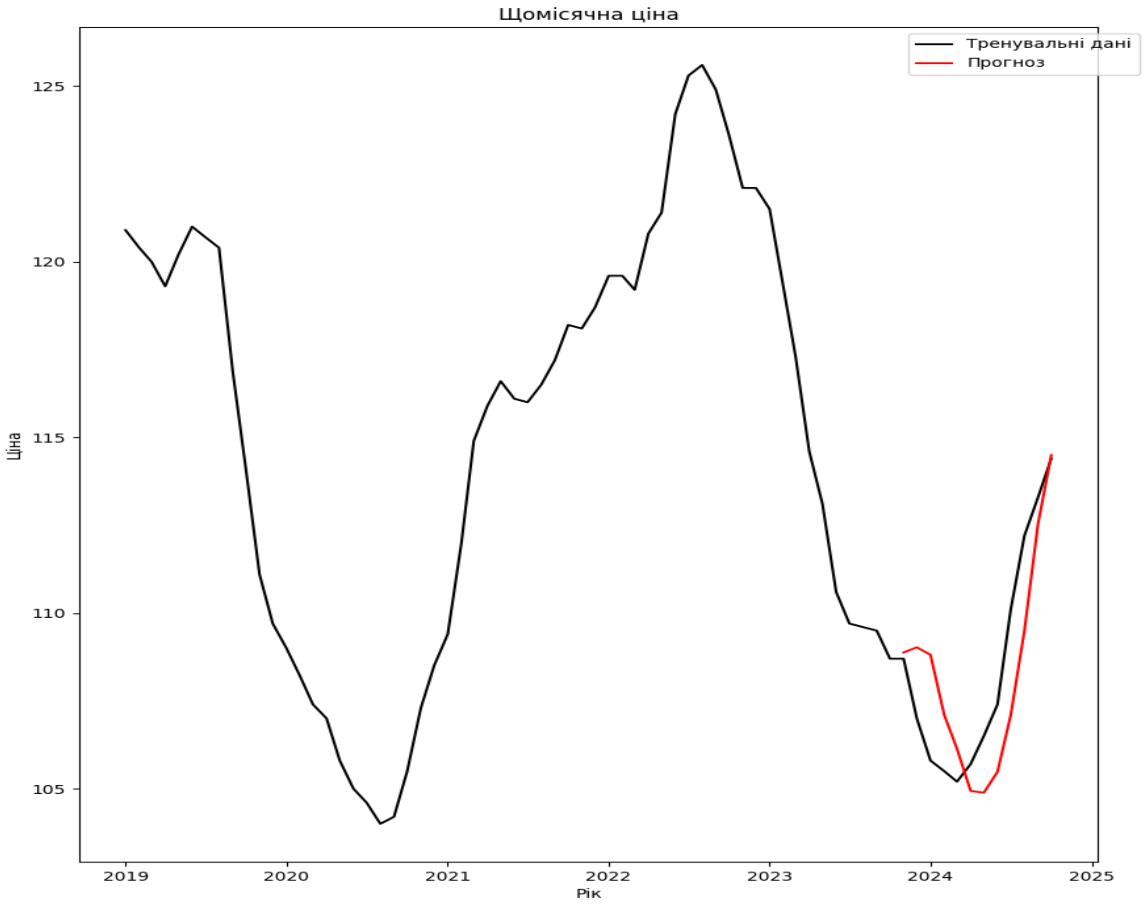



Рис. 3.5 Результат прогнозування вартості хлібу

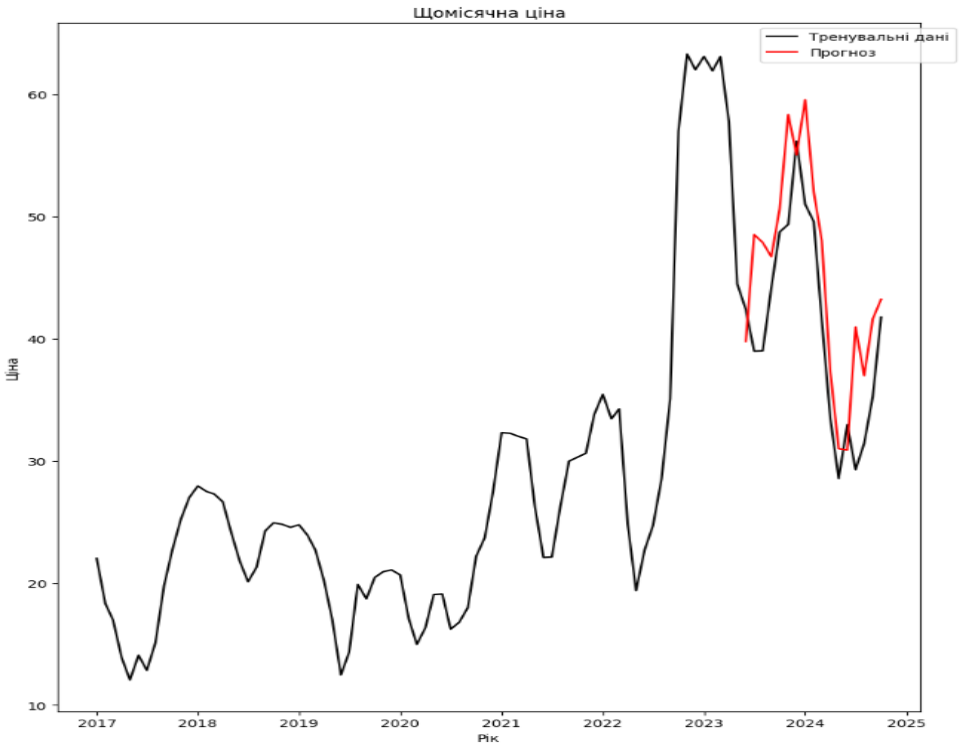



Рис. 3.6 Результат прогнозування вартості яєць

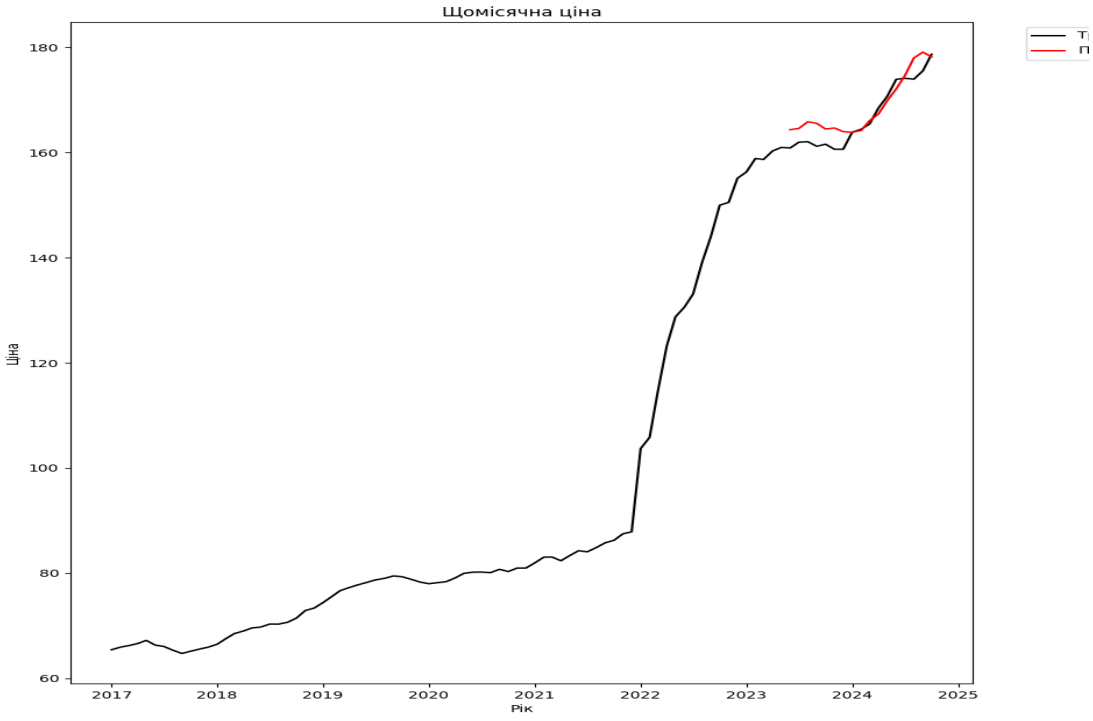



Рис. 3.7 Результат прогнозування вартості риби

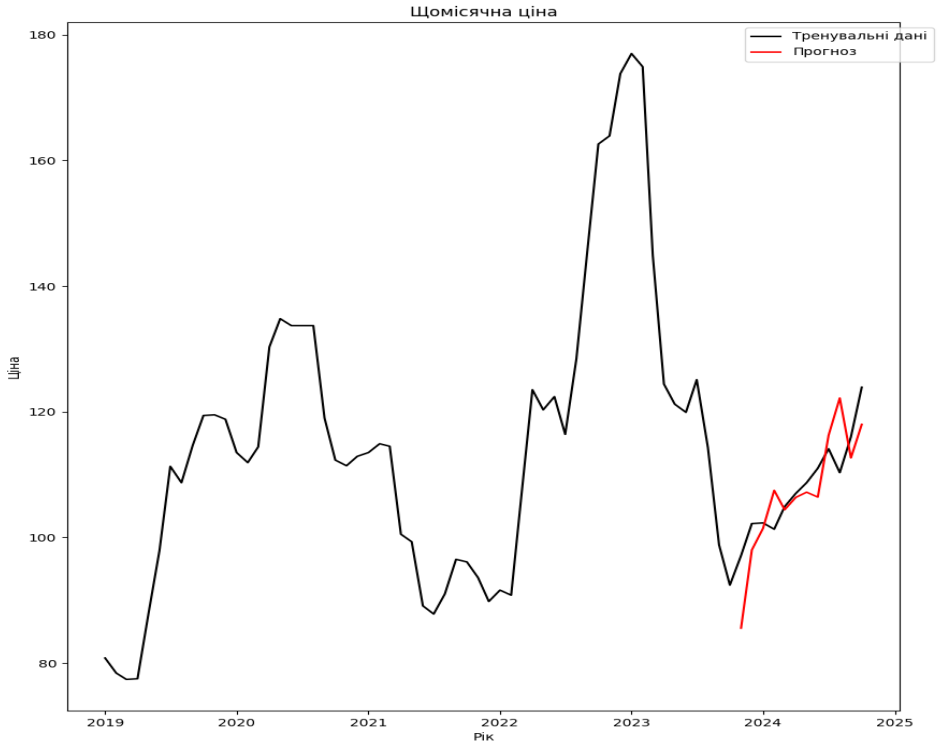



Рис. 3.8 Результат прогнозування вартості фруктів

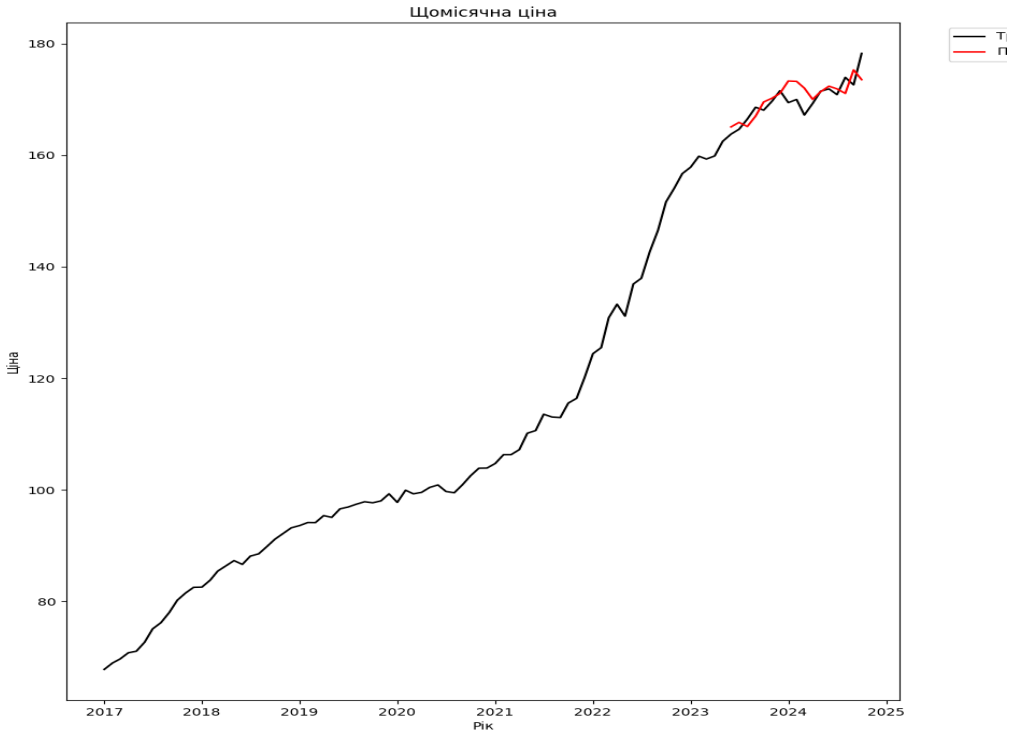



Рис. 3.9 Результат прогнозування вартості ковбаси

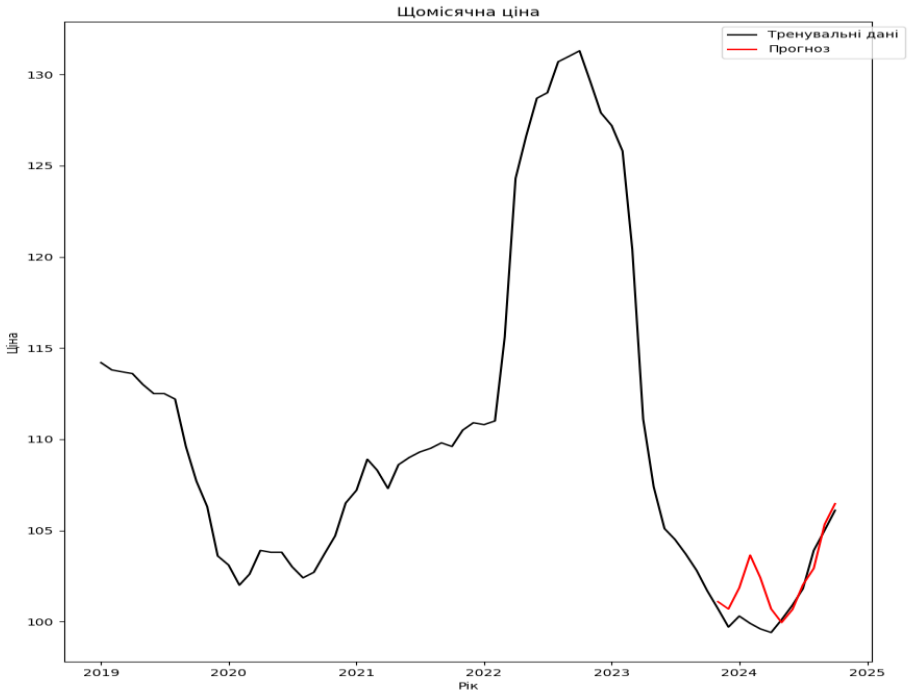



Рис. 3.10 Результат прогнозування вартості макаронних виробів

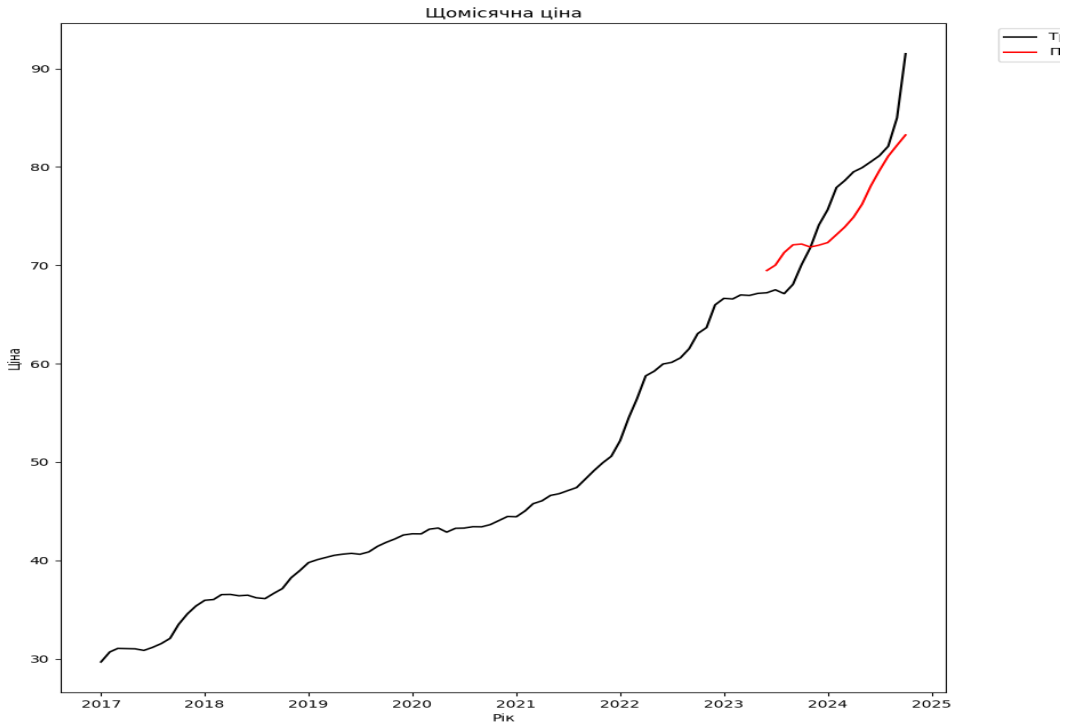



Рис. 3.11 Результат прогнозування вартості вершкового масла

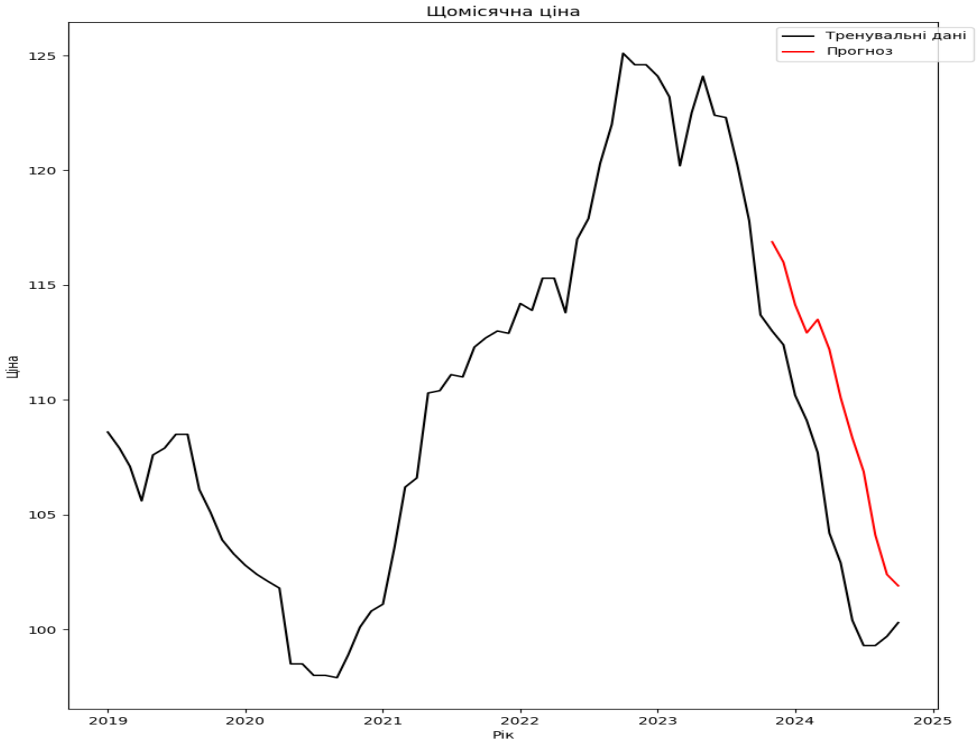



Рис. 3.12 Результат прогнозування вартості м’яса

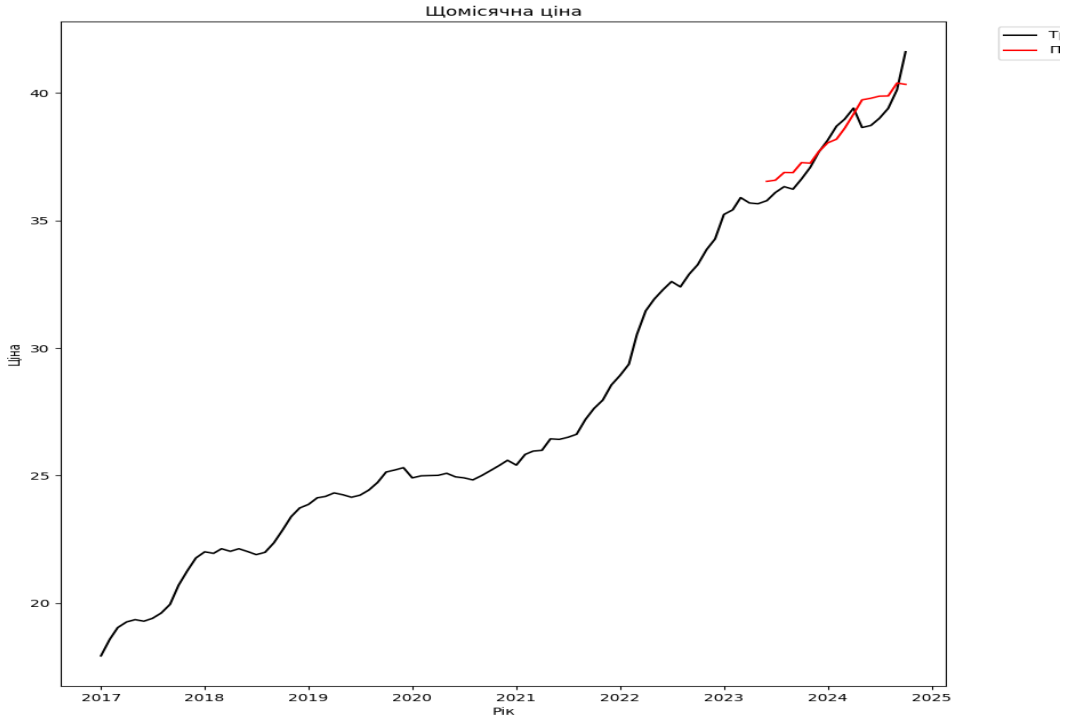



Рис. 3.13 Результат прогнозування вартості молока

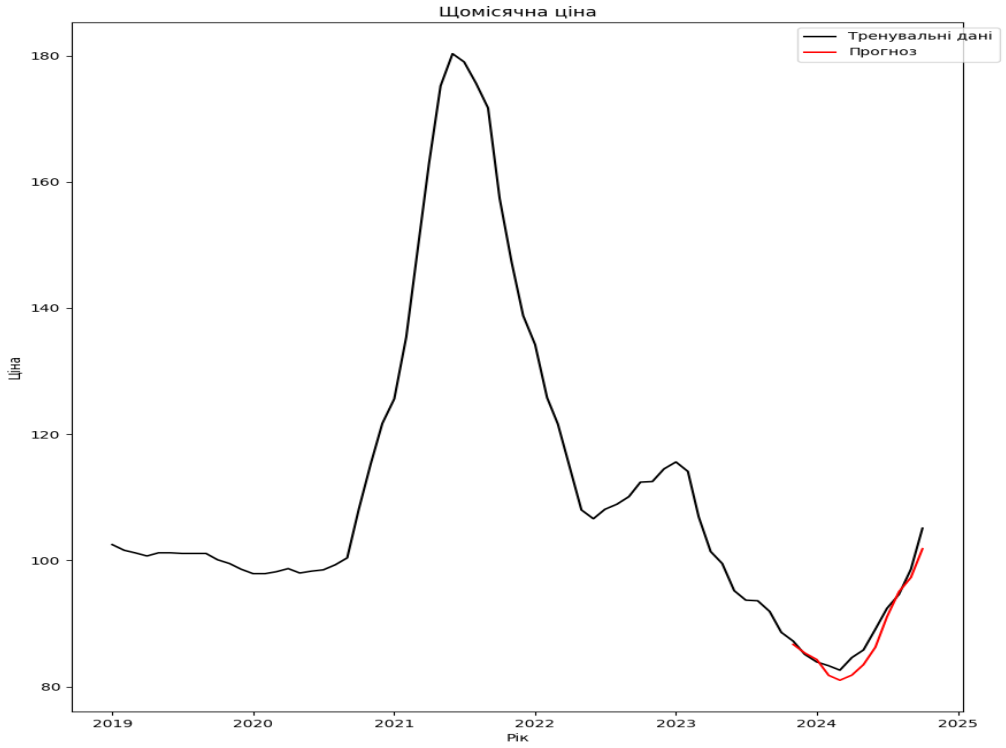



Рис. 3.14 Результат прогнозування вартості соняшникової олії

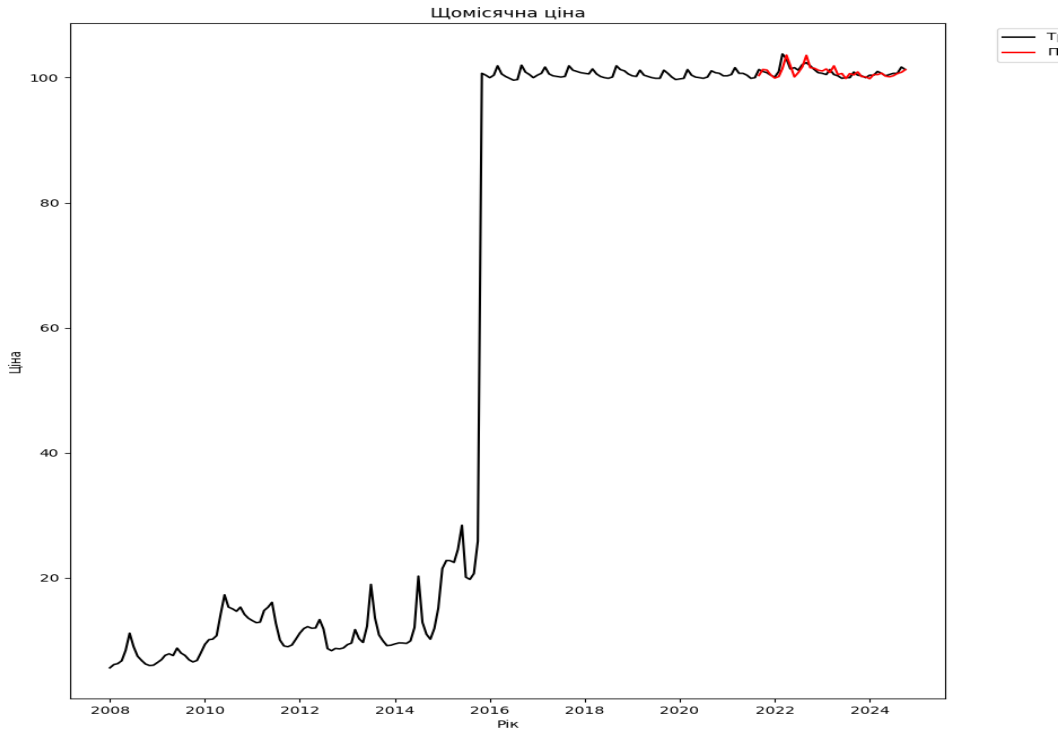



Рис. 3.15 Результат прогнозування вартості картоплі

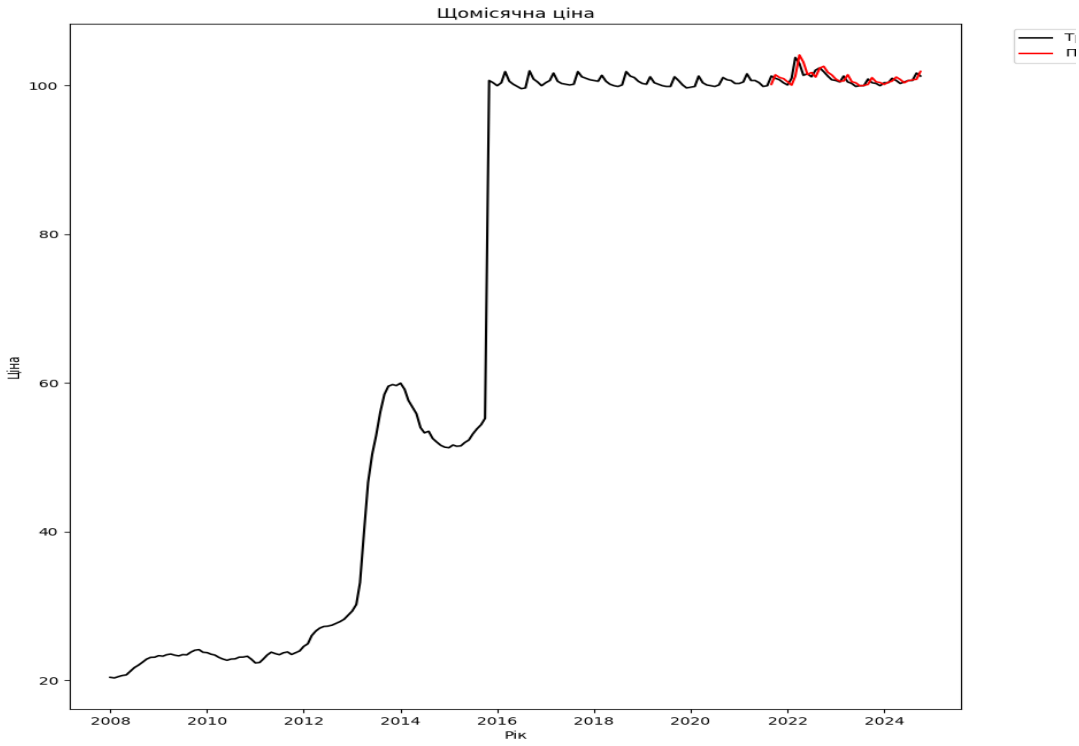



Рис. 3.16 Результат прогнозування вартості рису

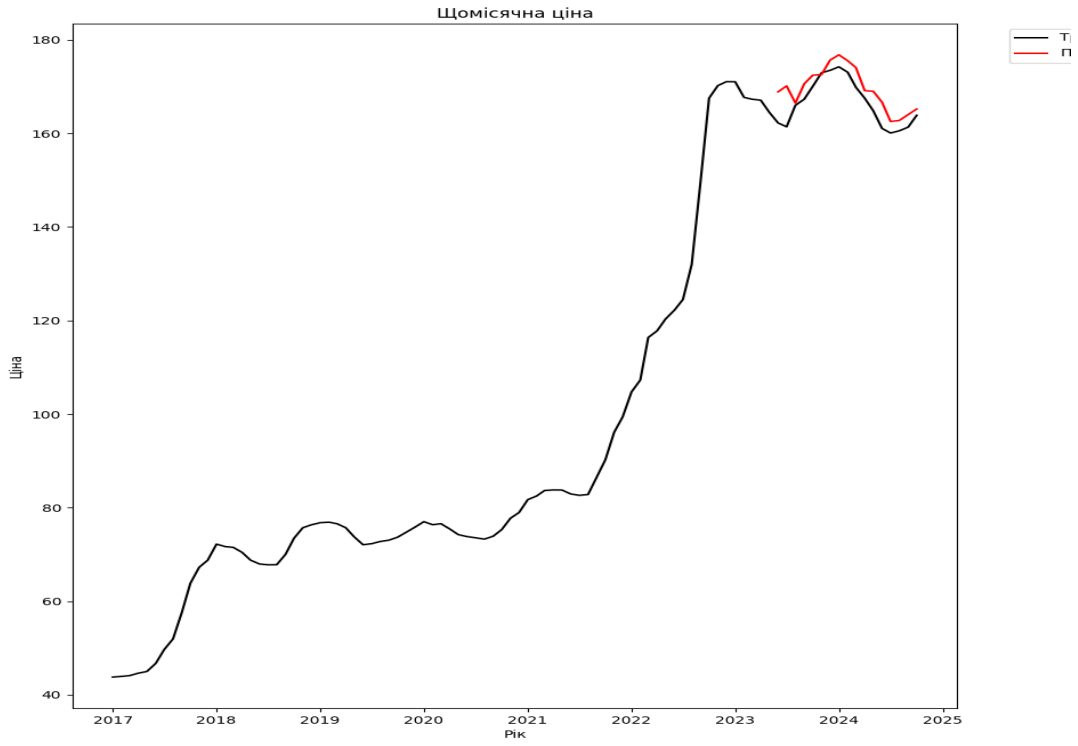



Рис. 3.17 Результат прогнозування вартості сала

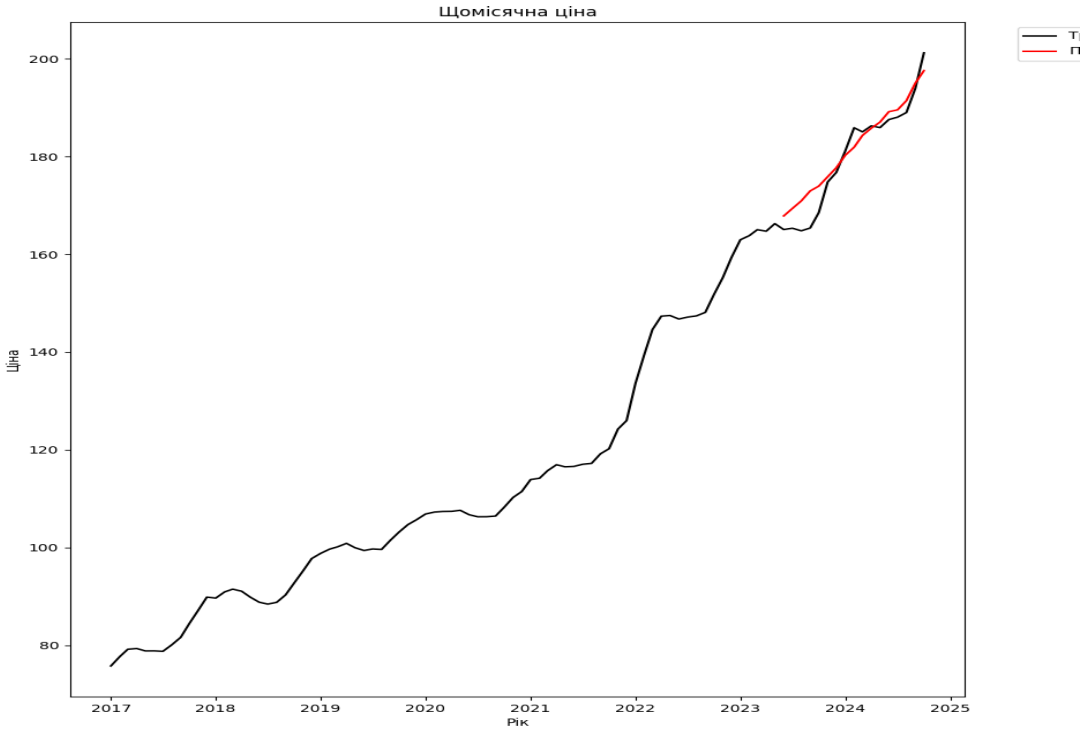



Рис. 3.18 Результат прогнозування вартості твердого сиру

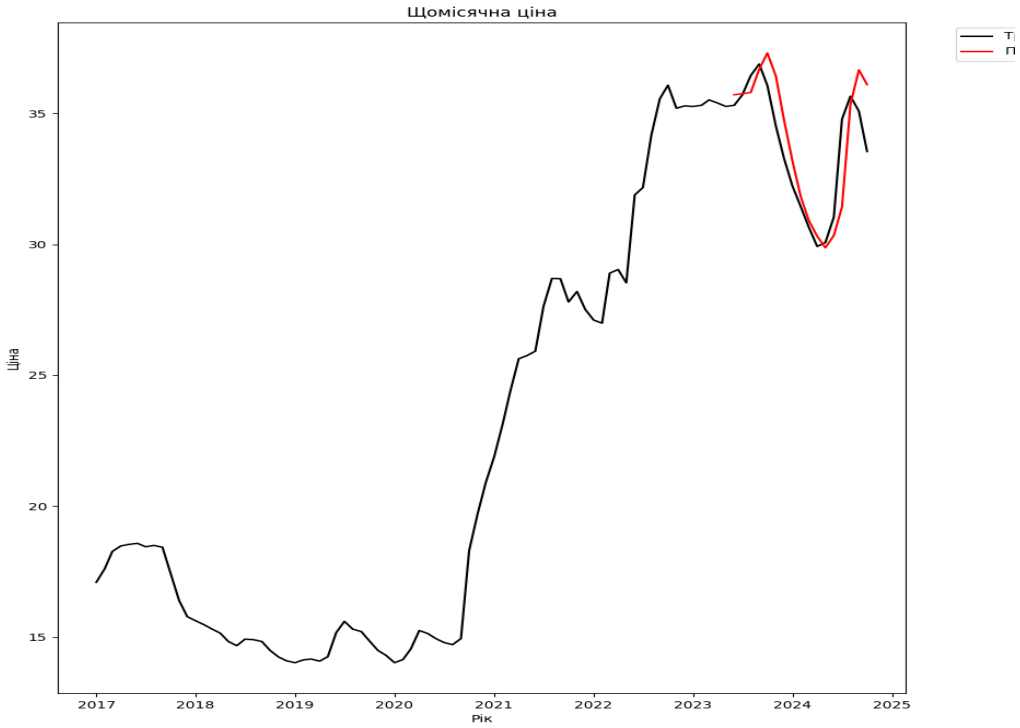



Рис. 3.19 Результат прогнозування вартості цукру

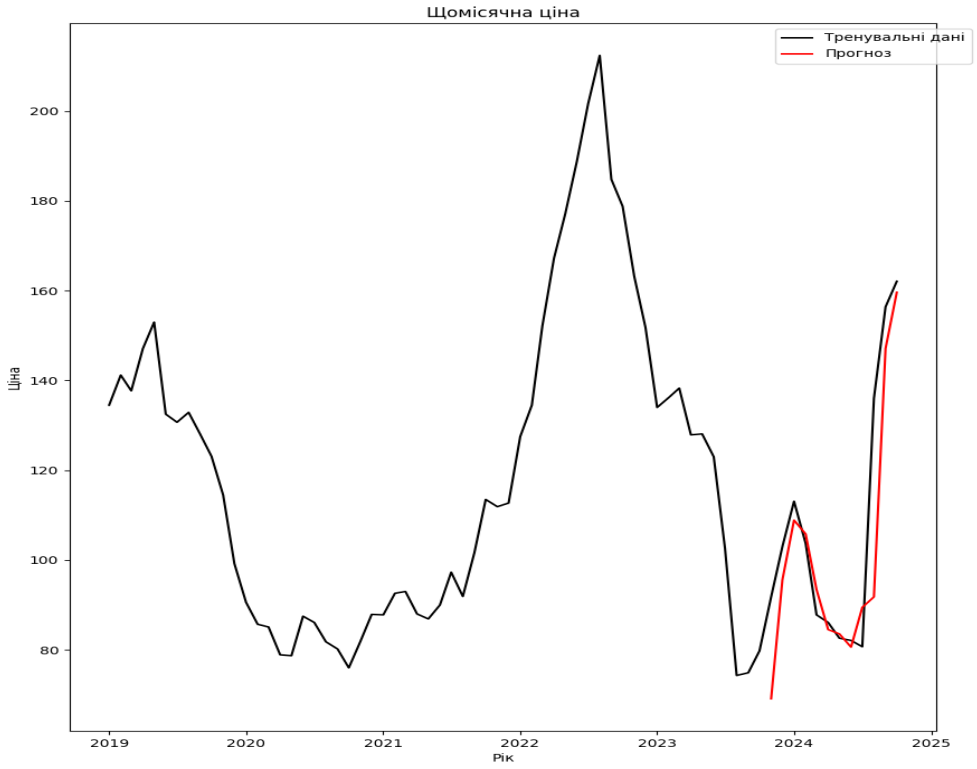



Рис. 3.20 Результат прогнозування вартості овочей

# ВИСНОВКИ

В процесі виконання магістерської роботи, що включала в себе дослідження пов’язані з темою роботи, а саме область передбачення за допомогою нейронних мереж, та економіки продуктової корзини було досягнуто наступних результатів:

1. Проведений детальний огляд існуючих наукових праць та методів у галузі прогнозування економічних показників, зокрема вартості продуктового кошика. Проаналізовано підходи, що використовуються для обробки та аналізу цінових даних, зокрема методи машинного навчання, регресійного аналізу, часових рядів та їхніх гібридних модифікацій. Виявлено переваги та недоліки кожного з методів. Особливу увагу було приділено підходам, які враховують сезонність та коливання ринкових цін, а також інтеграції додаткових факторів, таких як інфляція, зміни попиту та пропозиції. Це дослідження стало базою для створення власного підходу, який враховує ключові аспекти виявлених недоліків існуючих моделей.

2. Розроблено математичну модель для передбачення вартості продуктового кошика. У моделі враховано специфіку роботи з реальними економічними даними, зокрема їхню сезонність, кореляцію між різними товарами та можливі тенденції зростання чи зниження цін у часі. Для реалізації моделі було використано сучасні інструменти програмування, такі як Python із бібліотеками pandas, NumPy, scikit-learn та TensorFlow. Особливу увагу приділено параметрам, які впливають на точність прогнозу, зокрема вибору кількості прихованих шарів, функції активації.

3. Точність моделі перевірено за допомогою за допомогою коефіцієнта детермінації $R^2$.Аналіз показав, що модель ефективно прогнозує вартості продуктової корзини, але для деяких специфічних товарів із високою волатильністю (наприклад, сезонних фруктів або овочів) результати могли мати підвищену похибку. Це стало основою для подальших досліджень, спрямованих на покращення адаптивності моделі до змінних ринкових умов.

4. На основі аналізу недоліків існуючих підходів було запропоновано вдосконалений метод передбачення. Метод було протестовано на реальних даних, що підтвердило його здатність адаптуватися до складних ринкових умов.

Результати роботи підтверджують, що використання нейронних мереж для прогнозування вартості продуктового кошика є ефективним і перспективним. Розроблений метод дозволяє враховувати широкий спектр факторів, що забезпечує високу точність прогнозування. Експериментальні результати показали переваги запропонованого підходу порівняно з традиційними методами, а саме значне підвищення точності прогнозування. Реалізоване програмне забезпечення демонструє високу продуктивність, простоту інтеграції та потенціал для практичного використання у бізнесі та домогосподарствах. Окрім цього, запропонована методика є універсальною, що дозволяє її адаптувати до інших завдань прогнозування. Метод має певні обмеження, зокрема високу залежність точності від якості та обсягу вхідних даних. Деякі фактори, такі як політична нестабільність або глобальні кризи, важко точно врахувати через їх непередбачуваність. Розроблена модель потребує значних обчислювальних ресурсів для навчання, що може бути проблемою для малих організацій чи домогосподарств. Окрім цього, підхід із використанням нейронних мереж є складним для інтерпретації, що може ускладнити пояснення отриманих результатів для кінцевих користувачів.

# ПЕРЕЛІК ПОСИЛАНЬ


1. Споживчий кошик українця у 2024 – LVIV.MEDIA. LVIV.MEDIA. URL: https://lviv.media/ekonomika/88843-spozhivchij-koshik-ukrayincya-u-2024/ (дата звернення: 03.12.2024).
2. Фактори (області) зовнішнього середовища підприємства - бібліотека buklib.net. Головна - Бібліотека BukLib.net. URL: https://buklib.net/books/36593/#:~:text=1.,%20товарів,%20інформації%20та%20енергії (дата звернення: 03.12.2024).
3. Ecologymanual. *ecologymanual*. URL: https://ecologyknu.wixsite.com/ecologymanual/blank-5 (дата звернення: 08.12.2024).
4. PESTLE – аналіз як інструмент стратегічного аналізу ОТГ. https://hromada.canactions.com/pest/. URL: https://hromada.canactions.com/ (дата звернення: 03.12.2024).
5. Індекс інфляції (індекс споживчих цін) в Україні. «Дебет-Кредит» - Сервіси для бухгалтера. URL: https://services.dtkt.ua/catalogues/indexes/3-indeks-infliaciyi-indeks-spozivcix-cin-v-ukrayini (дата звернення: 03.12.2024).
6. Державна служба статистики. URL: https://www.ukrstat.gov.ua/ (дата звернення: 03.12.2024).
7. Статистична інформація. Міністерство енергетики України. URL: https://mev.gov.ua/taxonomy/term/111 (дата звернення: 03.12.2024).
8. Служба новостей. Ранні зернові — урожай 2020: очікування vs реальність. Куркуль – онлайн-асистент фермера. URL: https://kurkul.com/spetsproekty/866-ranni-zernovi--urojay-2020-ochikuvannya-vs-realnist (дата звернення: 03.12.2024).
9. Поле онлайн. Міністерство аграрної політики та продовольства України. URL: https://minagro.gov.ua/map (дата звернення: 03.12.2024).
10. Як нейромережі генерують та розпізнають картинки? Найкращі II генератори артів. KLONA. URL: https://klona.ua/uk/blog/artificial-intelligence-uk/yak-nejromerezhi-generuyut-ta-rozpiznayut-kartynky-najkrashhi-ii-generatory-

artiv#:~:text=Нейронні%20мережі%20навчають%20комп'ютер,мережі%20(ANN)%20та%20інші (дата звернення: 03.12.2024).

11. Довідник по Machine Learning – Multilayer Perceptron | База знань IT. База знань IT технологій. URL: https://itwiki.dev/data-science/ml-reference/ml-glossary/multilayer-perceptron (дата звернення: 03.12.2024).
12. Бондаренко Сергій. Що таке лінійна регресія: повний гайд від robot_dreams. robot_dreams - онлайн-курси для фахівців у сфері big data, machine learning, data science | Робот Дрімс. URL: https://robotdreams.cc/uk/blog/437-shcho-take-liniyna-regresiya (дата звернення: 03.12.2024).
13. Label Encoding in Python - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/ml-label-encoding-of-datasets-in-python/ (date of access: 03.12.2024).
14. Що таке дерево рішень?. Unite.AI. URL: https://www.unite.ai/uk/що-таке-дерево-рішень/ (дата звернення: 03.12.2024).
15. Система електронного забезпечення навчання ЗНУ. URL: https://moodle.znu.edu.ua/pluginfile.php?file=/486136/mod_resource/content/1/Лекція%209.pdf (дата звернення: 03.12.2024).
16. Довідник по Machine Learning – Random Forests | База знань IT. База знань IT технологій. URL: https://itwiki.dev/data-science/ml-reference/ml-glossary/random-forests (дата звернення: 03.12.2024).
17. Convolutional Neural Network (CNN) in Machine Learning - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/convolutional-neural-network-cnn-in-machine-learning/ (date of access: 03.12.2024).
18. 7.6. Convolutional Neural Networks (LeNet) – Dive into Deep Learning 1.0.3 documentation. Dive into Deep Learning – Dive into Deep Learning 1.0.3 documentation. URL: https://d2l.ai/chapter_convolutional-neural-networks/lenet.html (дата звернення: 03.12.2024).
19. Bangar S. AlexNet Architecture Explained. Medium. URL: https://medium.com/@siddheshb008/alexnet-architecture-explained-b6240c528bd5 (дата звернення: 03.12.2024).

20. Deep Residual Networks (ResNet, ResNet-50) - 2024 Guide - viso.ai. viso.ai. URL: https://viso.ai/deep-learning/resnet-residual-neural-network/ (дата звернення: 03.12.2024).
21. googlenet. MathWorks - Maker of MATLAB and Simulink - MATLAB & Simulink. URL: https://www.mathworks.com/help/deeplearning/ref/googlenet.html (дата звернення: 03.12.2024).
22. Keras documentation: MobileNet, MobileNetV2, and MobileNetV3. Keras: Deep Learning for humans. URL: https://keras.io/api/applications/mobilenet/ (дата звернення: 03.12.2024).
23. Very Deep Convolutional Networks (VGG) Essential Guide - viso.ai. viso.ai. URL: https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/ (дата звернення: 03.12.2024).
24. Deep Learning for time series prediction: Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM). Ciencia de datos, teoría y ejemplos prácticos en R y Python. URL: https://cienciadedatos.net/documentos/py54-forecasting-with-deep-learning (дата звернення: 03.12.2024).
25. Project Jupyter. Project Jupyter | Home. URL: https://jupyter.org/ (дата звернення: 03.12.2024).
26. Pandas - Python Data Analysis Library. pandas - Python Data Analysis Library. URL: https://pandas.pydata.org/ (дата звернення: 03.12.2024).
27. NumPy -. NumPy -. URL: https://numpy.org/ (дата звернення: 03.12.2024).
28. Matplotlib – Visualization with Python. Matplotlib – Visualization with Python. URL: https://matplotlib.org/ (дата звернення: 03.12.2024).
29. Scikit-learn: machine learning in Python – scikit-learn 1.5.2 documentation. scikit-learn: machine learning in Python – scikit-learn 0.16.1 documentation. URL: https://scikit-learn.org/stable/ (дата звернення: 03.12.2024).
30. IBM. What is a Recurrent Neural Network (RNN)? | IBM. IBM - United States. URL: https://www.ibm.com/topics/recurrent-neural-networks (дата звернення: 03.12.2024).

# ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО - КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Метод прогнозування вартості продуктової корзини за допомогою нейронних мереж

Виконав: студент групи ПДМ -63 Кутняк Максим Юрійович

Керівник: к.т.н., доцент кафедри ІПЗ Яскевич В.О.

Київ - 2025

### ЗАВДАННЯ НА МАГІСТЕРСЬКУ РОБОТУ

1. Провести детальний огляд існуючих наукових праць та методів у галузі прогнозування економічних показників, зокрема вартості продуктового кошика. Проаналізувати підходи, що використовуються для обробки та аналізу цінових даних, зокрема методи машинного навчання, регресійного аналізу, часових рядів та їхніх гібридних модифікацій. Виявлено переваги та недоліки кожного з методів.
2. Розробити математичну модель для передбачення вартості продуктового кошика. У моделі враховано специфіку роботи з реальними економічними даними,.
3. Оцінити точність моделі.
4. На основі аналізу недоліків існуючих підходів було запропонувати вдосконалений метод передбачення та протестувати метод на реальних даних.

## ПОРІВНЯЛЬНИЙ АНАЛІЗ

| Критерії | Модифікований MLP (Розроблений метод) | Лінійна регресія | Дерево рішень | Рандомні ліси | Конволюційні нейронні мережі | Рекурентні нейронні мережі |
|---|---|---|---|---|---|---|
| Складність моделювання | Підтримує складні нелінійні залежності | Лише лінійні залежності. | Нелінійні залежності, простота. | Добре працює на складних табличних даних. | Виділення просторових ознак. | Робота з послідовностями (часові ряди, текст). |
| Обсяг даних | Досягаються краще результати з великими наборами даних. | Добре для малих наборів. | Добре для середніх наборів. | Працює добре з великими даними. | Потребує великих наборів даних для навчання. | Потребує великих послідовностей. |
| Обчислювальна складність | Висока (GPU/CPU для навчання). | Низька. | Низька. | Середня (в залежності від реалізації). | Висока (GPU для оптимальної роботи). | Висока (GPU потрібні для ефективності). |
| Чутливість до даних | Потребує нормалізації та стандартизації даних. | Потребує нормалізації. | Не потребує нормалізації | Стійкий до пропусків та шуму. | Потребує попередньої обробки. | Чутливий до довжини та якості послідовностей. |

3

## ПОСЛІДОВНІСТЬ ДІЙ РОБОТИ МЕТОДУ

1. Завантаження датасету з даними про ціну продуктів.
2. Розбиття початкових даних на 3 ряди: тренд, сезонність та залишки.
3. Перетворення даних для роботи в програмній середі.
4. Нормалізація даних.
5. Визначення кількості скритих шарів для MLP.
6. Застосування MLP для навчання нейронної мережі для передбачення вихідного результату.
7. Порівняння вихідного результату з останніми 20 % вхідних даних з метою оцінки точності передбачення.

4

## МАТЕМАТИЧНА МОДЕЛЬ

1. Вхідний шар

Модель отримує вектор вхідних даних :

$x = (x_1, x_2, \dots, x_n)$,

де n — кількість вхідних змінних

2. Приховані шари

У прихованих шарах нейронна мережа застосовує лінійні перетворення з подальшою нелінійною активацією. Кожен нейрон прихованого шару виконує такі операції :

- Лінійне перетворення :

$z_j = w_j^T a_{j-1} + b_j$,

де $w_j^T$ - вектор ваг для нейронів у шарі $j$;

$a_{j-1}$ - активація з попереднього шару;

$b_j$ - зміщення.

Нелінійна активація:

В розробленому методі застосовується функція активації ї ReLU (Rectified Linear Unit) $f(x) = \max(0, x)$.

2. Вихідний шар

$y = w_{out}^T a_{out} + b_{out}$,

де $w_{out}^T$ - вектор ваг для нейронів у останньому шарі;

$a_{out}$ - вектор активацій останнього прихованого шару;

$b_{out}$ - зміщення .



## МАТЕМАТИЧНА МОДЕЛЬ

Оцінка точності

Точність передбачення моделі було обчислено за допомогою коефіцієнта детермінації $R^2$, який обчислюється за формулою:

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

де $y_i$ – справжнє значення;

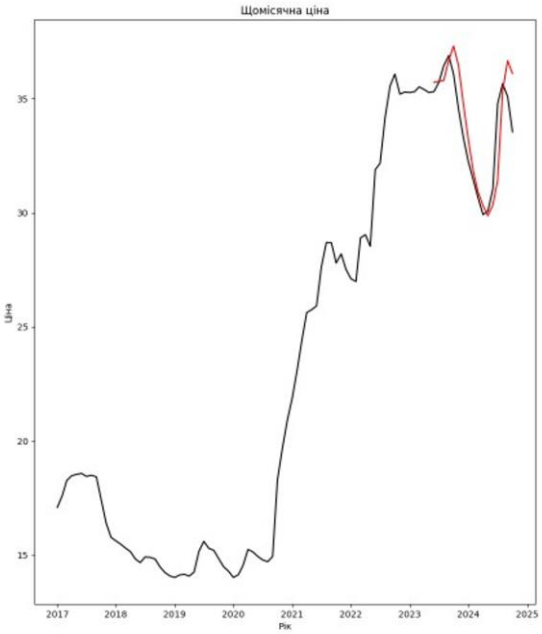
$\hat{y}_i$ – передбачене значення;

$\bar{y}$ – середнє значення $y$.

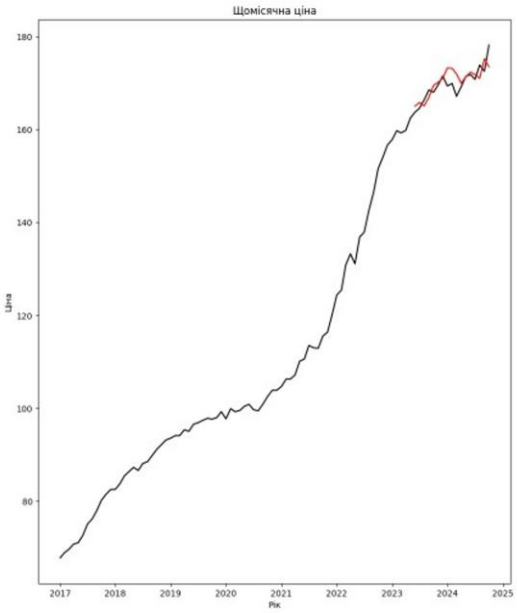
Результат може варіюватись 0 до 1, де 0 - модель не пояснює варіацію даних краще, ніж середнє значення, 1 - модель ідеально передбачає дані.

## РЕЗУЛЬТАТ ПРОГНОЗУ

Цукор




Ковбаса




7

## ОЦІНКА ТОЧНОСТІ

| | |
|---|---|
| яйця | 0,947513086 |
| хліб | 0,939278868 |
| твердий сир | 0,968023505 |
| риба | 0,9874993 |
| фрукти | 0,906683275 |
| ковбаса | 0,993955902 |
| картопля | 0,984491024 |
| рис | 0,987714653 |
| соняшникова олія | 0,986037923 |
| молоко | 0,972650709 |
| вершкове масло | 0,931611505 |
| макаронні вироби | 0,95910891 |
| м’ясо | 0,932001145 |
| цукор | 0,990220459 |
| сало | 0,992326033 |
| овочі | 0,97009841 |
| Середня точність | 0,965576 |

8

## ПОРІВНЯННЯ ТОЧНОСТІ

| Методи прогнозування | Середня точність |
|---|---|
| Розроблений метод | 0,965576 |
| Конволюційні нейронні мережі | 0,945869 |
| Рекурентні нейронні мережі | 0,958425 |
| Лінійна регресія | 0,252103 |



## ВИСНОВКИ

1. Проведено аналіз існуючих наукових джерел, для того щоб отримати використати отриманні знання для побудови власного методу передбачення вартості продуктової корзини.
2. Розроблено модель для передбачення цін продуктової корзини, та реалізовано програмний продукт який пройшов навчання та тестування на даних.
3. Було розроблено новий метод для покращення точності передбачення вартості продуктового кошику.
4. Проведено аналіз точності методу передбачення на реальних даних, використовуючи обрану метрику.

Розроблена система прогнозування вартості продуктового кошика поєднує в собі ефективність сучасних методів машинного навчання та гнучкість до зміни ринкових умов. Це забезпечує точне передбачення цін і створює основу для подальшого вдосконалення алгоритмів у контексті економічного прогнозування .

## ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ

**Тези доповідей:**

1. Кутняк М. Ю., Яскевич В. О. Аналіз алгоритму «Зворотного поширення помилки» для навчання нейронних мереж // Всеукраїнська науково-технічна конференція "Виклики та рішення в програмній інженерії ". (подано до друку)
2. Кутняк М.Ю., Яскевич В. О. Порівняльний аналіз популярних фреймворків Java // Всеукраїнська науково-технічна конференція "Виклики та рішення в програмній інженерії ". (подано до друку)

# ДОДАТОК Б. ЛІСТИНГ ОСНОВНОГО МОДУЛЮ

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from matplotlib.dates import DateFormatter
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import TimeSeriesSplit
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import RandomizedSearchCV
import warnings
warnings.filterwarnings('ignore')
df = pd.read_csv('bread.csv', parse_dates=['Month'], index_col=['Month'])
fig, ax = plt.subplots()
plt.xlabel('Рік')
plt.ylabel('Ціна')
plt.title('Щомісячна ціна')
plt.plot(df, color='black')
ax.xaxis.set_major_formatter(DateFormatter('%Y'))
# Set the figure size
plt.rcParams['figure.figsize'] = [10, 12]

from statsmodels.tsa.seasonal import seasonal_decompose
# Plot the decomposition components
sd = seasonal_decompose(df).plot()
k = 12
Z = []
for i in range(k + 1, df.shape[0] + 1):Z.append(df.iloc[(i - k - 1): i, 0])
Z = np.array(Z)
Z.shape
print(Z.shape)
split = int(0.8 * Z.shape[0])
print(split)
Z_train, Z_test = Z[:split, :], Z[split:, :]
X_train, y_train = Z_train[:, :-1], Z_train[:, -1]
X_test, y_test = Z_test[:, :-1], Z_test[:, -1]
param_distributions = {
'hidden_layer_sizes': [(np.random.randint(1, 100),) for _ in range(10)]
}
model = MLPRegressor(max_iter=1000, random_state=1)
```

```
random_search = RandomizedSearchCV(estimator=model, param_distributions=param_distributions, n_iter=10, cv=5, scoring='neg_mean_squared_error')
random_search.fit(X_train, y_train)
print("Best hidden_layer_sizes:", random_search.best_params_)
mlp = MLPRegressor(hidden_layer_sizes=random_search.best_params_['hidden_layer_sizes'], max_iter=1000, random_state=1)
mlp.fit(X_train, y_train)
print(mlp.score(X_train, y_train))
y_pred = mlp.predict(X_test)
fig, ax = plt.subplots()
plt.xlabel('Рік')
plt.ylabel('Ціна')
plt.title('Щомісячна ціна')

plt.plot(df, color='black', label='Тренувальні дані')
plt.plot(pd.Series(y_pred, index=df.index[-len(y_test):]), color='red', label='Прогноз')
plt.legend(bbox_to_anchor=(1.05, 1))
ax.xaxis.set_major_formatter(DateFormatter('%Y'))
plt.savefig('a.png')
```