

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Державний вищий навчальний заклад

“Ужгородський національний університет”

Факультет інформаційних технологій

Кафедра інформаційних управляючих системи та технології

ЗВІТ

з переддипломної практики

Реєстраційний № _____

Дата _____

Параска Богдан Володимирович

студент 4 курсу

денної форми навчання

залікова книжка № _____

ЗВІТ ПРО ПРОХОДЖЕННЯ ПЕРЕДДИПЛОМНОЇ ПРАКТИКИ

Спеціальність "Комп'ютерні науки (Інформатика)"

Рекомендовано до захисту з оцінкою:

" _____ " _____

Підпис керівника практики

" _____ " _____ 2025 р.

Керівник

практики від ВНЗ

доктор технічних наук,

професор Міца О. В.

Ужгород – 2025

ЗМІСТ

ВСТУП.....	3
РОЗДІЛ 1. ВІДОМОСТІ ПРО БАЗУ ПРАКТИКИ.....	4
1.1 Відомості про ТОВ «ЦБО СКАЙТЕК».....	4
РОЗДІЛ 2. ТЕОРЕТИЧНІ ЗАВДАННЯ	6
2.1 Поняття штучних нейронних мереж.....	6
2.2 Основи машинного навчання як фундаменту для нейромереж	8
2.3 Типи архітектур нейронних мереж у задачах прогнозування.....	14
РОЗДІЛ 3. ПРАКТИЧНЕ ЗАВДАННЯ	18
3.1. Вибір програмних засобів	18
3.2. Основні бібліотеки та фреймворки	19
3.3 Підготовка даних для навчання моделей.....	20
3.4 Створення моделей	22
3.5 Реалізація моделей і процес навчання	25
3.6 Аналіз результатів моделей	26
ВИСНОВКИ	27
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ ТА ДЖЕРЕЛ	28
ДОДАТКИ.....

ВСТУП

Прогнозування цін є одним із ключових завдань у фінансовій аналітиці, торгівлі та економіці загалом. У сучасних умовах, коли обсяг і складність даних зростають, все більшого значення набувають методи штучного інтелекту, зокрема нейронні мережі. Вони демонструють високу здатність до виявлення складних залежностей у часових рядах і дозволяють досягати точніших результатів у порівнянні з класичними статистичними методами.

Метою цього проєкту є дослідження ефективності різних архітектур нейронних мереж у задачах прогнозування цін (на прикладі криптовалют Bitcoin та Ethereum). Передбачається реалізація, тестування та порівняння таких моделей, як багат шаровий перцептрон (MLP), рекурентна нейронна мережа LSTM, згорткова нейронна мережа (CNN) та трансформери. Для досягнення цієї мети заплановано виконання наступних завдань:

Аналіз вимог до дипломного проєкту: буде сформульовано чіткі цілі дослідження, обсяг задач і обґрунтовано вибір теми з урахуванням актуальності прогнозування цін у сучасному цифровому середовищі.

Огляд існуючих підходів та технологій: проведено аналіз сучасних методів машинного навчання, що застосовуються для аналізу часових рядів, зокрема в контексті прогнозування фінансових показників. Обрані методи буде обґрунтовано на основі їх точності, складності реалізації та практичної придатності.

Підготовка даних та реалізація моделей: виконано обробку часових рядів, побудовано архітектури обраних моделей у середовищі Python з використанням бібліотек PyTorch, scikit-learn та інших.

Навчання та тестування моделей: буде проведено тренування моделей на історичних даних із подальшим тестуванням їхньої точності, стійкості та здатності до генералізації.

Візуалізація та порівняльний аналіз результатів: створено графіки реальних і прогнозованих значень, виконано оцінювання ефективності моделей за метриками RMSE, MAE та R^2 .

РОЗДІЛ 1. ВІДОМОСТІ ПРО БАЗУ ПРАКТИКИ

1.1 Відомості про ТОВ «ЦБО СКАЙТЕК»

Базою проходження практики є Центр бізнес-обслуговування «СкайТек» – компанія, що надає широкий спектр ІТ-послуг як для бізнесу, так і для приватних клієнтів. Основні напрямки діяльності центру охоплюють комп'ютерне обслуговування, автоматизацію бізнес-процесів, розробку веб-сайтів, бухгалтерський супровід та створення програмного забезпечення.

«СкайТек» відзначається гнучким підходом до клієнтів, індивідуальними консультаціями, а також можливістю виїзду спеціаліста або віддаленого усунення технічних проблем. Працівники компанії мають глибокі знання у сфері електронного документообігу, касового обладнання та інтеграції різноманітних систем обліку.


 <p>КОМП'ЮТЕРНЕ ОБСЛУГОВУВАННЯ</p> <p>Потрібно перевірити сервер? Хочете зробити ремонт комп'ютера? Потрібне обслуговування комп'ютерної мережі? Ми пропонуємо виконати ці та інші роботи якісно і швидко.</p>	 <p>РОЗРОБКА ВЕБ САЙТІВ</p> <p>Компанія "Sky Tek" пропонує Вам свої послуги у сфері розробки веб-сайту, який сповна задовольнить Ваші потреби. Чи це буде просто сайт-візитка, чи бізнес-сайт різної складності або ж Інтернет-магазин – наші фахівці докладуть максимальні зусилля, щоб Ви могли пишатися Інтернет-обличчям Вашої компанії.</p>	 <p>POS ОБЛАДНАННЯ ТА АВТОМАТИЗАЦІЯ</p> <p>Завдяки автоматизації торгівлі, облік Вашого товару стане легшим і точнішим, Вам буде легко контролювати Свій бізнес. Ми завжди готові запропонувати найбільш ефективне рішення для Вашої справи!</p> <p>Для автоматизації Вашого бізнесу ми пропонуємо наступне обладнання.</p>
 <p>БУХГАЛТЕРСЬКІ ПОСЛУГИ</p> <p>Ми пропонуємо бухгалтерські послуги організаціям різних форм власності та видів діяльності. - ведення бухгалтерського обліку - податковий облік та бухгалтерія для бізнесу різних масштабів - спрощена бухгалтерія та облік приватних підприємців - оперативне вирішення податкових питань.</p>	 <p>РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ</p> <p>Програмне забезпечення на замовлення - це можливість отримати програмний продукт, що точно відповідає потребам компанії, який здатний підвищити її конкурентоспроможність, підвищити ефективність роботи і забезпечити подальший ріст і розвиток.</p>	 <p>СИСТЕМИ ОБЛІКУ БІЗНЕСУ</p> <p>ІТ компанія "SKY TEK" спеціалізується на розробці та впровадженні рішень по автоматизації бізнесу і процесів обліку. Досвід та знання наших фахівців дозволяють інтегрувати обладнання та програмне забезпечення в єдину комплексну систему обліку та контролю.</p>

Рис 1.1 Основні послуги компанії

Центр бізнес-обслуговування "СкайТек" – це компанія, що стрімко розвивається, активно розширюючи свою діяльність у декількох ключових секторах ІТ-індустрії. Вона поєднує в собі різноманітні напрями роботи: від технічного обслуговування комп'ютерів до розробки складних програмних продуктів та реалізації інноваційних рішень для автоматизації бізнес-процесів. Це робить "СкайТек" не просто сервісною компанією, а й надійним ІТ-партнером для малого, середнього та великого бізнесу. Унікальність полягає в наданні повного спектру послуг: від первинної консультації та аналізу потреб клієнта до розробки, впровадження та технічної підтримки ІТ-продуктів або рішень.

Для студентів, які проходять практику в "СкайТек", фірма створює сприятливі умови для отримання практичного досвіду у професійній сфері. Практиканти мають змогу не лише ознайомитися з реаліями корпоративної інфраструктури, але й брати безпосередню участь у вирішенні практичних завдань: налаштуванні та обслуговуванні техніки, роботі над проєктами з розробки веб-сайтів, створенні або тестуванні програмного забезпечення, консультуванні клієнтів, інтеграції облікових систем та багато іншого. Цей досвід дозволяє краще зрозуміти специфіку роботи в ІТ, вдосконалити професійні навички, покращити комунікативні здібності та освоїти практичні інструменти, що використовуються в сучасному ІТ-середовищі.

Багатопрофільність компанії та її орієнтація на реальні бізнес-потреби дають практикантам можливість працювати з актуальними проєктами, вирішувати нестандартні завдання, аналізувати функціонування сучасних систем автоматизації, працювати в команді над проєктами, а також бачити весь процес роботи з клієнтом – від першого звернення до впровадження рішення. Це дозволяє сформувати комплексне розуміння роботи в інформаційних технологіях, покращити професійну підготовку та підвищити конкурентоспроможність на ринку праці після завершення навчання.

РОЗДІЛ 2. ТЕОРЕТИЧНІ ЗАВДАННЯ

2.1 Поняття штучних нейронних мереж

Штучна нейронна мережа (ШНМ) - це математична конструкція, що копіює принципи роботи біологічного мозку. Вона базується на великій кількості взаємопов'язаних вузлів, так званих "нейронів", які організовані у шари: вхідний, один чи декілька прихованих та вихідний. Така мережа здатна вчитися на прикладах, виявляючи закономірності у даних, та згодом робити прогнози або класифікувати нову інформацію.

Загалом, структура ШНМ має високу гнучкість і може бути пристосована для різноманітних задач - від розпізнавання зображень до передбачення часових рядів. Її ефективність значною мірою залежить від правильно обраної архітектури та методу навчання. Основними компонентами ШНМ є нейрони, вагові коефіцієнти, функція активації та алгоритм оптимізації. Кожен нейрон отримує сигнали від попередніх шарів, зважує їх, підсумовує та передає результат далі.

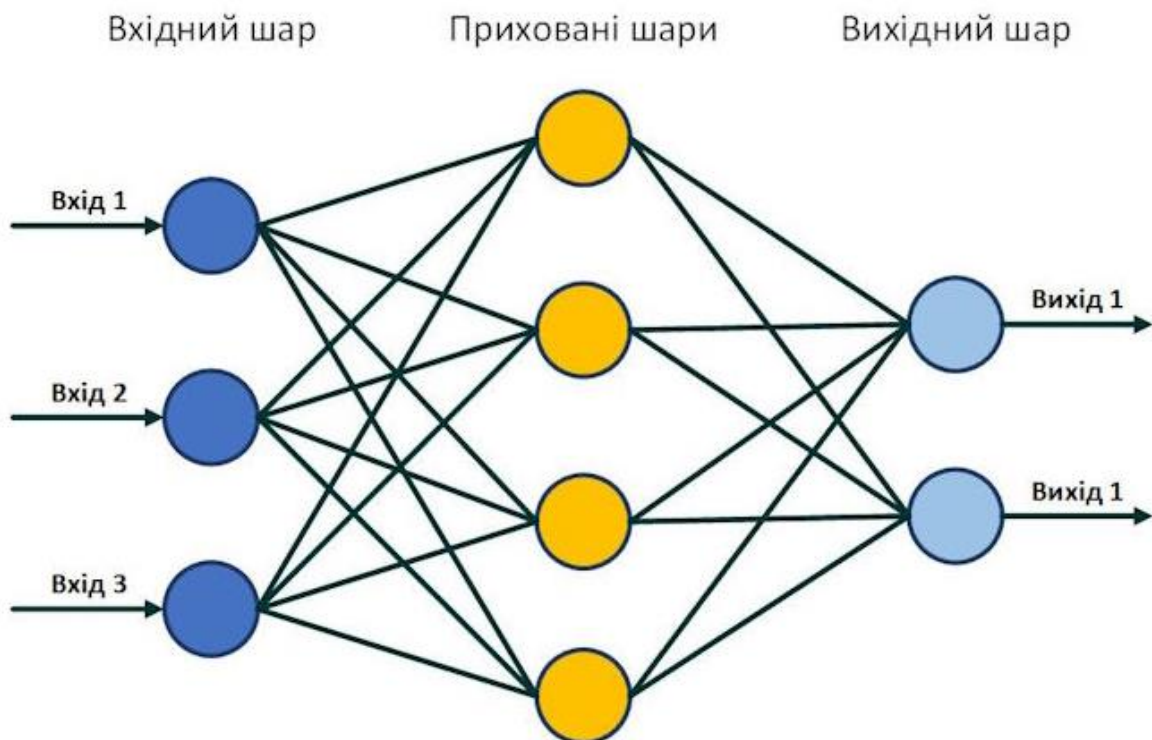


Рис. 2.1. Структура найпростішої нейронної мережі

На Рис. 2.1 представлено одношарову нейронну мережу, яка складається з одного входу, одного прихованого шару та одного виходу. Кожен зв'язок має вагу, яка змінюється в процесі навчання.

Навчання ШНМ є процесом налаштування ваг, у ході якого помилка між фактичним та очікуваним результатом мінімізується. Це може досягатися за допомогою різних методів – найпоширенішими з них є градієнтний спуск та його варіації. Під час зворотного поширення помилки (backpropagation) мережа "навчається", змінюючи ваги у напрямку зменшення помилки.

На Рис. 2.2 продемонстровано, як нейронна мережа обчислює похибку, а потім повертається шарами, змінюючи ваги для покращення точності.

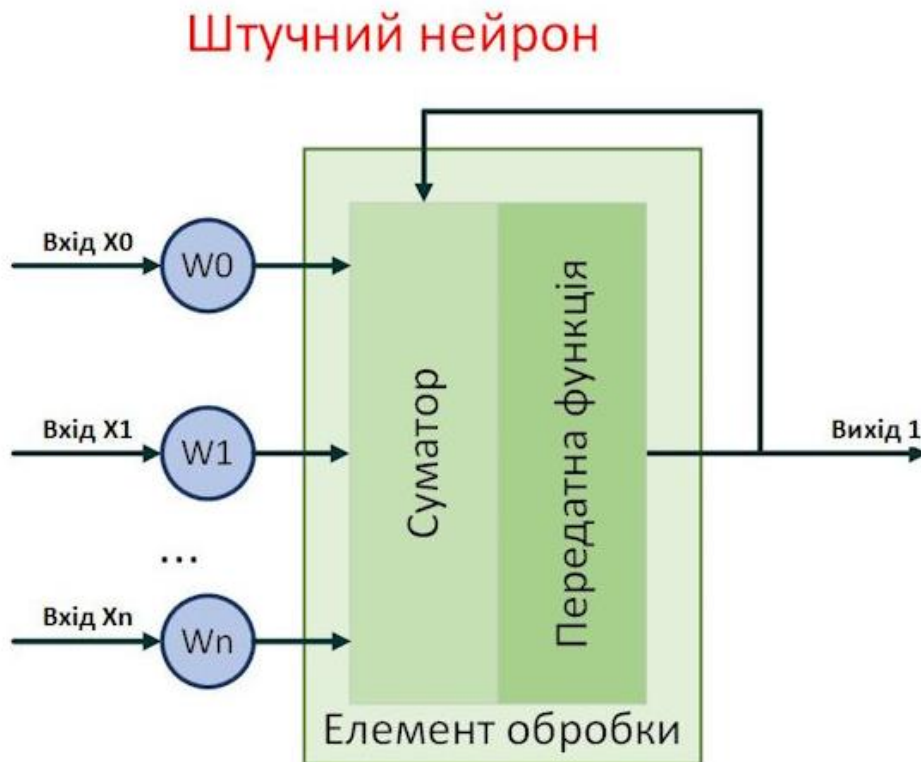


Рис. 2.2 – Схематичне зображення процесу навчання мережі за допомогою зворотного поширення помилки

Ключовим аспектом при побудові ефективної моделі є вибір архітектури нейромережі. Просте збільшення кількості шарів або нейронів не завжди приносить кращі результати – важливо уникати як недонавчання (underfitting),

так і перенавчання (overfitting). Для цього використовуються спеціальні техніки, серед яких регуляризація, дропаут або крос-валідація.

ШНМ активно застосовуються у задачах прогнозування цін через їх здатність розпізнавати складні нелінійні залежності в даних. Особливо ефективними є такі модифікації як рекурентні нейронні мережі (RNN), довготривала короткострокова пам'ять (LSTM) та трансформери.

Отже, штучні нейронні мережі – потужний інструмент в аналітиці та прогнозуванні. У наступних розділах буде розглянуто класифікацію типів ШНМ, методи їх навчання, а також їх застосування у задачах прогнозування економічних показників.

2.2 Основи машинного навчання як фундаменту для нейромереж

Машинне навчання (ML) є невід'ємною складовою штучного інтелекту та одночасно основою для побудови та навчання штучних нейронних мереж. Цей підхід дозволяє алгоритмам «вчитися» на основі даних, без явного програмування кожного кроку розв'язання задачі. У загальному сенсі, машинне навчання – це здатність комп'ютерних систем знаходити закономірності у даних і використовувати ці закономірності для прийняття рішень, прогнозів чи класифікацій.

машинне навчання визначається як: Метод аналізу даних, який автоматично будує аналітичну модель, використовуючи алгоритми, що ітеративно навчаються на даних. Це дозволяє комп'ютерам знаходити приховані інсайти без явного програмування, де шукати.

Це визначення добре підкреслює ключову перевагу ML – здатність до самостійного вдосконалення на основі досвіду.

Класифікація видів машинного навчання агалом машинне навчання поділяється на три основні типи:

- Навчання з учителем (Supervised Learning)
- Навчання без учителя (Unsupervised Learning)
- Підкріплювальне навчання (Reinforcement Learning)

На Рис. 2.3 зображено порівняння основних підходів ML. У таблиці наведено їхні характеристики, застосування та приклади.

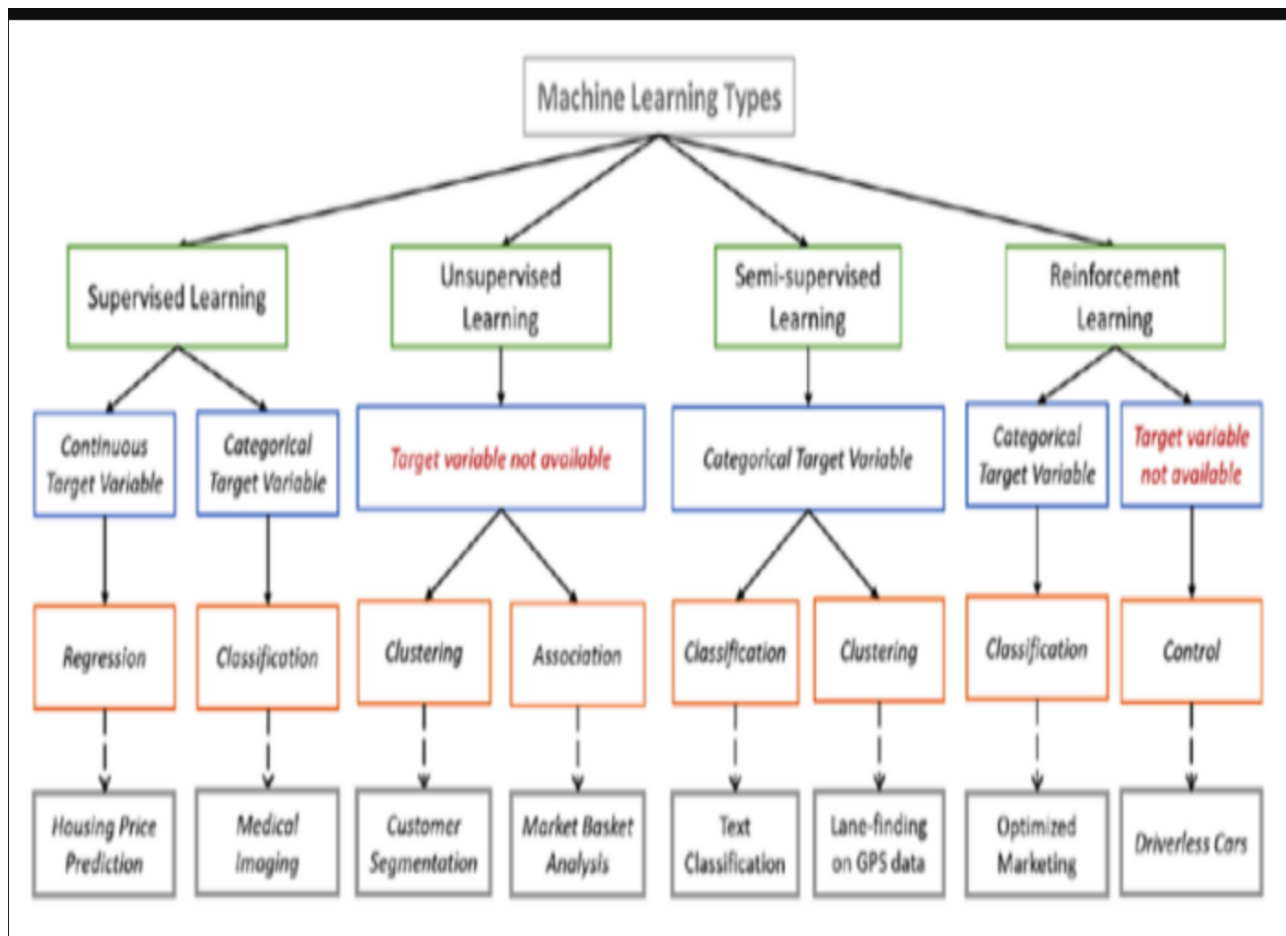


Рис. 2.3 – Схема типів машинного навчання

Навчання з учителем. Це найбільш поширений тип ML, при якому алгоритм навчається на заздалегідь маркованих даних. Іншими словами, кожному прикладу у вхідному наборі дано правильну відповідь. Система вивчає зв'язки між вхідними даними та виходами, щоб у майбутньому застосовувати ці зв'язки до нових випадків.

Приклади задач:

- Класифікація електронних листів як «спам» або «не спам»
- Прогнозування вартості нерухомості за площею, розташуванням тощо

На Рис. 2.4 показано, як система використовує позначені дані для побудови моделі.

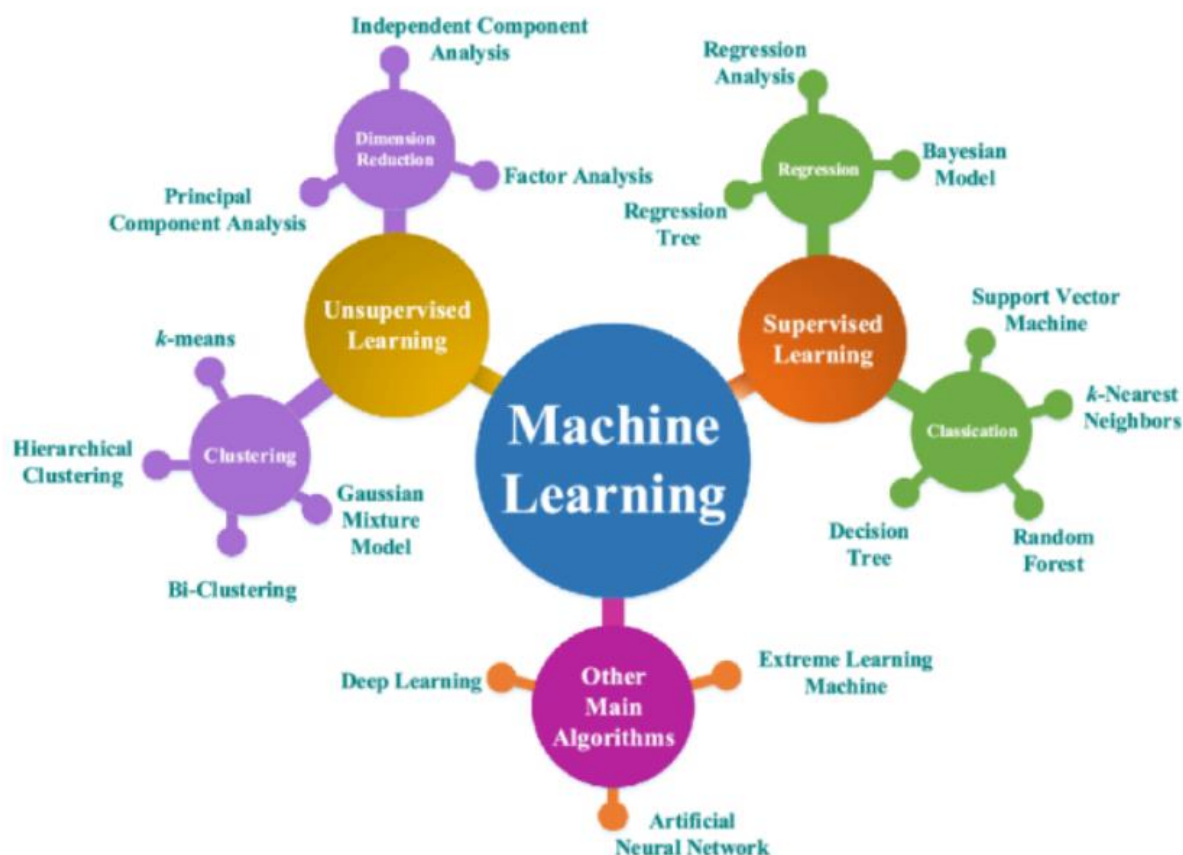


Рис. 2.4 – Схема роботи алгоритму з учителем

Навчання без наглядача (Unsupervised learning). Це різновид машинного навчання, де алгоритм опановує інформацію на основі масиву даних без маркування. Основна ціль алгоритму – самостійно виявити шаблони й віднайти структури в даних, згрупувати об'єкти за їхньою подібністю та, можливо, передбачити майбутні тенденції.

Навчання без наглядача часто використовується для розподілу даних на кластери за спільними характеристиками (кластеризація), скорочення кількості властивостей даних без втрати значущої інформації (зменшення розмірності) та виявлення нетипових даних або відхилень від загальної тенденції (виявлення аномалій).

До алгоритмів навчання без наглядача зараховують:

- Метод k-середніх - алгоритм кластеризації, який розділяє дані на наперед задану кількість груп.
- Алгоритм наближених k-середніх. Більш оптимізована версія попереднього алгоритму, призначена для обробки великих обсягів даних.
- Аналіз головних компонент. Метод зменшення розмірності, що відбирає найважливіші атрибути з масивів даних.

Також можливе поєднання двох вищезгаданих методів (з учителем та без нього). Такий гібрид, умовно, може бути позначений як "напівнаглядне навчання" (або змішане навчання).

У цьому випадку алгоритм вивчає дані, що містять як розмічені, так і не розмічені властивості. Позначені приклади служать для тренування алгоритму, як у випадку навчання з учителем, тоді як не розмічені використовуються для пошуку закономірностей у даних, подібно до навчання без наглядача.

"Напівнаглядне навчання" може показати високу ефективність у випадках, коли наявний незначний набір розмічених даних та великий обсяг не розмічених. Наприклад, під час самонавчання або взаємодії ML-моделей.

Навчання з підкріпленням (Reinforcement Learning, RL) – це потужна галузь машинного навчання, де інтелектуальний агент навчається ухвалювати оптимальні рішення шляхом безпосередньої взаємодії з динамічним довкіллям. Уявіть собі дресирування домашнього улюбленця: правильна поведінка заохочується ласощами (позитивне підкріплення), а небажана – ігнорується або викликає несхвалення (негативне підкріплення чи його відсутність). Подібним чином, в RL агент виконує певні дії у відповідь на поточний стан довкілля і отримує чисельний сигнал – "винагороду" або "штраф". Головна мета агента – не просто отримати миттєву винагороду, а розробити таку стратегію (політику) дій, яка максимізує сукупну, довгострокову винагороду. Цей процес відбувається методом проб і помилок, де агент поступово "розуміє", які дії призводять до кращих результатів у різних ситуаціях. На відміну від інших парадигм

машинного навчання, таких як навчання з учителем, де алгоритм отримує готові пари "вхід-правильний вихід", або навчання без учителя, де алгоритм шукає приховані структури в нерозмічених даних, RL агент вчиться самостійно, спираючись виключно на зворотний зв'язок від своїх дій.

Ключові переваги навчання з підкріпленням:

Навчання з підкріпленням пропонує низку унікальних переваг, що роблять його незамінним інструментом для вирішення широкого кола завдань:

- Здатність до самонавчання та автономності: Мабуть, найважливішою перевагою є те, що RL-моделі не потребують величезних, ретельно розмічених наборів даних. Агент вчиться, досліджуючи довкілля та отримуючи зворотний зв'язок. Це особливо цінно в ситуаціях, де збір або розмітка даних є надто дорогим, трудомістким або просто неможливим. Наприклад, ігровий ШІ може зіграти мільйони партій проти самого себе, поступово вдосконалюючи свою стратегію, не потребуючи даних про ігри людей-чемпіонів (хоча такі дані можуть прискорити навчання). Ще один приклад – робот, що вчиться ходити: він може починати з хаотичних рухів, але з часом, отримуючи позитивну винагороду за кожен успішний крок і негативну за падіння, він самостійно оптимізує свою ходу.

- Ефективність у надзвичайно складних задачах: RL демонструє вражаючі результати в задачах, де простір можливих рішень є гігантським, а оптимальну стратегію складно або й неможливо формалізувати за допомогою традиційних алгоритмів. Це стосується як ігрових середовищ, так і реальних застосунків. Наприклад, система AlphaGo від DeepMind, навчена за допомогою RL, змогла перемогти найкращих у світі гравців у Го – гру з астрономічною кількістю можливих ходів. Іншим прикладом є оптимізація роботи центрів обробки даних, де RL-агенти можуть керувати системами охолодження для зменшення енергоспоживання, враховуючи безліч динамічних факторів. Також RL використовується в робототехніці для навчання маніпуляторів складним діям, таким як збирання об'єктів або виконання точних операцій.

- Адаптивність та гнучкість до змін: RL-алгоритми здатні динамічно адаптуватися до змін у довкіллі або в самій задачі. Якщо умови змінюються, агент може скоригувати свою стратегію на основі нового досвіду, щоб продовжувати досягати максимальної винагороди. Це робить їх ідеальними для систем, що функціонують у непередбачуваних або еволюціонуючих середовищах. Наприклад, система управління трафіком на основі RL може адаптуватися до змін у потоках автомобілів протягом дня або до непередбачених подій, таких як дорожні роботи чи аварії, оптимізуючи роботу світлофорів для мінімізації заторів. Персоналізовані системи рекомендацій також можуть використовувати RL, щоб адаптуватися до мінливих інтересів користувача в реальному часі, пропонуючи більш релевантний контент.

Таким чином, навчання з підкріпленням відкриває шлях до створення по-справжньому інтелектуальних систем, здатних навчатися, адаптуватися та вирішувати завдання, які раніше вважалися прерогативою людського інтелекту. Його застосування охоплює все ширший спектр галузей, від ігор та робототехніки до фінансів, медицини та управління складними системами.

Етапи побудови моделі машинного навчання

Якщо узагальнити процес побудови системи машинного навчання, він складається з наступних етапів:

- Збір даних – отримання максимально повного та репрезентативного набору даних.
- Попередня обробка – очищення, нормалізація, кодування.
- Вибір моделі – визначення алгоритму навчання (дерева рішень, нейронні мережі тощо).
- Навчання моделі – власне побудова зв'язків.
- Оцінка точності – перевірка на тестових даних.
- Тестування та впровадження – реальна перевірка роботи системи.

2.3 Типи архітектур нейронних мереж у задачах прогнозування

Багатошаровий перцептрон (MLP, Multi-Layer Perceptron) є однією з найпоширеніших архітектур нейронних мереж, яка використовується в задачах класифікації, регресії та прогнозування. Основна ідея цієї моделі полягає у використанні декількох шарів нейронів, кожен з яких поєднаний із попереднім та наступним шарами через вагові коефіцієнти.

На відміну від простого перцептрона, що складається лише з вхідного та вихідного шарів, MLP має принаймні один прихований шар. Кожен нейрон у прихованому шарі застосовує активаційну функцію до зваженої суми вхідних сигналів. Це дозволяє мережі виявляти складні нелінійні залежності між вхідними і вихідними даними.

MLP працює за наступним принципом: вхідні дані подаються на перший шар, далі вони проходять через приховані шари, кожен з яких трансформує їх за допомогою активаційних функцій, і, врешті-решт, формуються результати на вихідному шарі.

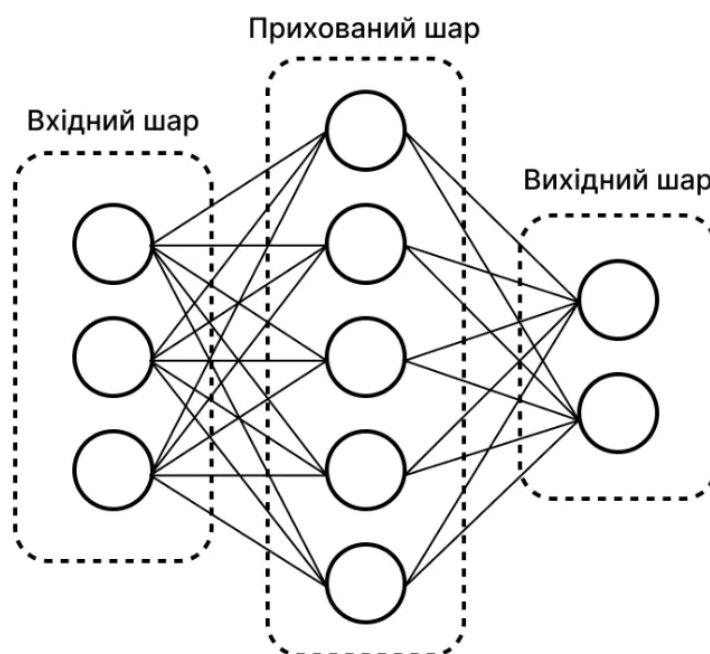


Рис. 2.5 – Схематичне зображення архітектури MLP з вхідним, прихованим і вихідним шарами.

Основні переваги MLP – це гнучкість у моделюванні даних, здатність до апроксимації довільних функцій, а також відносна простота реалізації.

MLP активно використовується у фінансовому секторі для прогнозування змін курсу валют, у медицині – для класифікації результатів діагностики, у промисловості – для прогнозування зносу деталей або якості продукції.

LSTM – Довготривала короткострокова пам'ять

Модель LSTM (Long Short-Term Memory) є різновидом рекурентних нейронних мереж (RNN), що спеціально розроблена для обробки та аналізу послідовностей даних, які мають довгострокові залежності. Звичайні RNN мають проблему з «забуванням» інформації, що була отримана багато кроків тому, а LSTM вирішує цю проблему за рахунок особливої структури своїх комірок пам'яті.

Кожен блок LSTM містить три головні компоненти: вхідний, вихідний і забувальний шлюзи. Ці шлюзи керують тим, яка інформація буде збережена, яка – передана далі, а яка – забута. Таким чином, модель має змогу ефективно зберігати важливу інформацію протягом тривалого часу і водночас відсіювати зайве.

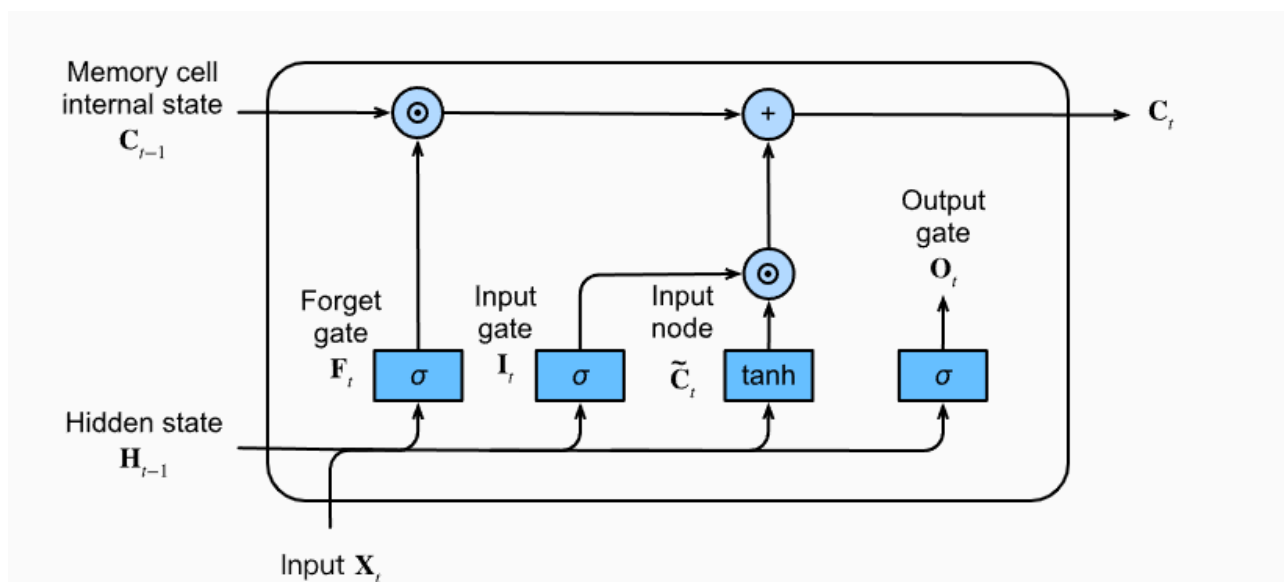


Рис. 2.6 Схематичне зображення роботи LSTM моделі

LSTM широко застосовується в обробці природної мови, зокрема для перекладу текстів, генерації мови, аналізу емоцій, а також у задачах прогнозування часових рядів – таких як прогноз попиту, цін або погодних умов.

Завдяки здатності працювати з тривалими послідовностями, LSTM значно покращує точність моделей, що мають справу зі складними часовими структурами, де врахування далекого контексту критично важливе.

Згорткова нейронна мережа (CNN)

Згорткові нейронні мережі (CNN, Convolutional Neural Networks) є архітектурою, що спеціально пристосована для обробки даних у вигляді сіток, наприклад зображень або багатовимірних тензорів. Основною операцією в CNN є згортка, яка дозволяє автоматично виявляти локальні ознаки – краї, текстури, форми тощо.

CNN складається з декількох згорткових шарів, шарів підвибірки (pooling) та повнозв'язних шарів. Кожен згортковий шар застосовує фільтри (ядра), що проходять по входу і створюють нові карти ознак, які потім узагальнюються на наступних шарах. Це дає змогу мережі навчатися складним ієрархічним представленням даних.

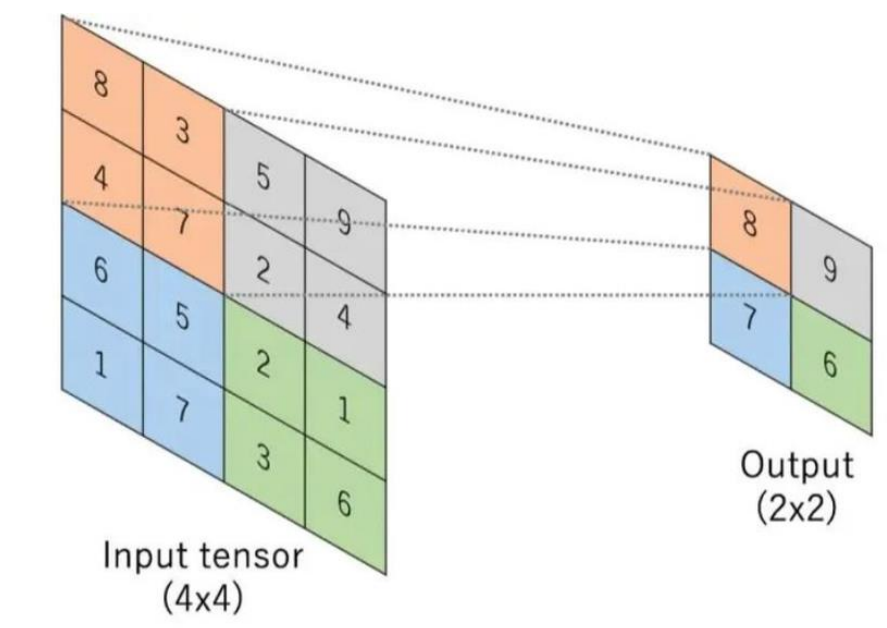


Рис. 2.7 Схема згортки та роботи з тензорами

У задачах прогнозування CNN може бути використана для обробки структурованих часових рядів, наприклад для виявлення шаблонів, що повторюються у фінансових даних або сенсорних записах. Особливо ефективна CNN у комбінації з іншими архітектурами, наприклад з LSTM.

Transformer

Архітектура Transformer є однією з найважливіших інновацій у сфері глибокого навчання останнього десятиліття. Ця модель була вперше представлена у 2017 році в роботі "Attention is All You Need" і з того часу стала основою для створення сучасних моделей, таких як BERT, GPT, T5 та багатьох інших.

Головною відмінністю Transformer від попередніх моделей є використання механізму self-attention (самоуваги), який дозволяє враховувати вагомість усіх елементів послідовності один щодо одного, незалежно від їхнього порядку. Це дозволяє моделі ефективно захоплювати контекст і взаємозв'язки між словами або значеннями в часових рядах.

У трансформерах відсутня рекурентність, що значно пришвидшує процес навчання та робить моделі добре масштабованими. Вони демонструють високу продуктивність у перекладі текстів, генерації мови, класифікації документів, а також дедалі частіше застосовуються для прогнозування часових рядів.

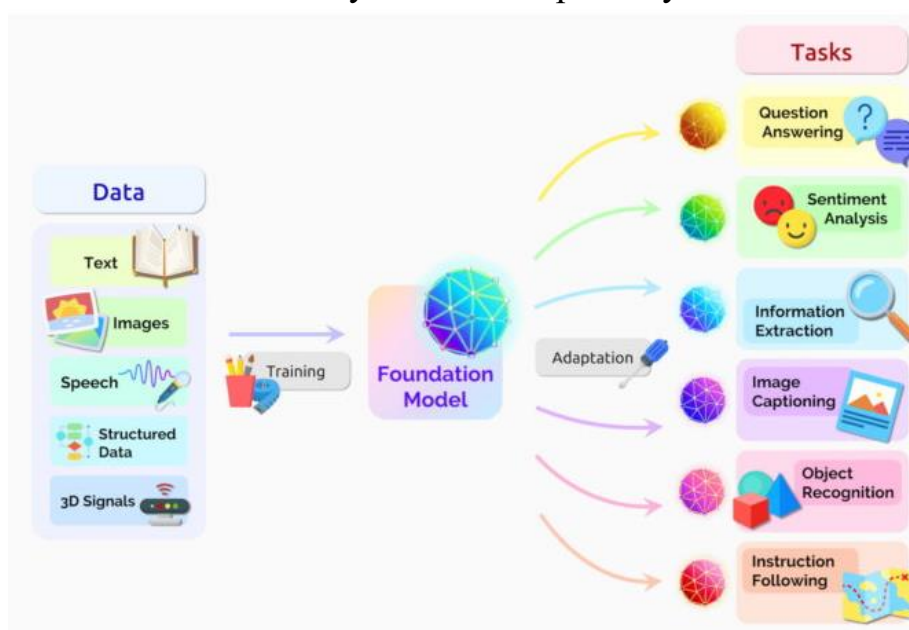


Рисунок 2.8 – Зображення можливостей моделей

РОЗДІЛ 3. ПРАКТИЧНЕ ЗАВДАННЯ

3.1. Вибір програмних засобів

Одним із ключових етапів у реалізації дипломного проєкту є визначення програмного середовища, у якому здійснюється розробка, тестування та аналіз моделей. Зважаючи на зручність, гнучкість та підтримку численних розширень, для реалізації обчислень та написання коду було обрано середовище Visual Studio Code (VS Code).

Visual Studio Code — це сучасний редактор вихідного коду, створений компанією Microsoft для операційних систем Windows, macOS та Linux. Він має відкритий вихідний код і доступний безкоштовно, що робить його привабливим вибором для студентів і розробників. Середовище підтримує велику кількість мов програмування, включаючи Python — основну мову, використану в цьому проєкті. Редактор забезпечує підсвічування синтаксису, автодоповнення коду (завдяки IntelliSense), інтеграцію з Git, налагоджувач, можливість роботи з віртуальними середовищами та зручний інтерфейс.

На відміну від традиційних IDE, VS Code дозволяє швидко створити робоче середовище за допомогою відкриття потрібної папки з файлами, а також легко адаптується до завдань будь-якої складності через розширення. Для роботи з Python використовувалось офіційне розширення Microsoft Python Extension, а також інші додаткові модулі, що полегшують аналіз даних і візуалізацію.

Основною мовою програмування, обраною для реалізації даного дослідження, є Python. Ця мова добре підходить для реалізації моделей машинного навчання та нейронних мереж завдяки наявності великої кількості спеціалізованих бібліотек, зокрема NumPy, Pandas, Matplotlib, PyTorch, scikit-learn та інших. Python має читабельний синтаксис, активну спільноту, велику кількість навчальних ресурсів, а також потужну підтримку для інтеграції з іншими інструментами.

Python є інтерпретованою мовою, що означає виконання коду відбувається безпосередньо під час його запуску, без попередньої компіляції. Це прискорює цикл розробки та полегшує налагодження.

Таким чином, комбінація Visual Studio Code як середовища розробки та Python як основної мови програмування створює оптимальні умови для ефективної реалізації задач дослідження методів навчання нейронних мереж у задачах прогнозування часових рядів.

3.2. Основні бібліотеки та фреймворки

Для втілення моделей прогнозування часових рядів у цьому дослідженні було залучено низку спеціалізованих бібліотек мови Python, кожна з яких відіграє важливу роль у підготовці даних, розробці та навчанні нейронних мереж, а також візуалізації результатів. У цьому розділі представлено короткий огляд основних бібліотек, які використовувалися в межах проєкту.

Бібліотека NumPy є фундаментом для наукових обчислень у Python. Вона надає підтримку багатовимірних масивів і матриць, а також численних математичних операцій над ними. Завдяки високій продуктивності та широкому функціоналу, NumPy активно застосовується для обробки даних, нормалізації, створення послідовностей значень та операцій з матрицями, що є критично важливими для функціонування нейронних мереж.

Pandas — це потужна бібліотека для роботи з табличними даними. Вона полегшує імпорт, аналіз, фільтрацію та обробку даних у форматі DataFrame. У цьому проєкті Pandas використовувалася для завантаження даних про ціни криптовалют, попередньої обробки часових рядів, перетворення дат та формування навчальних вибірок для моделей.

Matplotlib та Seaborn

Для візуалізації даних використовувалися бібліотеки Matplotlib та Seaborn. Matplotlib є стандартним інструментом для візуалізації в Python, а Seaborn — його надбудова з більш привабливим та гнучким оформленням графіків. Ці бібліотеки дозволили здійснити візуальний аналіз трендів, сезонності та залишків часових рядів, а також наочно представити результати прогнозування.

PyTorch — це гнучкий фреймворк для розробки та навчання нейронних мереж, розроблений Facebook AI Research. Його ключові переваги включають динамічне обчислювальне дерево, зручну систему автоматичного диференціювання та інтеграцію з CUDA для використання графічних процесорів. У межах цього проєкту PyTorch використовувався для створення моделей MLP, LSTM, CNN та Transformer. Кожна модель була реалізована як окремий клас, що успадковує `torch.nn.Module`, з можливістю налаштування структури, активацій, оптимізаторів та функцій втрат.

Бібліотека `scikit-learn` була застосована для попередньої обробки даних (наприклад, масштабування та розбиття на навчальні й тестові вибірки), а також для розрахунку метрик якості прогнозування, таких як MAE, RMSE, R^2 тощо. Крім того, вона використовувалася для порівняння традиційних моделей регресії та моделей машинного навчання з нейронними мережами.

XGBoost — бібліотека для реалізації градієнтного бустингу на деревах рішень. Вона дає змогу будувати точні моделі, які часто демонструють конкурентоспроможні результати навіть у порівнянні з нейронними мережами. У цьому дослідженні XGBoost виступає як одна з моделей для порівняння з архітектурами на основі PyTorch.

Бібліотека `TorchMetrics` була використана для зручного підрахунку метрик під час навчання моделей у PyTorch. Вона підтримує інтеграцію з PyTorch Lightning та дозволяє уніфікувати обчислення похибок на валідаційній та тестовій вибірках.

Всі ці бібліотеки є відкритими та мають активну спільноту користувачів, що сприяло їх ефективному використанню в дипломному дослідженні. Завдяки використанню цих інструментів вдалося досягти значної гнучкості в розробці, налаштуванні та порівнянні моделей прогнозування на основі різних архітектур.

3.3 Підготовка даних для навчання моделей

Перед тим, як почати навчання моделей для передбачення цін, необхідна відповідна підготовка даних. Перший етап полягає в нормалізації значень ряду цін. Для цього використовується об'єкт `MinMaxScaler` з бібліотеки

`sklearn.preprocessing`, який приводить всі значення до діапазону від 0 до 1. Це гарантує єдину шкалу для всіх вхідних даних та дозволяє уникнути переваги більших чисел під час обчислень. Масив цін попередньо перетворюється у двовимірний формат за допомогою методу `reshape`, що є необхідною вимогою для роботи скейлера.

Далі визначається обчислювальний пристрій, на якому буде відбуватися навчання. Якщо на комп'ютері наявна графічна карта з підтримкою CUDA, обчислення будуть виконуватися на ній, в іншому разі — на центральному процесорі. Це забезпечується за допомогою об'єкта `torch.device`, який автоматично обирає між `"cuda"` та `"cpu"`.

Наступним кроком є створення навчальних послідовностей для подачі на вхід нейронної мережі. Для цього реалізована функція `create_sequences`, яка приймає масив нормалізованих цін та розмір вікна `window_size`. Ця функція формує послідовності фіксованої довжини, кожна з яких містить `window_size` послідовних значень ряду, а також відповідне цільове значення — елемент, що йде безпосередньо після поточної послідовності. У результаті формуються дві матриці — `X` з послідовностями та `y` з відповідними мітками.

Отримані дані розподіляються на навчальну та тестову вибірки у співвідношенні 80% до 20%. Для цього обчислюється індекс розподілу, після чого відбувається нарізка масивів `X` та `y` за допомогою індексації. Важливо, що порядок даних не змінюється, оскільки часові ряди вимагають збереження послідовності.

Після розподілу всі дані перетворюються у тензори типу `float32` за допомогою функції `torch.tensor`, що є необхідною умовою для подальшої роботи з бібліотекою `PyTorch`. Результатом цього етапу є чотири об'єкти —

`X_train_tensor`, `y_train_tensor`, `X_test_tensor` і `y_test_tensor`, які готові для подачі в модель під час процесу навчання.

```

scaler = MinMaxScaler()
price_scaled = scaler.fit_transform(price.reshape(-1, 1))
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

def create_sequences(data, window_size):
    sequences = []
    targets = []
    for i in range(len(data) - window_size):
        seq = data[i:i+window_size]
        label = data[i+window_size]
        sequences.append(seq)
        targets.append(label)
    return np.array(sequences), np.array(targets)

window_size = 20
X, y = create_sequences(price_scaled, window_size)

split = int(len(X) * 0.8)
X_train, y_train = X[:split], y[:split]
X_test, y_test = X[split:], y[split:]

X_train_tensor = torch.tensor(X_train, dtype=torch.float32)
y_train_tensor = torch.tensor(y_train, dtype=torch.float32)
X_test_tensor = torch.tensor(X_test, dtype=torch.float32)
y_test_tensor = torch.tensor(y_test, dtype=torch.float32)

```

Рис. 3.1 Блок коду для підготовки даних

3.4 Створення моделей

В рамках дослідження було втілено п'ять різних архітектур для передбачення часових рядів цін, кожна з яких має свої особливості щодо обробки послідовностей та представлення інформації. Усі моделі, за винятком XGBoost, реалізовані на базі бібліотеки PyTorch як класи, що успадковують `nn.Module`.

Модель MLP реалізується як базова багатошарова перцептронна архітектура. Вона містить два повнозв'язні шари: вхідний (`fc1`) з активацією ReLU та вихідний (`fc2`), який видає одне передбачуване значення. Вона оперує

векторами фіксованої довжини та не враховує послідовну природу даних, але слугує орієнтиром для порівняння складніших моделей.

```
class MLPModel(nn.Module):
    def __init__(self, input_size, hidden_size=64):
        super(MLPModel, self).__init__()
        self.fc1 = nn.Linear(input_size, hidden_size)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(hidden_size, 1)

    def forward(self, x):
        x = self.relu(self.fc1(x))
        return self.fc2(x)
```

Рис. 3.2 Код моделі MLP

Модель LSTM побудована на основі довготривалої короткочасної пам'яті та розроблена для ефективної роботи з послідовностями. Вона складається з двох шарів LSTM-блоку (nn.LSTM) із заданою кількістю прихованих станів, після чого вихід останнього шару передається до лінійного шару (fc). Ця модель дозволяє зберігати інформацію про попередні кроки в часовому ряду.

```
class LSTMModel(nn.Module):
    def __init__(self, input_size, hidden_size=64, num_layers=2):
        super(LSTMModel, self).__init__()
        self.lstm = nn.LSTM(input_size, hidden_size, num_layers, batch_first=True)
        self.fc = nn.Linear(hidden_size, 1)

    def forward(self, x):
        out, _ = self.lstm(x)
        return self.fc(out[:, -1, :])
```

Рис. 3.3 Код моделі LSTM

CNN-модель реалізується як одношаровий згортковий класифікатор з вхідним каналом у вигляді послідовності, яка спершу перетворюється у формат для Conv1d. Після проходження через згортковий шар (conv1), активацію ReLU та адаптивний пулінг (AdaptiveMaxPool1d), результат подається на кінцевий

повнозв'язний шар. Такий підхід дозволяє моделі виділяти локальні закономірності у вхідних вікнах даних.

```
class CNNModel(nn.Module):
    def __init__(self, input_size, seq_len=10):
        super(CNNModel, self).__init__()
        self.conv1 = nn.Conv1d(in_channels=1, out_channels=32, kernel_size=3)
        self.relu = nn.ReLU()
        self.pool = nn.AdaptiveMaxPool1d(1)
        self.fc = nn.Linear(32, 1)

    def forward(self, x):
        x = x.view(x.size(0), 1, -1)
        x = self.relu(self.conv1(x))
        x = self.pool(x)
        x = x.view(x.size(0), -1)
        return self.fc(x)
```

Рис. 3.4 Код моделі CNN

Також реалізовано модель Transformer, яка використовує позиційне кодування (pos_encoder) та механізм багатоголової уваги (nn.TransformerEncoder) для обробки вхідної послідовності. Спершу вхідні дані перетворюються на простір з розмірністю d_model через лінійне перетворення

```
class TransformerModel(nn.Module):
    def __init__(self, input_size, seq_len=20, d_model=64, nhead=4, num_layers=2):
        super(TransformerModel, self).__init__()
        self.seq_len = seq_len
        self.d_model = d_model

        self.embedding = nn.Linear(input_size, d_model)

        self.pos_encoder = nn.Parameter(torch.randn(1, seq_len, d_model))

        encoder_layer = nn.TransformerEncoderLayer(d_model=d_model, nhead=nhead, batch_first=True)
        self.transformer = nn.TransformerEncoder(encoder_layer, num_layers=num_layers)

        self.fc = nn.Linear(d_model, 1)

    def forward(self, x):
        x = self.embedding(x)
        x = x + self.pos_encoder[:, :x.size(1), :]
        x = self.transformer(x)
        return self.fc(x[:, -1, :])
```

Рис. 3.5 Код моделі Transformer

(embedding), після чого додається позиційна інформація. Результат передається до трансформер-енкодера, а фінальне передбачення формується через повнозв'язний шар. Цей підхід дозволяє моделі ефективно виявляти як локальні, так і глобальні залежності у часі.

3.5 Реалізація моделей і процес навчання

Для втілення моделей були задіяні бібліотеки PyTorch (для нейромережових архітектур) та XGBoost (для градієнтного бустингу). Кожна модель була реалізована як окремий клас або функція, що надає гнучкість під час навчання та тестування.

Моделі нейронних мереж (MLP, LSTM, CNN, Transformer) успадковують від `torch.nn.Module`. Їхня структура містить типові шари для кожної архітектури: повнозв'язні (`Linear`) для MLP, рекурентні (`LSTM`) для LSTM, згорткові (`Conv1d`) для CNN та трансформерні енкодери (`TransformerEncoder`) для Transformer-моделі.

Навчання моделей PyTorch організовано у декілька етапів:

Підготовка даних: функція `prepare_tensors` форматує дані згідно архітектури (наприклад, MLP потребує плоского формату, інші - послідовностей).

Цикл навчання: функція `train_model` запускає оптимізацію моделі на визначену кількість епох з функцією втрат `MSELoss` та оптимізатором `Adam`. Під час процесу зберігається історія втрат для подальшого аналізу.

Оцінювання моделі: функція `evaluate_and_plot` здійснює зворотну трансформацію масштабованих прогнозів у вихідний масштаб, обчислює `RMSE`, та візуалізує графік реальних та передбачених значень.

Інтеграція процесу: функція `run_torch_model` є обгорткою для виконання повного циклу тренування та візуалізації для кожної моделі. Вона дозволяє проводити експерименти в автоматизованому режимі для порівняння результатів різних архітектур.

Відтак, запропонована реалізація забезпечує універсальність, зручність запуску та візуальну інтерпретацію результатів роботи моделей прогнозування часових рядів.

3.6 Аналіз результатів моделей

На фінальній стадії було виконано оцінювання точності кожної впровадженої моделі прогнозування. За критерій ефективності обрали кореневу середньоквадратичну помилку (RMSE), що дозволяє виміряти середнє відхилення прогнозованих значень від фактичних, зважаючи на масштаб вихідних даних.

Нижче наведено узагальнені результати моделювання:

Модель	RMSE
MLP	388.2173
LSTM	610.9784
CNN	398.6617
Transformer	697.1194
DLinear	1375.2516

Таблиця 3.1 Результати навчання моделей

Згідно з даними таблиці, першість за результатами здобула модель MLP, продемонструвавши найнижче значення RMSE. Модель CNN також виявила доволі високий рівень точності. Складніші архітектури, зокрема LSTM та Transformer, показали гірші результати, що, ймовірно, є наслідком перенавчання або невідповідності між обсягом даних та архітектурною складністю.

Найгірший показник зафіксовано у моделі DLinear, що, імовірно, обумовлено невідповідністю її структури специфіці даних, або необхідністю глибшого налаштування гіперпараметрів.

Загалом, отримані результати підтверджують ефективність застосування спрощених моделей, таких як MLP або CNN, для завдань прогнозування на невеликих часових рядах.

ВИСНОВКИ

Переддипломна практика є ключовим етапом підготовки майбутнього фахівця у сфері комп'ютерних наук, оскільки дозволяє закріпити здобуті теоретичні знання та набутти цінного практичного досвіду. У ході практики було зосереджено увагу на реалізації та дослідженні моделей штучних нейронних мереж для задач прогнозування часових рядів, що безпосередньо пов'язано з тематикою дипломної роботи.

Завдяки практиці вдалося детальніше ознайомитися з сучасними підходами до побудови моделей прогнозування, зокрема на базі архітектур MLP, LSTM, CNN, Transformer та XGBoost. Було реалізовано програмну частину дослідження, здійснено навчання моделей на реальних даних, а також проведено оцінювання їхньої ефективності з використанням метрик якості.

Під час практики поглибились навички роботи з бібліотеками PyTorch, sklearn, matplotlib, а також було вдосконалено вміння візуалізувати результати та робити висновки на основі отриманих показників. Це дало змогу не лише виявити сильні й слабкі сторони кожної з моделей, а й сформуванати чітке розуміння етапів розробки та тестування систем інтелектуального аналізу даних.

Окрім технічного досвіду, практика посприяла професійному самовизначенню та кращому усвідомленню власного потенціалу у сфері машинного навчання та аналізу даних.

Таким чином, проходження переддипломної практики стало важливою складовою не лише підготовки до захисту кваліфікаційної роботи, а й формування професійних компетентностей, необхідних для подальшої діяльності в галузі інформаційних технологій.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ ТА ДЖЕРЕЛ

1. Штучна нейронна мережа [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Штучна_нейронна_мережа – Назва з екрана. – Дата звернення: 27.05.2025.
2. SkyTek – центр бізнес обслуговування [Електронний ресурс] – Режим доступу: <https://skytek.in.ua/> – Назва з екрана. – Дата звернення: 27.05.2025.
3. Штучні нейронні мережі – вступ та основи [Електронний ресурс] – Режим доступу: <https://itmaster.biz.ua/programming/vision/neural-networks.html> – Назва з екрана. – Дата звернення: 27.05.2025.
4. The Transmitted. Що таке MLP у машинному навчанні? [Електронний ресурс]. – Режим доступу: <https://thetransmitted.com/adlucem/shho-take-mlp-u-mashynnomu-navchanni/> – Назва з екрана.
5. IT Wiki. Long Short-Term Memory (LSTM) [Електронний ресурс]. – Режим доступу: <https://itwiki.dev/data-science/ml-reference/ml-glossary/long-short-term-memory-lstm> – Назва з екрана. Дата звернення: 27.05.2025.
6. Speka. Як працює згортова нейронна мережа: просте пояснення [Електронний ресурс]. – Режим доступу: <https://speka.media/yak-pracyuje-zgortkova-neironna-mereza-proste-poyasnennya-9er7jl> – Назва з екрана. Дата звернення: 27.05.2025.
7. NVIDIA Blog. What is a Transformer Model? [Електронний ресурс]. – Режим доступу: <https://blogs.nvidia.com/blog/what-is-a-transformer-model/> – Назва з екрана. Дата звернення: 27.05.2025.
8. IT Wiki. XGBoost [Електронний ресурс]. – Режим доступу: <https://itwiki.dev/data-science/ml-reference/ml-glossary/xgboost> – Назва з екрана. Дата звернення: 27.05.2025.
9. GOODFELLOW I., BENGIO Y., COURVILLE A. *Deep Learning*. – Cambridge, MA: MIT Press, 2016. – 800 с.