НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ
імені адмірала Макарова

Навчально-науковий інститут автоматики та електротехніки
(повне найменування інституту, факультету)

Кафедра комп'ютеризованих систем управління
(повна назва кафедри)

«Допущений до захисту»

Завідувач кафедри

Черно О.О.

«22» грудня 2024 р.

Пояснювальна записка

до кваліфікаційної роботи

на здобуття другого (магістерського) рівня вищої освіти
(освітньо-кваліфікаційний рівень)

на тему: Інформаційно-вимірювальна система оперативного екологічного моніторингу водного середовища з прогнозом стану якості води на основі машинного навчання

Виконав студент **6** курсу, **6341м** групи

спеціальності **«174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка»**
(шифр і назва спеціальності)

Давидік М. І.
(прізвище та ініціали)

Керівник **Топалов А. М.**
(прізвище та ініціали)

м. Миколаїв - 2024

Національний університет кораблебудування імені адмірала Макарова

НН Інститут автоматики і електротехніки
Кафедра комп'ютеризованих систем управління
Освітньо-кваліфікаційний рівень **«магістр»**
Спеціальність **174 «Автоматизація та комп'ютерно-інтегровані технології та робототехніка»**
Освітня програма **«Комп'ютеризовані системи управління та автоматика»**

ЗАТВЕРДЖУЮ
Завідувач кафедри КСУ

_____**Черно О.О.**
« » **2024 р.**

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНА РОБОТУ СТУДЕНТУ

Давидіку Максиму Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи: *Інформаційно-вимірювальна система оперативного екологічного моніторингу водного середовища з прогнозом стану якості води на основі машинного навчання*

керівник роботи *Топалов Андрій Миколайович кандидат технічних наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "9" жовтня 2024 р. № 254-уч

2. Строк подання роботи *15.11.24*
3. Вихідні дані до роботи Мова Python, датасет параметрів водного середовища, бібліотеки: pandas, numpy, tensorflow, sklearn, seaborn, matplotlib.

4. Зміст розрахунково-пояснювальної записки (основні задачі дослідження)
Огляд основних понять про машинне навчання та його використання у сфері екологічного моніторингу
Методи та засоби для реалізації моделі машинного навчання
Розробка та налаштування моделі машинного навчання
Охорона праці

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Загальне представлення сфери машинного навчання
Класифікція методів машинного навчання

Приклад задачі регресії

Приклад задчі класифікації

Приклад задачі виявлення аномалій

Діаграма використання машинного навчання у різних сферах діяльності

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Охорона праці	*Топалов А. М. кандидат технічних наук, доцент*		

7. Дата видачі завдання: «10» вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційна проектування	Строк виконання	Примітка
1	*Огляд основних понять про машинне навчання та його використання у сфері екологічного моніторингу*	*10.10.2024*	*Виконано*
2	*Методи та засоби для реалізації моделі машинного навчання*	*23.10.2024*	*Виконано*
3	*Розробка та налаштування моделі машинного навчання*	*12.11.2024*	*Виконано*
4	*Охорона праці*	*17.12.2024*	*Виконано*

Студент _____ Давидік М. І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Топалов А. М.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Обсяг кваліфікаційної роботи сторінки, у якій: рисунки та використаних джерел, таблиць.

Кваліфікаційна робота присвячена розробці інформаційно-вимірювальної системи оперативного екологічного моніторингу водного середовища з прогнозом стану якості води на основі машинного навчання.

В процесі розробки моделі машинного навчання, модель була спроектова та протестована.

Ключові слова: Python, машинне навчання, екологія, JavaScript, інформаційно-вимірювальна система.

ABSTRACT

The scope of the qualification work page, which includes: drawings and used sources, tables.

The qualification work is devoted to the development of an information and measurement system for operational ecological monitoring of the water environment with a forecast of the state of water quality based on machine learning.

In the process of developing a machine learning model, the model was designed and tested.

Keywords: Python, machine learning, ecology, JavaScript, information and measurement system.

Зміст

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ **7**

ВСТУП ..... Ошибка! Закладка не определена.

РОЗДІЛ 1. ОГЛЯД ОСНОВНИХ ПОНЯТЬ ПРО МАШИННЕ НАВЧАННЯ ТА ЙОГО ВИКОРИСТАННЯ У СФЕРІ ЕКОЛОГІЧНОГО МОНІТОРИНГУ **9**

1.1 Основи машинного навчання **9**

1.1.1 Типи машинного навчання **11**

1.1.2 Задачі машинного навчання **13**

1.1.3 Етапи розробки моделі **17**

1.2 Огляд існуючих рішень у сфері екологічного моніторингу **18**

РОЗДІЛ 2. МЕТОДИ ТА ЗАСОБИ ДЛЯ РЕАЛІЗАЦІЇ МОДЕЛІ МШИННОГО НАВЧАННЯ **30**

2.1 Мова Python **30**

2.2 Опис бібліотек для написання моделі **33**

2.3 Методи аналізу часових рядів **38**

РОЗДІЛ 3. РОЗРОБКА ТА НАЛАШТУВАННЯ МОДЕЛІ МАШИННОГО НАВЧАННЯ **47**

3.1 Підготовка даних до роботи з моделью машинного начання 47

3.2 Аналіз даних 48

3.3 Написання моделі машинного навчання та її тестування 54

РОЗДІЛ 4.ОХОРОНА ПРАЦІ ТА НАВКОЛИШНЬОГО СЕРЕДОВИЩА 65

4.1. Охорона праці при роботі з комп'ютерною технікою 65

4.2. Охорона довкілля **74**

ВИСНОВОКИ 76

ПЕРЕЛІК ПОСИЛАНЬ 77

ДОДАТОК А 79

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

SMA – Simple Moving Average (проста ковзна середня)

EMA - Exponential Moving Average (експоненціальна ковзна середня)

WMA – Weighted Moving Average (зважена ковзна середня)

ML – Machine Learning (машинне навчання)

MA - Moving Average (ковзна середня)

ВСТУП

У наш час машинне навчання це потужний інструмент який активно використовується у сфері моніторингу навколишнього середовища. Реалізація нових способів збору, аналізу та інтерпретації екологічних даних, а також їх обробки та виявлення у них прихованих закономірностей робить машинне та глибоке навчання одним з найкращих рішень для аналізу складної та динамічної природи екологічних даних. Можливость прогнозування змін клімату та отримання даних моніторингу біометричних параметрів покращує наше розуміння навколишнього середовища та дає можливість приймати набагато ефективніші рішеня у сфері екологічного менеджменту та політики на основі фактичних даних.

РОЗДІЛ 1
ОГЛЯД ОСНОВНИХ ПОНЯТЬ ПРО МАШИННЕ НАВЧАННЯ ТА ЙОГО ВИКОРИСТАННЯ У СФЕРІ ЕКОЛОГІЧНОГО МОНІТОРИНГУ

1.1. Основи машинного навчання

Машинне навчання це одна з підгалузей сфери інформаційних технологій яка спеціалізується на створенні алгоритмів які для покращення результатів обробки використовують вже наявні данні. Данні можуть бути природними, створеними людиною, або генеровані іншим[1] алгоритмом. Також машинне навчання можна описати як процес вирішення певної проблеми у два кроки: 1) сбір та обробка даних. 2) створення статистичної моделі на основі цього набору даних.

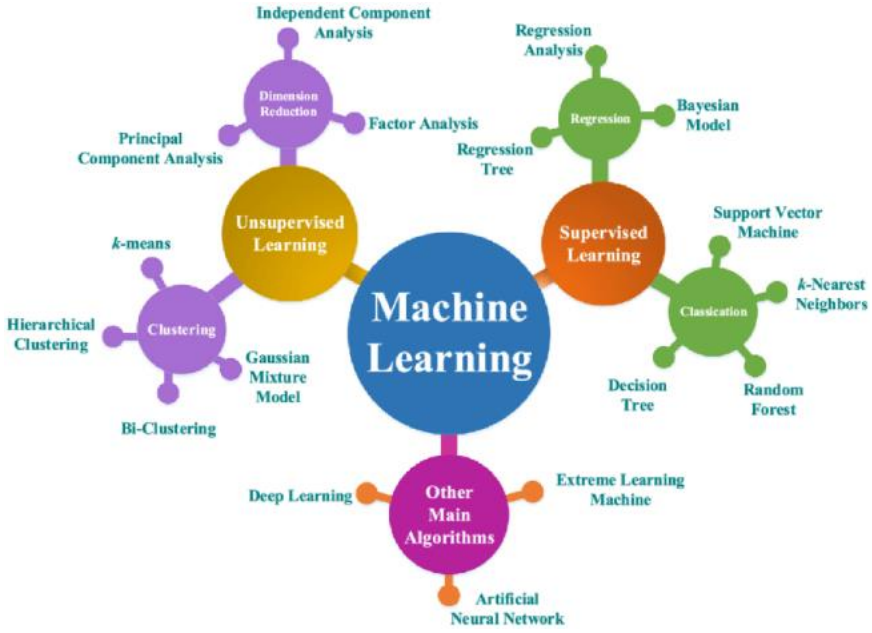

Рис. 1.1 Загальне представлення сфери машинного начання

Основна мета машинного навчання – передбачення результату за наявними вхідними даними. Чим ширша видірка вхідних даних, тим ефективніше модель знайде закономірності та видасть точніший результат. Виходячи з цього для реалізації моделі машинного навчання нам потрібні три речі: дані, ознаки та алгоритми.

Дані. Процесс представлення даних відрізняється залежно від задачі, самі данні при цьому можуть бути як дискретними так і абстрактними.

Ознаки. Ознаки дають можливість моделі краще обробляти існуючі дані завдяки конкретизації запитів. Для оптимізації процессу роботи моделі реалізується вибір ключових ознак для використання[2] їх у подальшому навчанні. Це процесс відбувається автоматично для мінімізації суб'єктивності моделі.

Алгоритми. Алгоритм обирається в залежності від задчі яку має вирішити модель. В свою чергу від обраного методу буде залежит точність, швидкість роботи та розмір готової моделі. Але також під час вибору алгоритму потрібно брати до у ваги наявність так званих «брудних» (нечіткі, неточні, з пропусками), що може призвети до погіршення роботи моделі. Рішенням ціє проблеми є розширення вибірки даних[6].

Типи ознаки

Поділяють ознаки на якісні та кількісні. Для визначення типу, потрібно з'ясувати, чи об'єктивно можна виміряти досліджувану характеристику за допомогою чисел[5].

У якісних ознаках інформація наведена характеристиками або числами, тоді як самі спостереження можна поділити на декілька груп.

Номінальні ознаки використовують для позначення змінних, що не мають кількісного вираження. Номінальні ознаки не прив'язані до порядку що означає що підсумковий результат не залежить від зміни порядку значень.

Порядкові ознаки представляють собою суміш числових і категоріальних ознак. Ознаки можна розбити на категорії, але числа, що асоціюються з кожною категорією, мають значення. Такі ознаки можна узагальнювати за допомогою частотності, пропорцій, відсоткових часток, а візуалізувати – за допомогою колових і стовпчастих діаграм. Крім того, можна використовувати відсотки, медіану, моду, міжквартильний розмах. На додаток до порядкових та номінальних є особливий тип категоріальних ознак – бінарні (двійкові). Бінарні ознаки набувають лише два значення: «так» або «ні», що можна подати як 1 та 0. Бінарні ознаки широко використовуються в класифікаційних моделях машинного навчання. Як приклади бінарних ознак можна навести такі ситуації: скасувала людина підписку чи ні, купила машину чи ні.

Кількісні ознаки. Інформацію записують у вигляді чисел, воня являє собою об’єктивне вимірювання чи підрахунок. Температура, вага, кількість транзакцій – ось приклади кількісних ознак. Такі ознаки також називають числовими.

Дискретні кількісні ознаки – це підрахунок випадків наявності характеристики, результату, предмета, діяльності. Ці виміри неможливо поділити на більш дрібні частини без втрати сенсу[6].

Неперервні ознаки можуть набувати практично будь-якого числового значення і можуть бути поділені на менші частини, включаючи дробові та десяткові значення. Неперервні змінні часто вимірюють за шкалою.

1.1.1 Типи машинного навчання

Виділятью чотири основні типи машинного навчання (рис. 1.2)

Рис. 1.2 Класифікація методів машинного навчання

Навчання з вчителем (Supervised Learning)

Цей тип навчання передбачає наявність так званого «вчителя» який заздалегідь відмічає усі необхідні дані які модель має використати для начання. Варто зазначити що «вчителем» може бути не лише людина ай програма або алгоритм. Цей метод є одним з най ефективніших та часто використовуємих[11].

Навчання без вчителя (Unsupervised Learning)

Цей метод передбачає що машина має сама виявити закономірності у обраних даних. На пратці цей метод частіше використовують для аналізу та підготовки даних, а не як основний алгоритм. Одним з можливих варіантів використання цього методу як основного алгоритму є ситуацію при якій розмічення даних, людиною, неможливе.

Навчання з підкріпленням (Reinforcement Learning)

Метод базується на реалізації так званого «агента» який контролюється комп’ютером. Агент шляхом проб та помилок повинен маємаксимізувати позитивні та мінімізувати негативні відклики. Такі алгоритми використовуються для навчання моделей що будуть застосовуватися у динамічному середовищі та мають швидко до нього адаптуватися.

Це метод найбільш схожий на навчання людини так як має у собі систему покарань та нагород[24].

Навчання з частковим залученням вчителя (Semi-supervised Learning

Навчання з частковим залученням вчителя (Semi-supervised Learning) — це підхід у машинному навчанні, який поєднує елементи навчання з учителем (supervised learning) і без учителя (unsupervised learning). У цьому підході використовуються як мітки для невеликої частини даних, так і для значної частини — тільки необроблені або немітовані дані,в основному, алгоритм має невелику кількість даних із правильними мітками (приклади з учителем) і велику кількість немітованих даних. Метою є навчання моделі, використовуючи обидва типи даних (мітовані й немітовані), щоб створити точніші прогнози, ніж у випадку використання тільки мітованих даних.

Це корисно в реальних ситуаціях, коли анотовані дані дорогі чи трудомісткі у зборі, а немітовані дані доступні в великій кількості[23].

1.1.2 Задачі машинного навчання

Найпопулярніші задачі машинного навчання: регресія; класифікація; кластеризація; зменшення розмірності; виявлення аномалій; пошук правил.

Регресія

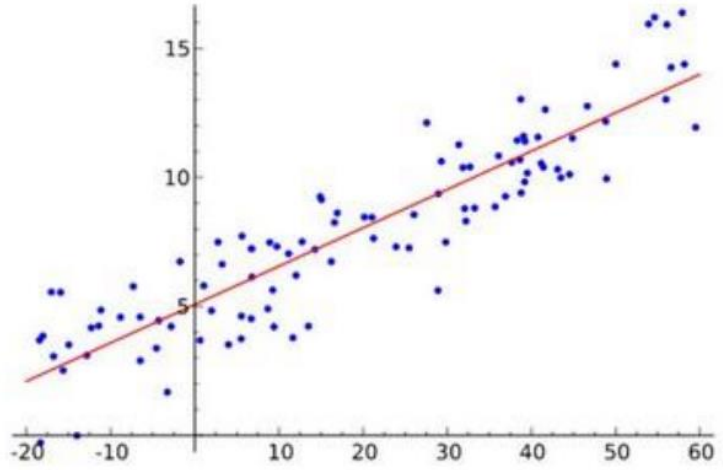

Рис. 1.3 Приклад задачі регресії

Регресія (рис. 1.3(— це один із ключових методів статистичного аналізу та машинного навчання, який використовується для моделювання залежності між однією або кількома незалежними змінними (факторами) та залежною змінною (цільовою величиною). Основна мета регресії — передбачити значення залежної змінної на основі значень незалежних змінних або виявити зв'язок між ними.

Часто регресію використовують для вирішення задач прогнозування так як для неї необхідно встановити функціональну залежністі між вхідними і неперервними вихідними змінними.

Класифікація

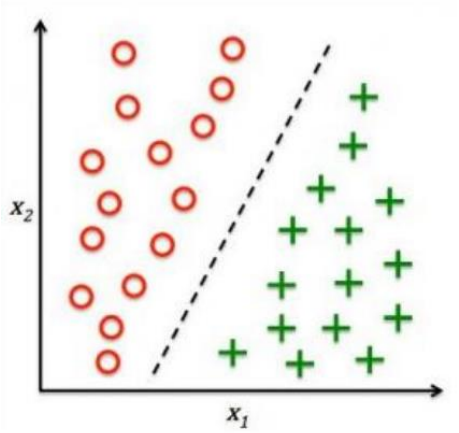

Рис. 1.4 Приклад задачі класифікації

Принцип класифікації (рис. 1.4) полягає у встановлені функціональної залежності між вхідними і вихідними змінними. Класифікації використовується для вирішення задач приналежності об’єктів до одного зі заздалегідь відомих класів[22].

Алгоритми класифікації дозволяють розділити об’єкти відповідно до зазначених заздалегідь класів. Найпростішим із технічного погляду завданням класифікації є бінарна класифікація – коли об’єкти потрібно розподілити між двома класами. Для єфективної роботи алгоритму класифікації дані з категоріями та ознаками мають бути уже розміченими. Залежно від певних ознак алгоритм визначає, до якого з класів можна віднести об’єкт.

Кластеризація

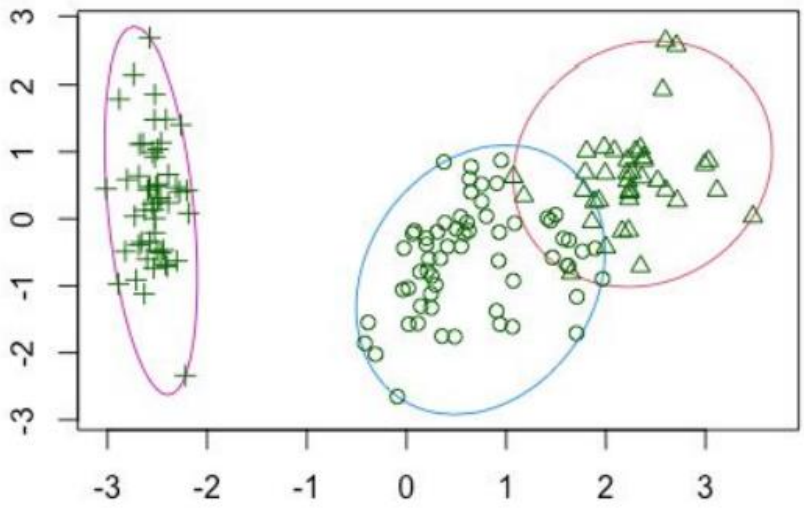

Рис. 1.5 Приклад задачі кластеризації

Кластеризація (рис. 1.5) це процесс групування об’єктів за їх властивостями. Об’єкти у кожному кластері повинні мати однакові ознаки відповідно до того у який кластер вони були зргруповані і відрізнятися від об’єктів, що увійшли в інші кластери. Чим більше схожі об’єкти в середині

кластера і чим більше відмінностей між кластерами, тим точніше кластеризація[22].

Також алгоритми кластеризації дозволяє розділити об’єкти в разі якщо класи заздалегідь не зазначені, а кластери сформуются за подібністю елементів.

Виявлення аномалій

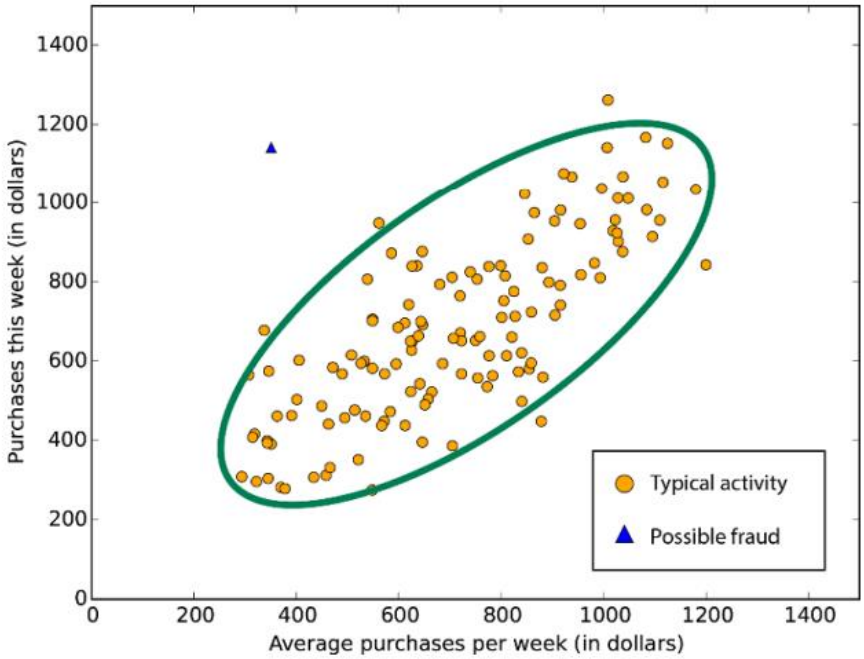

Рис. 1.6 Приклад задачі виявлення аномалій

Методи виявлення аномалій (рис. 1.6) поділяються на три основні категорії, кожна з яких має свої особливості підходу до ідентифікації відхилень у даних[25].

Методи виявлення аномалій без нагляду (неконтрольовані алгоритми). Цей метод працює із непозначеним набором даних, виходячи з припущення, що більшість зразків у цьому наборі є нормальними. Неконтрольовані алгоритми шукають дані, які найменш відповідають основній структурі або шаблонам решти набору. Вони часто застосовуються в ситуаціях, коли відсутні розмічені дані, і включають методи, такі як

кластеризація, зниження розмірності або ізоляція відхилень (наприклад, Isolation Forest).

Методи контрольованого виявлення аномалій**.** Цей методи потребує попередньо розміченого набору даних, де зразки позначено як «нормальні» або «аномальні». Вони використовують алгоритми машинного навчання для навчання класифікатора, здатного розрізняти нормальні та аномальні дані. Важливою особливістю задачі є незбалансований характер набору даних, оскільки аномальні зразки зазвичай значно рідші за нормальні. Контрольовані методи включають алгоритми, такі як логістична регресія, SVM, деревоподібні моделі (наприклад, Random Forest) або нейронні мережі.

Методи напів-контрольованого виявлення аномалій. Цей підхід базується на побудові моделі, що представляє нормальну поведінку, використовуючи набір даних, у якому доступні лише нормальні зразки. Потім тестові дані перевіряються на основі того, наскільки вони відповідають цій моделі. Напів-контрольовані методи застосовуються, коли розмічені аномальні дані відсутні або їх недостатньо для повноцінного навчання. Алгоритми, що часто використовуються, включають моделі на основі щільності (наприклад, Gaussian Mixture Model), автоенкодери або моделі на основі підтримки щільності, такі як One-Class SVM.

1.1.3 Етапи розробки моделі

Процес розробки певної моделі МН складається із етапів:

- процес підготовки (представлення) даних
- процес конструювання алгоритму
- процес тренування алгоритму на наявних даних
- процес валідації алгоритму на тестових даних

Першим етапом є процес представлення даних. Від якості представленних даних в МН буде залежит ефективність її роботи так як це є

одним із ключових параметрів моделі МН. В залежності від задачі, процес представлення даних для алгоритму відрізнятиметься. Одним із найпоширеніших методів представлення даних у МН є ознакове описання об’єкту. Якість роботи алгоритму сильно залежить від того, наскільки ознаки, які ми даємо йому на вхід, справді впливають на мітку, яку ми отримуємо на виході. Другим етапом е визначення типу задачі (конструювання алгоритму). Третій і четвертий етапи – вибір правильних параметрів та підготовка тренувальних даних.

1.2. Огляд існуючих рішення у сфері екологічного моніторингу

За останні три десятиліття досягнуто значного прогресу у створенні та впровадженні сучасних чисельних гідродинамічних методів прогнозування екологічних процессів[4]. Це стало можливим завдяки стрімкому розвитку потужних комп’ютерних технологій, удосконаленню математичних моделей, які стали більш реалістичними й точними, а також суттєвому збільшенню сфер використання машинного навчання (рис. 1.7).

Рис. 1.7 Діаграма використання машинного навчання у різних сферах діяльності

Persefoni

Persefoni — це інтелектуальна платформа для управління вуглецевими викидами, яка використовує штучний інтелект. Її головна мета — допомагати компаніям точно оцінювати, контролювати та звітувати про свої викиди вуглецю. У зв'язку зі зростанням вимог до дотримання кліматичних норм, Persefoni стала популярним рішенням серед промислових підприємств, оскільки значно спрощує процес обліку вуглецевих викидів завдяки своїм аналітичним інструментам. Використовуючи штучний інтелект для аналізу даних, платформа полегшує моніторинг викидів, дозволяючи організаціям приймати більш ефективні рішення щодо зменшення свого впливу на довкілля. Крім того, Persefoni автоматизує звітність, забезпечуючи відповідність міжнародним стандартам і регуляторним вимогам із максимальною точністю та прозорістю[3].

Платформа використовує передові алгоритми для аналізу великих обсягів даних і забезпечує комплексний підхід до управління викидами, враховуючи весь життєвий цикл продуктів і процесів компанії. Завдяки інструментам Persefoni організації можуть інтегрувати дані з різних джерел, включаючи постачальників, логістичні ланцюжки та операційні системи, що робить оцінку викидів більш повною та точною.

Використовуючи штучний інтелект для аналізу вуглецевого сліду, платформа дозволяє компаніям не тільки оцінювати свій вплив на довкілля, але й ефективно скорочувати його шляхом впровадження стратегій сталого розвитку. Це включає оптимізацію енергоспоживання, пошук альтернативних рішень і створення більш екологічно чистих процесів виробництва.

Крім цього, Persefoni автоматизує звітність, що особливо важливо для компаній, які прагнуть відповідати міжнародним стандартам, таким як GHG Protocol або TCFD, а також новим кліматичним регуляціям у різних країнах.

Прозорість і точність даних, що забезпечується платформою, сприяє підвищенню довіри серед зацікавлених сторін, таких як інвестори, регулятори та громадськість.

Persefoni виділяєтьс зручним інтерфейсом високими показниками аналітичних можливостей. Інтегруючи дані з усього ланцюга доданої вартості організації, Persefoni надає глибоке розуміння джерел викидів і пропонує дієві стратегії зменшення викидів вуглецюх[3].

ARIA

ARIA від BrainBox AI — це передовий інструмент на основі штучного інтелекту, створений для значного скорочення енергоспоживання та зменшення викидів парникових газів шляхом оптимізації роботи систем HVAC (опалення, вентиляції та кондиціонування повітря). Використовуючи сучасні алгоритми машинного навчання, ARIA автономно налаштовує функціонування HVAC, враховуючи такі фактори, як рівень заповнюваності приміщення, прогнози погоди, режим роботи будівлі, поведінку користувачів і енергоспоживання. Завдяки цьому система забезпечує точний і ефективний контроль температури, якості повітря та комфорту, мінімізуючи енергетичні втрати та знижуючи витрати.

ARIA є не лише інтелектуальною, але й адаптивною системою, яка постійно самонавчається, використовуючи дані з реального часу. Це дає можливість системі ставати ефективнішою з кожним днем, автоматично виявляючи оптимальні рішення для кожної конкретної будівлі. Такий підхід сприяє тривалому скороченню операційних витрат, значній економії енергії та зменшенню впливу на довкілля.

Окрім цього, ARIA інтегрується з існуючими інфраструктурами HVAC без необхідності кардинальних змін, що робить її доступним і простим у впровадженні рішенням. Вона підтримує відповідність будівлі сучасним

екологічним стандартам і може використовуватися як частина стратегії сталого розвитку компанії або власника будівлі.

Таким чином, ARIA від BrainBox AI допомагає не лише досягати значної економії коштів, але й скорочувати вуглецевий слід будівель, одночасно підтримуючи комфортні умови для їхніх мешканців та сприяючи глобальній боротьбі з кліматичними змінами.

Atmo

Atmo — це високотехнологічний інструмент на основі штучного інтелекту, спеціально розроблений для точного, швидкого та адаптивного прогнозування погоди, що значно перевершує традиційні метеорологічні методи за швидкістю й точністю. Використовуючи передові алгоритми машинного навчання та моделювання великих обсягів даних, Atmo надає детальні прогнози, які є критично важливими для таких галузей, як сільське господарство, транспорт, енергетика, авіація та управління стихійними лихами.

Цей інноваційний підхід дозволяє Atmo обробляти величезні масиви атмосферних даних у режимі реального часу, враховуючи складні кліматичні взаємодії, і забезпечує високоточні прогнози для конкретних місцевостей. Наприклад, інструмент може передбачити екстремальні погодні явища, такі як сильні бурі, посухи чи раптові зливи, допомагаючи громадам і бізнесу вживати проактивних заходів для мінімізації ризиків і втрат.

Особливо важливу роль Atmo відіграє в умовах посилення кліматичних змін, які спричиняють більш часті та непередбачувані погодні аномалії. Надійність і точність прогнозів Atmo є ключовими для стратегічного планування стійкості, зокрема в адаптації до змін клімату, запобіганні збиткам і забезпеченні безпеки населення.

Atmo також відрізняється своєю здатністю адаптуватися до змінних умов і інтегрувати дані з різних джерел, включаючи супутникові знімки,

сенсори IoT та наземні метеостанції. Завдяки цьому інструмент може не лише надавати прогнози, а й формувати рекомендації для оптимізації операцій у реальному часі.

Орієнтованість Atmo на розв'язання як екологічних, так і комерційних завдань дозволяє йому ефективно підтримувати сталий розвиток, забезпечуючи підприємства й громади необхідною інформацією для прийняття обґрунтованих рішень. Інструмент не лише підвищує ефективність реагування на погодні виклики, а й сприяє глобальним зусиллям із захисту людей і економічних ресурсів від наслідків змін клімату.

IBM Environmental Intelligence Suite

IBM Environmental Intelligence Suite — це потужна платформа на основі штучного інтелекту, створена для підтримки компаній у моніторингу, прогнозуванні та реагуванні на кліматичні ризики, а також в оцінці впливу на навколишнє середовище. Інтегруючи погодні, кліматичні та оперативні дані, цей інструмент надає організаціям змогу приймати обґрунтовані рішення, оптимізувати використання ресурсів і підвищувати стійкість своїх ланцюгів постачання в умовах зростаючих кліматичних викликів.

Платформа пропонує розширені функції, такі як моніторинг екологічних показників у реальному часі, аналітика ризиків і прогнозування кліматичних змін. Це дозволяє компаніям не лише захищати свою діяльність від потенційних перебоїв, але й досягати довгострокових цілей сталого розвитку. Крім того, IBM Environmental Intelligence Suite включає інструменти для точного обліку викидів вуглецю, дотримання нормативних стандартів і звітності щодо екологічної ефективності.

Особливу цінність платформа має для галузей, які найбільше залежать від кліматичних умов і ресурсів, таких як сільське господарство, енергетика, логістика, транспорт і роздрібна торгівля. Наприклад, сільськогосподарські компанії можуть використовувати платформу для прогнозування

екстремальних погодних явищ, енергетичні підприємства — для оптимізації роботи інфраструктури в умовах змін навколишнього середовища, а роздрібні мережі — для управління постачанням товарів під час кліматичних ризиків.

Ключовою перевагою IBM Environmental Intelligence Suite є її здатність автоматизувати аналіз великих обсягів даних і пропонувати конкретні рекомендації для адаптації бізнес-процесів. Організації можуть швидко реагувати на зміни в екологічному середовищі, мінімізувати збитки та зменшувати вуглецевий слід, залишаючись конкурентоспроможними в умовах глобальних змін.

Платформа не лише допомагає бізнесу інтегрувати стратегії сталого розвитку, але й сприяє підвищенню прозорості в питаннях екологічної відповідальності. IBM Environmental Intelligence Suite є комплексним рішенням для тих, хто прагне забезпечити стійкість, адаптивність і довгостроковий успіх своєї діяльності в епоху кліматичних змін.

FlyPix AI

FlyPix AI пропонує інноваційну платформу геопросторового аналізу, яка використовує супутникові та дронові зображення для ефективного моніторингу стану навколишнього середовища. Завдяки передовим алгоритмам штучного інтелекту, FlyPix AI здатна обробляти великі обсяги візуальних даних, забезпечуючи детальний аналіз змін землекористування, стану рослинності та впливу людської діяльності на довкілля навіть на масштабних географічних територіях.

Ця платформа є особливо корисною для таких галузей, як сільське господарство, гірничодобувна промисловість, лісове господарство та урбаністика, де необхідно регулярно відстежувати екологічні зміни та забезпечувати відповідність екологічним нормам. FlyPix AI дозволяє бізнесам не лише виявляти зміни у використанні земель чи деградацію

екосистем, але й прогнозувати можливі наслідки цих змін завдяки аналізу трендів у часі.

Основними перевагами FlyPix AI є її зручний інтерфейс і можливість проводити аналіз у реальному часі. Компанії можуть отримувати швидкі та точні інсайти, які допомагають їм приймати обґрунтовані рішення щодо управління земельними ресурсами, зменшення впливу на навколишнє середовище та оптимізації екологічних процесів.

Платформа автоматизує складні процеси геопросторового аналізу, що значно скорочує час і витрати на обробку даних. Це дозволяє підприємствам адаптуватися до змін у навколишньому середовищі та розробляти стратегії сталого розвитку, спрямовані на мінімізацію екологічних ризиків і підтримку екосистем.

FlyPix AI також забезпечує інтеграцію з іншими системами управління екологічними даними, створюючи комплексний підхід до моніторингу й аналізу. Це робить платформу незамінним інструментом для компаній, які прагнуть активно реагувати на екологічні виклики, підвищувати свою ефективність і сприяти реалізації глобальних цілей сталого розвитку.

CarbonBright

CarbonBright — це передовий інструмент на основі штучного інтелекту, створений для миттєвої оцінки екологічного впливу споживчих товарів протягом усього їхнього життєвого циклу. Використовуючи технології аналізу життєвого циклу (LCA), CarbonBright надає підприємствам детальне розуміння їхнього вуглецевого сліду — від видобутку сировини до утилізації продуктів.

Інструмент аналізує дані по всьому ланцюжку постачання, допомагаючи ідентифікувати «гарячі точки» — зони високого впливу на

навколишнє середовище, де можна впровадити зміни для скорочення викидів. Завдяки цьому, CarbonBright підтримує компанії в переході до більш екологічно відповідального виробництва, сприяючи сталому розвитку та зниженню впливу на клімат.

Зручний інтерфейс платформи дозволяє бізнесу відстежувати екологічний слід своїх продуктів у режимі реального часу, візуалізувати дані та забезпечувати прозору комунікацію з клієнтами. Це значно полегшує досягнення корпоративних цілей у сфері сталого розвитку та демонстрацію екологічних ініціатив для споживачів і зацікавлених сторін.

Платформа є ідеальним рішенням для компаній у галузях виробництва, роздрібної торгівлі, харчової промисловості та будь-яких інших сферах, де контроль за екологічним впливом продуктів має критичне значення. Завдяки CarbonBright підприємства можуть не лише зменшити свій вуглецевий слід, але й підвищити прозорість екологічних зусиль, сприяючи довірі клієнтів і партнерів.

Infogrid

Infogrid — це платформа на основі штучного інтелекту, яка допомагає оптимізувати управління будівлями, приділяючи особливу увагу енергоефективності та сталому розвитку. Використовуючи передові технології для моніторингу таких параметрів, як температура, вологість, якість повітря та інші умови навколишнього середовища, Infogrid дає змогу підприємствам створювати більш здорові, комфортні та економічно ефективні простори.

Платформа інтегрується з існуючими системами управління будівлями та обробляє дані в реальному часі для оптимізації роботи систем опалення, вентиляції, кондиціонування повітря (HVAC) та освітлення. Це сприяє значному зниженню енергоспоживання, експлуатаційних витрат і викидів вуглецю. Крім того, функція прогнозної аналітики Infogrid дозволяє

ідентифікувати потенційні технічні проблеми, надаючи змогу компаніям своєчасно вживати заходів і уникати дорогого ремонту або збоїв у роботі.

Infogrid знаходить застосування у різних галузях, включно з комерційною нерухомістю, закладами охорони здоров'я та індустрією гостинності, де ефективне управління будівлями має ключове значення для досягнення операційної досконалості. Завдяки підтримці практик сталого розвитку, Infogrid допомагає організаціям оптимізувати використання ресурсів, мінімізувати екологічний вплив і досягати стратегічних екологічних цілей.

Sylvera

Sylvera — це інноваційна платформа, заснована на штучному інтелекті, яка спеціалізується на перевірці та оцінці проєктів компенсації викидів вуглецю, забезпечуючи прозорість, точність і надійність у глобальних ініціативах зі скорочення вуглецевого сліду. Використовуючи передові алгоритми для аналізу супутникових знімків, екологічних даних і проектної документації, Sylvera оцінює реальну ефективність заходів із уловлювання та зберігання вуглецю, таких як лісовідновлення, заліснення та секвестрація вуглецю в ґрунті.

Ця платформа надає компаніям необхідні інструменти для прийняття обґрунтованих рішень щодо інвестицій у вуглецеві компенсації, гарантуючи, що придбані вуглецеві кредити мають реальний і довготривалий позитивний вплив на навколишнє середовище. Завдяки Sylvera підприємства можуть впевнено підтримувати свої стратегії вуглецевої нейтральності, відповідати нормативним вимогам і демонструвати прозору звітність перед зацікавленими сторонами.

Sylvera є ключовим інструментом для організацій, які прагнуть досягти стійкості та уникнути ризиків, пов'язаних із «зеленим відмиванням». Завдяки глибокій аналітиці та ретельній перевірці проєктів, платформа допомагає

підприємствам будувати довіру, демонструвати справжню відданість екологічним цілям і сприяти досягненню глобальної вуглецевої нейтральності.

Vortexa

Vortexa — це високотехнологічна платформа, побудована на базі штучного інтелекту, що збирає, аналізує та прогнозує глобальні дані про енергетичні вантажі. Вона забезпечує прозорий і детальний огляд руху сирої нафти, нафтопродуктів і природного газу в усьому світі. Використовуючи алгоритми штучного інтелекту, Vortexa аналізує дані про перевезення та торгівлю, допомагаючи компаніям ухвалювати обґрунтовані рішення щодо закупівлі, торгівлі енергоресурсами та стратегічного позиціонування на ринку.

Прогностичні можливості платформи дають змогу підприємствам оптимізувати ланцюги постачання, знижувати операційні витрати, підвищувати ефективність використання енергоресурсів і адаптуватися до змін у динамічному глобальному середовищі. Особливу цінність Vortexa надає компаніям у сферах енергетики, логістики та фінансів, де оперативний доступ до даних про енергетичні потоки та їх аналіз у реальному часі є ключовим для успішного стратегічного планування.

Завдяки акценту на прозорості та екологічній ефективності, Vortexa підтримує організації у досягненні їхніх цілей сталого розвитку. Вона допомагає зменшити вуглецевий слід, забезпечуючи більш раціональне управління енергією, і сприяє побудові стійкішого глобального енергетичного ринку. Ця платформа є важливим інструментом для підвищення екологічної відповідальності та сталості енергетичного сектору.

FarmLab

FarmLab — це інноваційна платформа моніторингу навколишнього середовища, що використовує можливості штучного інтелекту для оцінки та управління станом ґрунту, сприяючи сталому сільському господарству та скороченню викидів вуглецю. Завдяки інтеграції дистанційного зондування, супутникових знімків і аналізу даних про ґрунт, FarmLab забезпечує фермерів, землевласників і аграрні підприємства детальною інформацією про рівень вуглецю в ґрунті, концентрацію поживних речовин і загальний екологічний стан землі.

Ця платформа є незамінним інструментом у галузях сільського господарства та землеустрою, де якість ґрунту впливає на продуктивність культур, здатність поглинати вуглець і екологічну стійкість територій.

FarmLab надає аналітичні можливості на базі штучного інтелекту, що дозволяють відстежувати динаміку змін у стані ґрунту, приймати обґрунтовані рішення щодо управління земельними ресурсами та впроваджувати практики регенеративного землеробства. Це сприяє збереженню родючості ґрунтів, підвищенню їхньої стійкості до деградації та зменшенню впливу аграрної діяльності на клімат. Крім того, FarmLab допомагає оптимізувати використання добрив і води, знижуючи витрати та підтримуючи екологічний баланс.

Висновоки за розділом 1

У першому розділі було розглянуто основні поняття про машинне навчання. Проведен аналіз методів машинного навчання та основних його етапів. Також було розглянуто та проаналізовано існуючі рішення застосування машинного навчання у сфері екологічного моніторингу.

РОЗДІЛ 2

МЕТОДИ ТА ЗАСОБИ ДЛЯ РЕАЛІЗАЦІЇ МОДЕЛІ МШИННОГО НАВЧАННЯ

2.1 Мова Python

Мова програмування Python найкраще підходить для таких завдань, оскільки він досить простий у порівнянні з іншими мовами. Більше того, він має відмінні показники обробки даних. Python - популярна мова програмування, створена в 1991 році Гвідо ван Россумом. Вона має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис та динамічне введення тексту Python, поряд з інтерпретованою природою, роблять її ідеальною мовою для сценаріїв та швидкої розробки додатків у багатьох областях на більшості платформ[25]. Вона може бути використана для веб-розробки на стороні сервера, розробки програмного забезпечення та системних сценаріїв (рис. 2.1).

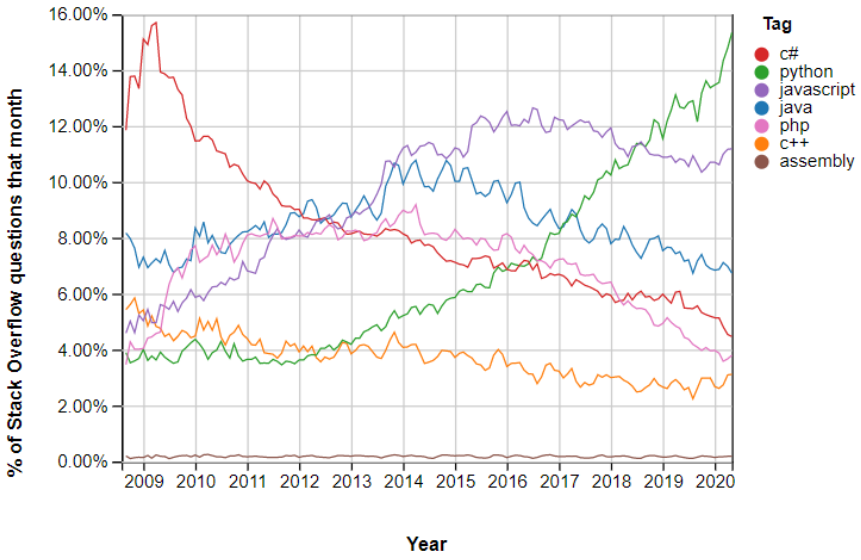

Рис. 2.1 Графік який демонструє зростання популярності Мови Python

Python працює на декількох платформах (Windows, Mac, Linux, Raspberry Pi тощо) і має простий англійський синтаксис, що дозволяє розробникам писати програми з меншою кількістю рядків коду, ніж інші мови програмування. Python можна вважати процедурною, об'єктно-орієнтованою або функціональною мовою. Однією з основних причин, чому Python використовується для машинного навчання, є те, що вона має багато фреймворків, які спрощують процес кодування та скорочують час розробки. Працюючи з Python, програмісту не потрібно приділяти великої уваги безпосередньому написанню коду: він може зосередити всю свою увагу на вирішенні більш складних проблем, пов’язаних з машинним навчанням. Простий синтаксис Python допомагає програмісту перевірити складні алгоритми з мінімальним часом реалізації. Ще однією перевагою Python у машинному навчанні є його гнучкість: наприклад, програміст має вибір між об’єктно-орієнтованим підходом та сценаріями. Python допомагає поєднувати різні типи даних. Для того, чтобі стати фахівцем в Machine learning, потрібно знати не тільки як систематизувати та аналізувати дані, як працюють базові алгоритми, вивчати реальні кейси, а й знати мову програмування Python, розібратися буде простіше. Вищезазначені фактори пояснюють, чому Python так широко використовується у машинному навчанні, його простота та гнучкість допомагає працювати над складними алгоритмами з мінімальним часом реалізації[24].

Python у штучному інтелекті актуальний у найрізноманітніших галузях. У медицині за допомогою цієї мови програмування оцінюють результати обстеження для раннього виявлення захворювань. У сфері фінансів Python використовується для прогнозування ринкових трендів і автоматизації торгових стратегій. Що стосується штучного інтелекту в маркетингу, то Python знадобиться для аналізу поведінки споживачів і розробки персоналізованих рекламних кампаній. Ця мова програмування також знаходить застосування в ігровій індустрії, адже, впроваджуючи штучний

інтелект в ігри, розробники створюють складніші й реалістичніші сюжети. Але й це ще не все.

Обробка природної мови (NLP). Python активно використовується для розробки систем, які здатні аналізувати, розуміти й генерувати текст. Це потрібно для створення чатботів, автоматичних перекладачів тощо.

Аналіз соціальних мереж. Python застосовується для збору й аналізу даних із соціальних мереж, щоб виявляти тренди, відстежувати репутацію брендів і аналізувати взаємодію користувачів. Так штучний інтелект допомагає просувати товари й послуги[25].

Автоматизація виробничих процесів. Python у штучному інтелекті застосовується для автоматизації виробничих процесів і моніторингу обладнання. Алгоритми машинного навчання прогнозують поламання обладнання та оптимізують роботу виробничих ліній.

Кібербезпека. Python використовують для розробки систем штучного інтелекту, які виявляють та усувають кіберзагрози. Моделі машинного навчання допомагають зафіксувати аномалії в мережевому трафіку й автоматично реагувати на потенційні ризики[12].

Ці приклади демонструють, наскільки широкий спектр можливостей має Python у штучному інтелекті. Завдяки потужним інструментам та бібліотекам можна ефективно впроваджувати AI-технології в найрізноманітніші сфери життя. Сьогодні це не тільки рішення для професіоналів галузі штучного інтелекту, але й шанс для молоді розпочати свою кар’єру. Що раніше ви зануритесь у світ програмування, то більше у вас шансів досягти успіху в цій перспективній галузі!

Величезна кількість бібліотек є однією з основних причин, через яку Python є найпопулярнішою мовою програмування, що використовується для штучного інтелекту. Бібліотека — це модуль або група модулів, які включають попередньо написаний фрагмент коду. Бібліотеки Python надають безліч елементів базового рівня, щоб розробникам не потрібно було кодувати їх із самого початку щоразу. ML вимагає безперервної обробки даних, а

бібліотеки Python якраз дозволяють отримувати доступ до даних, обробляти та перетворювати їх. Нижче ділимося з вами добіркою найкращих бібліотек для роботи з AI та ML.

Низький бар'єр входу. Працювати у сфері машинного навчання та штучного інтелекту означає постійно мати справу з купою даних, які потрібно встигати обробляти найзручнішим та найефективнішим способом. Бар'єр низького входу дозволяє більшій кількості фахівців з обробки даних швидко освоїти Python і почати використовувати мову, не витрачаючи надто багато зусиль на її вивчення. Оскільки Python нагадує повсякденну англійську, розробники можуть одразу працювати зі складними системами.

Гнучкість. Python пропонує можливість вибрати використання ООП або скриптів, не потребує перекомпілювання вихідного коду, дозволяє поєднувати Python з іншими мовами для досягнення своїх цілей. Крім того, гнучкість мови дозволяє розробникам вибирати стилі програмування, які їм комфортні (імперативний, функціональний, об'єктно-орієнтований, процедурний та інші) або навіть комбінувати їх.

2.2 Опис бібліотек для написання моделі

Бібліотеки з відкритим вихідним кодом Python — це лише одна з причин, чому ця мова є популярною для завдань машинного навчання та штучного інтелекту. Python також вирізняється своєю універсальністю та гнучкістю, оскільки його можна поєднувати з іншими мовами програмування, коли це необхідно для досягнення кращих результатів. Крім того, Python працює на більшості операційних систем і платформ, що робить його доступним і надійним вибором для розробки у різних середовищах.

Хоча впровадження глибоких нейронних мереж і складних алгоритмів машинного навчання може бути дуже ресурсоємким і вимагати багато часу,

Python пропонує численні пакети та інструменти, які значно пришвидшують цей процес. Завдяки широкому спектру готових рішень для машинного навчання, таких як TensorFlow, PyTorch, і scikit-learn, розробники можуть фокусуватися на вирішенні конкретних завдань, а не на розробці інфраструктури.

Python також є мовою об'єктно-орієнтованого програмування (ООП), що дозволяє ефективно структурувати та категоризувати дані. Це забезпечує зручність і масштабованість, дозволяючи організувати код у вигляді об'єктів і класів, що значно полегшує його підтримку та розширення. Всі ці фактори роблять Python ідеальним вибором для тих, хто працює в галузі машинного навчання та штучного інтелекту.

NumPy

NumPy є однією з найкращих бібліотек Python для машинного навчання та штучного інтелекту. Це потужний інструмент з відкритим вихідним кодом, який широко використовується для виконання різноманітних математичних операцій над масивами і матрицями. NumPy є однією з найпопулярніших наукових бібліотек, на яку покладаються багато дослідників і розробників для обробки та аналізу даних.

Масиви NumPy є набагато ефективнішими у порівнянні з традиційними списками Python, оскільки займають менше пам'яті і працюють значно швидше, що робить їх ідеальними для обробки великих обсягів даних. Вони також дозволяють легко виконувати операції, такі як транспонування, зміна форми масивів та маніпулювання даними, що підвищує гнучкість роботи з ними.

Завдяки своїй простоті та ефективності, NumPy є чудовим вибором для оптимізації продуктивності моделей машинного навчання, допомагаючи зменшити складність обробки даних та підвищити ефективність обчислень. Це дозволяє розробникам зосередитися на створенні і тренуванні моделей,

замість того, щоб витрачати час на оптимізацію базових математичних операцій.

Основні функції бібліотеки:

Array - функція створення масиву зі списку. Функція array приймає два аргументи: список, який буде конвертований в масив, та тип елемента масиву. Доступ до елементів масиву здійснюється так само, як і до елементів списку

Array Slicing (нарізання масиву) - працює з багатовимірними масивами за аналогією з одновимірними. Кожен зріз застосовується як фільтр встановленого виміру. Якщо потрібно вказати, що використовуються всі елементи даного вимірювання, використовуйте.

Reshape - створення багатовимірного масиву можна створювати з одномірного. Це дозволяє уникнути складної нотації із комами при завданні нових масивів.

Seaborn

Seaborn — це бібліотека для створення статистичної графіки на Python. Він будується на основі matplotlib і тісно інтегрується зі структурами даних pandas.

Seaborn допомагає вам досліджувати та розуміти свої дані. Його функції побудови графіків працюють із кадрами даних і масивами, що містять цілі набори даних, і внутрішньо виконують необхідне семантичне відображення та статистичне агрегування для створення інформативних графіків. Його орієнтований на набори даних декларативний API дозволяє зосередитися на тому, що означають різні елементи ваших графіків, а не на деталях їх малювання.

Seaborn — це єдина бібліотека, яку нам потрібно імпортувати для цього простого прикладу. За домовленістю він імпортується зі скороченням sns.

За лаштунками seaborn використовує matplotlib для малювання своїх сюжетів. Для інтерактивної роботи рекомендується використовувати інтерфейс Jupyter/IPython у режимі matplotlib, інакше вам доведеться викликати matplotlib.pyplot.show(), коли ви хочете побачити графік.

Pandas

Pandas — це потужна бібліотека для аналізу та маніпулювання даними в Python, яка широко використовується для обробки табличних, часових рядів та інших структурованих даних. Вона забезпечує високий рівень абстракції та зручні інструменти для роботи з даними, такі як обробка відсутніх значень, групування даних, злиття та приєднання таблиць, агрегація та перетворення даних.

Бібліотека Pandas пропонує швидкий і ефективний спосіб керування та дослідження даних, надаючи серії та кадри даних, які ефективно представляють дані, а також маніпулюють ними різними способами.

TensorFlow

Ще одна безкоштовна бібліотека Python із відкритим вихідним кодом, TensorFlow, спеціалізується на диференційованому програмуванні. Бібліотека складається з набору інструментів і ресурсів, які дозволяють початківцям і професіоналам будувати моделі DL і ML, а також нейронні мережі[14].

TensorFlow складається з гнучкої архітектури та фреймворку, що дозволяє працювати на різних обчислювальних платформах, таких як CPU та

GPU. З огляду на це, він працює найкраще, коли працює на тензорному процесорі (TPU). Бібліотека Python часто використовується для реалізації навчання з підкріпленням у моделях ML і DL, і ви можете безпосередньо візуалізувати моделі машинного навчання.

Matplotlib

Matplotlib — це об’єднання NumPy і SciPy, і його було розроблено, щоб замінити потребу у використанні власної статистичної мови MATLAB. Комплексна безкоштовна бібліотека з відкритим кодом використовується для створення статичних, анімованих та інтерактивних візуалізацій у Python.

Бібліотека Python допомагає зрозуміти дані перед тим, як перенести їх на обробку даних і навчання для завдань машинного навчання. Він покладається на набори інструментів Python GUI для створення графіків і графіків за допомогою об’єктно-орієнтованих API. Він також надає інтерфейс, схожий на MATLAB, тому користувач може виконувати подібні завдання, що й MATLAB.

Scikit-learn

Бібліотека Scikit-learn – найпоширеніший вибір для вирішення завдань класичного машинного навчання. Вона надає широкий вибір алгоритмів навчання з учителем та без вчителя.

Навчання з учителем передбачає наявність розміченого датасета, у якому відомо значення цільової ознаки. У той час як навчання без вчителя не передбачає наявності розмітки в датасеті - потрібно навчитися отримувати корисну інформацію з довільних даних.

Однією з основних переваг бібліотеки є те, що вона працює на основі декількох поширених математичних бібліотек, і легко інтегрує їх один з одним.

Scikit-learn широко використовується для промислових систем, в яких застосовуються алгоритми класичного машинного навчання, для досліджень, а також для новачків, які тільки роблять перші кроки в галузі машинного навчання[13].

2.3 Методи аналізу часових рядів

Часовий ряд (time series) — це впорядкована послідовність вимірювань певної змінної, проведених через регулярні або нерегулярні інтервали часу. Характерною ознакою часових рядів є їхня залежність від хронологічного порядку, тобто кожне спостереження має часову мітку, що визначає його положення у послідовності. Приклади часових рядів включають температуру повітря, виміряну щогодини протягом доби; щоденний обсяг продажів товарів; щорічну врожайність зернових культур; динаміку біржових котирувань акцій; обмінний курс валют; та рівень інфляції[21].

Головною особливістю часових рядів є те, що порядок спостережень є суттєвим для їхнього аналізу, оскільки час відіграє ключову роль у розумінні структури даних. Це вирізняє часові ряди з-поміж випадкових вибірок, де порядок спостережень не має значення.

Рівні часового ряду часто залежать один від одного. Майбутні значення змінної можуть бути функцією від її минулих значень. Закон розподілу ймовірностей значень часового ряду змінюється з часом, а математичні очікування та дисперсії величин також можуть бути залежними від часової компоненти.

Часовий ряд зазвичай розглядають як комбінацію кількох компонент: тренд (trend) — довгострокова тенденція змін, яка може бути зростаючою, спадною або стабільною; сезонна складова — регулярні періодичні коливання, що повторюються з певною частотою (наприклад, щомісячні чи щорічні); циклічна складова — коливання, які відбуваються протягом тривалих періодів і мають непостійні амплітуду та частоту.

Для виявлення тренда використовуються методи регресійного аналізу або ковзного середнього. Аналіз сезонної компоненти здійснюється за допомогою моделей сезонного згладжування, авторегресії або сезонної декомпозиції. Циклічні коливання досліджують за допомогою спектрального аналізу або гармонійних моделей.

Згладжування часових рядів є ключовим етапом аналізу, спрямованим на зменшення впливу випадкових коливань і виокремлення основних закономірностей у даних. Для цього застосовують такі методи, як фільтрування (зокрема, за допомогою перетворення Фур'є), експоненціальне згладжування, а також кускову апроксимацію із використанням многочленів.

Прогнозування часових рядів базується на припущенні про збереження виявлених закономірностей у майбутньому. Цей підхід є особливо ефективним у короткостроковій перспективі, коли вплив зовнішніх чинників мінімальний. Однак методи прогнозування часових рядів мають обмеження: вони працюють за умови стабільного розвитку і не здатні враховувати різкі зміни в динаміці[16].

Часові ряди зазвичай включають кілька основних компонентів: тренд, сезонність і циклічність. Тренд відображає загальний напрямок зміни даних у часі, тоді як сезонні складові характеризуються періодичними змінами з регулярними інтервалами — наприклад, днями, тижнями чи місяцями. Циклічні коливання, на відміну від сезонних, є менш регулярними та тривалішими, їхня тривалість може варіюватися від кількох років до десятиліть. Розуміння цих складових дозволяє ефективніше моделювати часові ряди, виділяти ключові тенденції та прогнозувати майбутній розвиток.

До основних методів аналізу часових рядів належать:

Метод ковзного середнього.

Експоненціальне згладжування.

Проектування тренда.

Ці методи дозволяють ефективно оцінювати часові ряди, розділяти їх на складові та використовувати отримані дані для моделювання та прогнозування майбутніх тенденцій.

Метод рухомого (ковзного) середнього

Ковзна середня (Moving Average, MA) — це популярний технічний індикатор, який відображає середнє значення ціни активу за певний період часу. Основною метою ковзної середньої є згладжування короткострокових цінових коливань, щоб виділити довгострокові тенденції. Завдяки цьому індикатору трейдери можуть більш ефективно визначати загальний напрямок ринку, аналізувати імпульси та виявляти ключові рівні підтримки й опору.

Однією з ключових переваг ковзної середньої є її здатність усувати ринковий шум — тобто випадкові цінові коливання, які не мають значного впливу на загальну картину. Це дозволяє трейдерам краще розпізнавати реальні тренди та уникати хибних сигналів, які можуть вводити в оману під час прийняття рішень[21].

Метод ковзної середньої заснований на принципі згладжування. Ідея полягає в тому, щоб виключити вплив дрібних і непостійних змін цін, надаючи пріоритет загальній тенденції. Цей інструмент широко використовується у криптотрейдингу для:

Ідентифікації трендів: Визначення, чи перебуває ринок у висхідному (бичачому) або низхідному (ведмежому) русі.

Пошуку торгових сигналів: Наприклад, перетин ціни з ковзною середньою або перетин двох ковзних середніх з різними періодами може вказувати на зміну тренду.

Визначення рівнів опору та підтримки: Ковзна середня часто виступає як динамічний рівень, до якого ціна прагне повернутися.

Ковзна середня може бути різних видів, залежно від способу розрахунку:

Проста ковзна середня (SMA, Simple Moving Average): Це середнє значення цін за обраний період. Наприклад, 20-денна SMA обчислюється шляхом додавання цін закриття за 20 днів і поділу на 20.

Експоненціальна ковзна середня (EMA, Exponential Moving Average): Надає більше ваги останнім даним, що робить її більш чутливою до нових змін ціни.

Зважена ковзна середня (WMA, Weighted Moving Average): Враховує вагові коефіцієнти, де останні ціни мають більшу вагу, ніж попередні.

Таким чином, ковзна середня є одним із базових, але дуже ефективних інструментів для аналізу ринку, який допомагає трейдерам приймати більш обґрунтовані рішення щодо купівлі, продажу чи утримання активів.

Просте ковзне середнє (SMA) — один із найпростіших і найпоширеніших індикаторів у технічному аналізі. Воно обчислюється як середнє арифметичне значення цін активу за обраний період часу. SMA відноситься до групи трендових індикаторів і допомагає визначати як початок, так і завершення тенденції. Кут нахилу лінії SMA дає змогу оцінити швидкість руху ціни, а також визначити силу тренду. Завдяки своїй простоті, SMA часто слугує основою для створення більш складних технічних індикаторів. Інколи його називають "лінією тренду". Формула простого ковзного середнього(1):

$$SMA = \frac{\sum_{i=1}^{n} P_i}{n} \quad (1)$$

де Pi – ціни на ринку;

n — основний параметр, що визначає період SMA, тобто кількість цін, які враховуються у розрахунку. Цей параметр іноді називають "довжиною згладжування" або "порядком середнього".

Просте ковзне середнє є звичайним середнім арифметичним від цін за певний період. SMA являє собою якийсь показник ціни рівноваги за певний період, чим коротше SMA, тим за менший період береться рівновага. Усереднюючи ціни, воно завжди слідує за головною тенденцією ринку, фільтруючи дрібні коливання.

SMA фільтрує незначні цінові коливання, дозволяючи трейдерам зосередитися на основній тенденції. Коротке SMA (з меншим nnn) швидше реагує на зміну ціни, що робить його корисним для визначення швидких змін у тренді натомість SMA (з більшим nnn) згладжує ціну на триваліших інтервалах, мінімізуючи вплив короткострокових коливань.

Через свою конструкцію SMA завжди запізнюється відносно реальних змін ціни. У трендових стратегій це може призводити до втрати частини прибуткового руху на вході й виході з ринку. SMA надає однакову вагу як старим, так і новим цінам. Це є недоліком, оскільки нові дані зазвичай більш релевантні для поточної ринкової ситуації. Ця проблема частково вирішується використанням експоненціального ковзного середнього (EMA), яке враховує ваги для нових цін.

Зважене ковзне середнє (англ. Weighted Moving Average – WMA). Одним з недоліків SMA є присвоєння при його розрахунку всім цінам однакових ваг при усередненні незалежно від того, ближче чи далі вони від поточного моменту. Цей недолік усунуто у зваженому ковзному середньому. WMA, таким чином, є звичайною модифікацією SMA з вагами підібраними так, що останні ціни мають більшу вагу[9].

Зважене ковзне середнє визначається за формулою (2):

$$WMA = \frac{\sum_{i=1}^{n} P_i * W_i}{\sum_{i=1}^{n} W_i} \quad (2)$$

де Pi – значення ціни і-періодів тому, (і сьогодні = 1);

Wi – значення ваг для ціни і-періодів тому.

Зважене ковзне середнє являє собою арифметичне зважене коливань цін за певний період. В якості аналітичного інструменту воно знімає частину недоліків звичайного ковзного, але не усуває їх повністю.

Зважене ковзне середнє являє собою арифметичне зважене коливань цін за певний період. В якості аналітичного інструменту воно знімає частину недоліків звичайного ковзного, але не усуває їх повністю.

Експоненційне ковзне середнє (англ. Exponential Moving Average – EMA) зменшує помилку, надаючи більшу вагу останнім цінами у порівнянні з більш далекими цінами. Цей метод дозволяє більш швидко реагувати на поточні зміни ціни в порівнянні з SMA. Вага, що надається останній ціні, залежить від періоду ковзної середньої. Чим коротший період EMA, тим більша вага надаватиметься останньою ціною.

Експоненційне ковзне середнє може бути визначене двома шляхами – як відсоткове ковзне середнє або як періодичне ковзне середнє. Відповідно в відсотковому ковзному, єдиним параметром є вага (відсоток), а в періодній – період КС.

Основна формула виглядає наступним чином (3):

$$EMA = \frac{EMA_{i-1}*(n-1)+2*P_i}{n+1} \quad (3)$$

де EMA – експонентна ковзна середня;

Pi – значення ціни в і-му періоді;

n – період розрахунку;

EMAi−1 – значення EMA попереднього періоду.

Необхідно зазначити, що теоретично в розрахунку цієї ковзної використовуються всі ціни, за весь період її побудови і, незважаючи на те, що

вплив старих цін зникає з часом, він не зникає до кінця. Ефект старих цін зникає швидше для більш коротких EMA, в порівнянні з більш довгими. На реальному графіку різниця між SMA і EMA не дуже велика, хоча і є присутня. Вважається, що експоненційне ковзне все ж краще відображає ринкові ціни при інших рівних умовах, оскільки вплив кожної попередньї ціни убуває експоненційно з його віддаленістю від поточної ціни.

Метод експоненціального згладжування

Метод експоненціального згладжування — це метод аналізу та прогнозування часових рядів, який дозволяє згладити випадкові коливання в даних та виділити основні тенденції. Метод базується на тому, що кожне нове значення має більшу вагу порівняно з попередніми, а вплив спостережень поступово зменшується в часі.

Метод використовує вагову середню, де ваги експоненційно зменшуються для старіших даних. Це означає, що найбільш актуальні спостереження мають більший вплив на оцінки, а давніші — менший.

Формула простого експоненціального згладжування (4):

$$S_t = aY_t + (1 - a)S_{t-1} \quad (4)$$

Де: S_t – згладжене значення моменту часу t; Y_t – фактичне значення моменту часу t; S_{t-1} – згладжене значення для попереднього періоду t-1; a – коефіцієнт згладжуваннння (0 < a < 1).

Коефіцієнт α критичний для точності методу. Якщо α близьке до 1, то більший вплив мають останні спостереження, Результат швидше реагує на зміни в даних, але може бути чутливим до шуму. Якщо α близьке до 0, то враховуються майже всі попередні дані, згладжування повільне, результат менш чутливий до коливань.

Метод проектування тренда

Метод проєктування тренда є основним інструментом аналізу часових рядів і часто використовується як частина більш складних методів прогнозування, які враховують сезонність та інші компоненти.

Метод проєктування тренда — це один із підходів до прогнозування часових рядів, який базується на припущенні, що основна тенденція в даних буде продовжуватися у майбутньому. Метод використовується, коли дані демонструють стабільний тренд і незначний вплив сезонних або циклічних коливань. Метод проєктування тренда полягає у знаходженні математичної моделі, яка описує залежність досліджуваного показника від часу (наприклад, лінійної, експоненціальної або степеневої функції). Ця модель використовується для екстраполяції (прогнозування) значень у майбутньому.

Рівняння тренду має вигляд:

Лінійний тренд:

$$y_t = a + b * t \quad (5)$$

Експоненціальний тренд:

$$y_t = a * e^{b*t} \quad (6)$$

Степеневий тренд:

$$y_t = a * t^b \quad (7)$$

де t – час, a і b параметри які визначаються на основі даних.

Після цього проводиться аналіз адекватності моделі в ході якого оцінюють, наскільки добре обраний тренд відповідає реальним даним. Для цього використовуються коефіцієнт детермінації. Здійснюється перевірка залишків (випадковість, нормальність, незалежність). Якщо модель недостатньо точна, вибирається інший тип функції тренду.Далі проводять екстраполяція тренду шляхом обчислення прогнознів значення для майбутніх періодів. Після цього іде інтерпретація результатів та остаточна оцінка

Висновок за розділом 2

У другому розділі було розглянуто та обрано основні методи та засоби для подальшої розробки моделі машинного навчання. Було обрано мову написання та бібліотеки які будуть використані під час розробки, також були розглянуті основні методи прогнозування часових рядів що також знадобится у ході розробки.

РОЗДІЛ 3

РОЗРОБКА ТА НАЛАШТУВАННЯ МОДЕЛІ МАШИННОГО НАВЧАННЯ

3.1 Підготовка даних до роботи з моделью машинного начання

Спершу підключемо до проекту усі необхідні біліотеки (рис. 3.1)

```
import numpy as np
import pandas as pd

+ Code

import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
from sklearn.model_selection import train_test_split
```

Рис. 3.1 Список бібліотек використаних під час розробки

Після підключення усіх необхідних бібліотек перейдемо до обробки даних. Для зчитування даних з дата сету використаємо команду `pd.read_csv` (рис. 3.2).

```
# load the data that is needed into the application
df = pd.read_csv('water_potability.csv')
df.head()
```

Рис. 3.2 Зчитування даних з датасету

У проекті використовується бібліотека параметрів якості води з відкритих джерел.

Після зчитування даних з дата сету створену кореляційну матрицю за допомогою `sns.heatmap` для виявлення кореляцій між параметрами близькість до одиниці означає сильно виражену позитивну лінійну залежність (рис. 3.3).

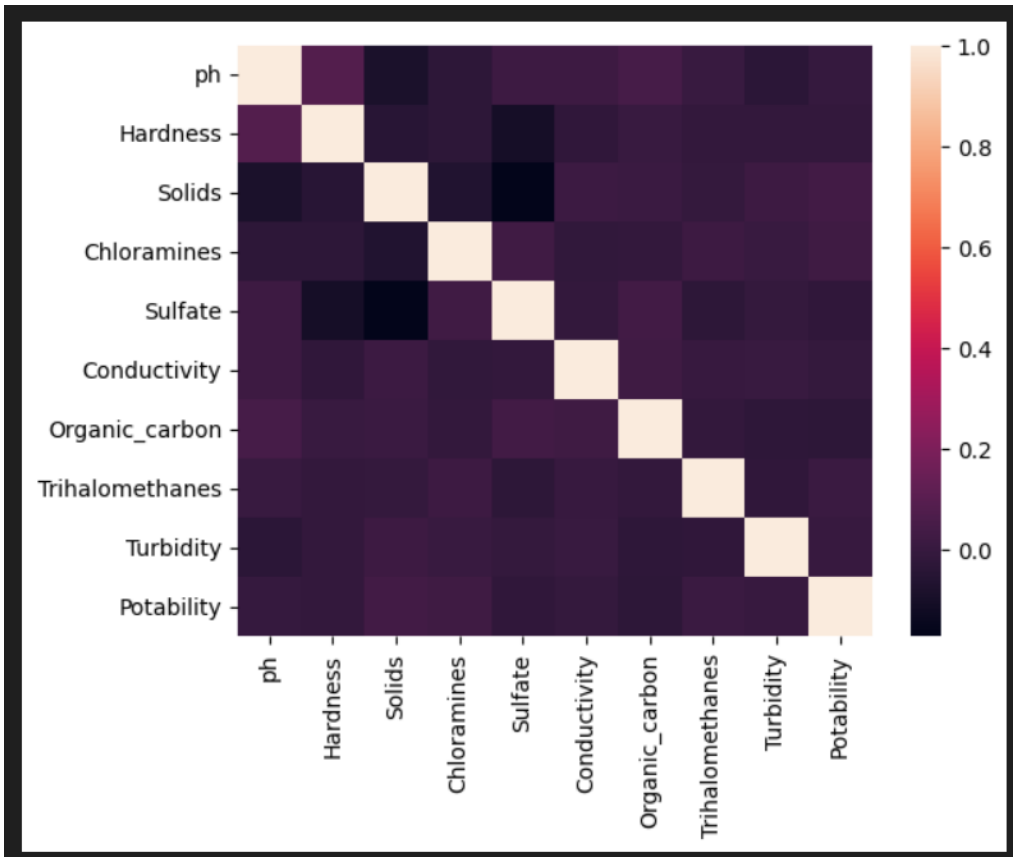

Рис. 3.3 Кореляційна матриця

3.2 Аналіз даних

Після цього проводимо аналіз даних на наявність загублених значень (рис. 3.4).

```
missing_data = df.isnull().sum()
total = df.isnull().count()
percent = (missing_data/total) * 100
missing_data = pd.concat([missing_data, percent], axis=1, keys=['Total', 'Percent'])
missing_data
```

Рис. 3.4 Виявлення прогалин у дата сеті

Розглянемо поетапно кожну строчку коду, щоб зрозуміти, як працює цей фрагмент і для чого він використовується.

`missing_data = df.isnull().sum ()` - команда обчислює кількість відсутніх (null) значень у кожному стовпці DataFrame df. `df.isnull()` повертає новий DataFrame того ж розміру, де кожен елемент містить True, якщо значення відсутнє (NaN), і False, якщо значення є. .sum() підсумовує True (які в Python дорівнюють 1) по кожному стовпцю, щоб знайти загальну кількість відсутніх значень.

Отримується серія де індекс відповідає назві стовпця, а значення — кількість пропущених даних у ньому.

`total = df.isnull().count()` - підраховує загальну кількість елементів у кожному стовпці DataFrame df. df.isnull() повертає DataFrame такого ж розміру, як і df, але з логічними значеннями. count() обчислює кількість ненульових елементів (не обов'язково NaN) у кожному стовпці. Однак тут підраховуються всі значення (оскільки isnull() лише створює копію DataFrame, і `.count()` повертає кількість усіх рядків у кожному стовпці).

Отримується серія (Series), яка показує загальну кількість значень (включаючи пропущені) у кожному стовпці.

`percent = (missing_data/total) * 100` - Обчислює відсоток відсутніх значень у кожному стовпці. `missing_data` містить кількість пропущених значень у кожному стовпці. total містить загальну кількість значень у кожному стовпці. Поділ `missing_data / total` обчислює частку пропущених значень відносно загальної кількості. Множення на 100 переводить результат у відсотки.

Отримується серія (Series), яка показує відсоток пропущених даних у кожному стовпці.

`missing_data = pd.concat([missing_data, percent], axis=1, keys=['Total', 'Percent'])` - Об’єднує кількість пропущених значень (missing_data) та їх відсоток (percent) у єдиний DataFrame. pd.concat() об'єднує два об'єкти (в даному випадку серії missing_data і percent). axis=1 вказує, що об'єднання відбувається по стовпцях (тобто серії стають стовпцями нового DataFrame). `keys=['Total', 'Percent']` задає назви стовпців у новому DataFrame.

`missing_data` - отримується DataFrame, де: Перший стовпець (Total) містить кількість відсутніх значень у кожному стовпці вихідного DataFrame df. Другий стовпець (Percent) містить відсоток відсутніх значень.

Після виконання вищенаведеного коду отримуємо таблицю з відсотковими значеннями кількості загублених даних у дата сеті (рис. 3.5).

	Total	Percent
ph	491	14.987790
Hardness	0	0.000000
Solids	0	0.000000
Chloramines	0	0.000000
Sulfate	781	23.840049
Conductivity	0	0.000000
Organic_carbon	0	0.000000
Trihalomethanes	162	4.945055
Turbidity	0	0.000000
Potability	0	0.000000

Рис. 3.5 Таблиця з відсотковими значеннями кількості загублених даних у дата сеті

Далі заповнюємо прогалини у дата сеті за допомогою методу заповнення даними з сусідніх рядів (рис. 3.6)

```
df['ph'] = df['ph'].fillna(df['ph'].mean())
df['Sulfate'] = df['Sulfate'].fillna(df['Sulfate'].mean())
df['Trihalomethanes'] = df['Trihalomethanes'].fillna(df['Trihalomethanes'].mean())

df.info()
```

Рис. 3.6 Заповнення прогалин у дата сеті

`df['ph'] = df['ph'].fillna(df['ph'].mean())` - команда замінює всі відсутні значення (NaN) у стовпці ph середнім значенням (mean) для цього стовпця. `df['ph']` - вибірка стовпця ph із DataFrame df.; `df['ph'].mean()` - обчислює середнє арифметичне для значень стовпця ph, ігноруючи NaN. `.fillna()`- заповнює всі NaN у стовпці переданим значенням (у цьому випадку середнім значенням). У результаті отримуємо що усі NaN у стовпці ph замінюються на середнє значення цього стовпця.

`df['Trihalomethanes']` - аналогічно до першого рядка, замінює всі відсутні значення (NaN) у стовпці Sulfate середнім значенням для цього стовпця. `df['Sulfate']` - вибірка стовпця Sulfate із DataFrame df. `df['Sulfate'].mean()` - обчислює середнє значення в стовпці Sulfate. `fillna()`- замінює всі NaN у стовпці Sulfate на середнє значення. Усі NaN у стовпці Sulfate замінюються середнім значенням цього стовпця.

`df['Trihalomethanes'].fillna(df['Trihalomethanes'].mean())` - замінює відсутні значення (NaN) у стовпці Trihalomethanes середнім значенням цього стовпця. df['Trihalomethanes']: вибірка стовпця Trihalomethanes із DataFrame df. `df['Trihalomethanes'].mean()`: обчислює середнє значення для стовпця Trihalomethanes. `.fillna()`: замінює всі NaN у стовпці на обчислене середнє.

Усі NaN у стовпці Trihalomethanes замінюються середнім значенням цього стовпця.

`df.info()`- виводить коротку інформацію про DataFrame df, включаючи:

Далі обираємо прогнозуємий показник , у даному проекті був обраний показник приданості води до питного виживання так як він залежить від багатьох факторів що дае змогу на основі провести кореляцію між різними чинниками, а також цей показник є практичним для різних сфер застосування (рис 3.7).

```
# feature selection
X = df.drop('Potability', axis=1)
y = df['Potability']
```

Рис. 3.7 Вибір прогнозуємого показника

Далі масштабуємо дані для пришвидшення навчання моделі (рис. 3.8).

```
# scale the data
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()

X = scaler.fit_transform(X)
```

Рис. 3.8 Масштабування даних

MinMaxScaler - це клас із бібліотеки scikit-learn, який масштабує числові дані в заданий діапазон (за замовчуванням від 0 до 1). Використовується для

нормалізації даних перед передачею їх у моделі машинного навчання. Це корисно, коли моделі чутливі до масштабу даних (наприклад, методи, що базуються на відстані, як SVM або KNN). Створюєтьє об'єкт scaler, який буде використовувати методи .fit() і .transform() для масштабування даних.

`X = scaler.fit_transform(X)` - Масштабує всі числові ознаки (фічі) у масиві X в діапазон від 0 до 1. `scaler.fit_transform(X)` виконує два етапи перший: `fit()` - обчислює мінімальне та максимальне значення для кожної ознаки (колонки) у X; другий: Зберігає ці значення в об'єкті scaler.. `transform()` - Масштабує кожне значення в X за формулою (8):

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (8)$$

Після цього розділяємо дані на тренувальні та тестові (рис. 3.9).

```
# split the data into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Рис. 3.9 Розподіл даних

`train_test_split` - функція яка розділяє дані на навчальну вибірку (для тренування моделі) та тестову вибірку (для оцінки моделі).

Параметри: X - вхідні ознаки (фічі) після масштабування. Y - мітки (цільова змінна). `test_size=0.2` - означає, що 20% даних буде виділено для тестової вибірки, а решта 80% — для навчання.
`random_state=42` - фіксує початковий стан генератора випадкових чисел, щоб забезпечити відтворюваність результатів.

Команда розбиває масиви X та y на чотири частини: `X_train` - вхідні ознаки для навчальної вибірки (80% даних). `X_test` - вхідні ознаки для тестової вибірки (20% даних). `y_train` - мітки для навчальної вибірки.

`y_test` - мітки для тестової вибірки. Тепер є два набори даних: навчальний набір - для тренування моделі та Тестовий набір для оцінки її продуктивності на нових, невідомих даних.

3.3 Написання моделі машинного навчання та її тестування

Далі будуємо модель машинного навчання використовуючи Tensorflow (рис. 3.10)

```
# build the model
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(128, kernel_regularizer=tf.keras.regularizers.l2(0.001), activation='relu', input_dim=X_train.shape[1]),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(64, kernel_regularizer=tf.keras.regularizers.l2(0.001), activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

Рис. 3.10 Код моделі машинного навчання

`Sequential` - це клас із Keras, який дозволяє створювати модель нейронної мережі, додаючи шари послідовно один за одним.

У цьому випадку ми створюємо багатошарову нейронну мережу (MLP - Multi-Layer Perceptron).

Перший шар:

`Dense(128)` - Щільний (fully connected) шар із 128 нейронів. Кожен нейрон отримує вхідні дані від усіх нейронів попереднього шару (або від вхідного шару).

`kernel_regularizer=tf.keras.regularizers.l2(0.001)` - додає L2-регуляризацію до ваг шарів. Це запобігає перенавчанню,

штрафуючи великі ваги в моделі. Гіперпараметр 0.001 визначає силу регуляризації.

`activation='relu'`- використовується функція активації ReLU (Rectified Linear Unit). Вона замінює всі від'ємні значення на 0, залишаючи позитивні незмінними (9):

$$f(x) = \max(0, x) \text{ (9)}$$

`input_dim=X_train.shape[1]` - вказує розмір вхідного шару, який відповідає кількості ознак (фіч) у навчальних даних.

Dropout шар:

Потрібен для регуляції, за рахунок випадкового відключення 30% нейронів у цьому шарі під час тренування. Допомагає запобігти перенавчанню, змушуючи модель працювати краще, навіть якщо деякі нейрони "вимкнені".

Вихідний шар:

Вихідний шар має 1 нейрон, що генерує єдиний вихід (ймовірність). Підходить для задач бінарної класифікації.

`activation='sigmoid'`- використовується сигмоїдна функція активації, яка перетворює вихід у значення в діапазоні [0, 1] (10):

$$f(x) = \frac{1}{1-e^{-x}} \text{ (10)}$$

Загальна структура моделі

Вхідний шар (128 нейронів) - масштабовані дані подаються на вхід. Регуляризація і функція активації ReLU забезпечують ефективне навчання. Прихований шар (64 нейрони) - Зменшується кількість нейронів для більш

компактного представлення, із застосуванням Dropout та L2-регуляризації. Вихідний шар (1 нейрон) - генерується ймовірність, яка використовується для передбачення бінарного класу.

Використання L2-регуляризації та Dropout зменшує ризик перенавчання, тоді як функції активації ReLU та Sigmoid дозволяють ефективно навчатися на даних.

Після цього компілюємо модель (рис. 3.11)

```
optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)
model.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['accuracy'])
```

Рис. 3.11 Налаштування моделі для навчання

Adam (Adaptive Moment Estimation) - один із найпопулярніших алгоритмів оптимізації в глибокому навчанні.

Він поєднує переваги оптимізаторів Momentum і RMSProp, адаптуючи швидкість навчання для кожного параметра моделі на основі середніх значень градієнтів та їхніх квадратів.

`learning_rate=0.001` - Це базова швидкість навчання. Вона визначає, наскільки великі кроки робить модель у напрямку мінімізації функції втрат. Значення 0.001 є типовим стартовим вибором для Adam.

`optimizer=optimizer` - Модель використовуватиме Adam як алгоритм оптимізації, який допомагає налаштувати ваги під час навчання, щоб мінімізувати втрати.

`loss='binary_crossentropy'` - Функція втрат (loss function) у задачах бінарної класифікації (де цільові значення — 0 або 1) зазвичай використовується функція втрат binary_crossentropy.

Формула для функції втрат (11):

$$L = -\frac{1}{N}\sum_{i=1}^{N}[y_i * \log(\tilde{y_i}) + (1 - y_i) * \log(1 - \tilde{y_i})] \quad (11)$$

Де: y_i – істинне значення; $\tilde{y_i}$ – передбачена ймовірність; N – кількість зразків у міні-партії.

`binary_crossentropy` - штрафує модель, якщо її передбачення відхиляються від істинних міток, стимулюючи її навчитися точніше класифікувати дані.

`metrics=['accuracy']` - оцінює частку правильно передбачених міток серед усіх зразків (12):

$$Accuracy = \frac{\text{Кількість правильних передбачень}}{\text{Загальна кількість зразків}} \quad (12)$$

Метрика не впливає на процес навчання, але використовується для моніторингу продуктивності моделі на навчальних і тестових даних.

Далі налаштовуємо параметри для припинення навчання моделі (рис. 3.12)

```
# callbacks
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
```

Рис. 3.12 Налаштування припенння навчання моделі за допомогою зворонього виклику

Early stopping — метод, який автоматично зупиняє навчання моделі, коли її продуктивність на валідаційних даних перестає покращуватися, щоб запобігти перенавчанню. Якщо модель не покращує свою продуктивність (тобто, не зменшується значення функції втрат) на валідаційних даних, алгоритм припиняє навчання на певній ітерації.

Параметри:

`monitor='val_loss'` - означає, що ми спостерігаємо за змінами функції втрат на валідаційних даних (val_loss).

Під час кожної епохи тренування TensorFlow буде оцінювати модель на валідаційному наборі даних, і якщо значення втрат на цьому наборі не покращуються, навчання буде припинено. Вибір val_loss (втрата на валідаційних даних) замість loss на тренувальних даних важливий, оскільки це допомагає оцінити, чи не виникає перенавчання.

`patience=10` - це кількість епох, протягом яких модель може "терпіти" без покращення на валідаційних даних перед тим, як зупинити навчання. Наприклад, якщо значення `val_loss` не зменшується протягом 10 епох, навчання буде припинено.

Чим більше значення patience, тим більше часу модель має для пошуку оптимальних параметрів, але це також збільшує ризик перенавчання.

`restore_best_weights=True` - коли навчання зупиняється через early stopping, параметр `restore_best_weights=True` гарантує, що модель буде повернена до найкращих ваг, які вона мала на момент мінімальної функції втрат на валідаційних даних.

Це важливо, тому що модель може почати перенавчатися після деякого моменту, і, повернувшись до кращих ваг, ми отримаємо найкращу можливу модель, яка досягла мінімуму на валідаційних даних.

Далі відбувається тренування моделі (рис. 3.13)

```
# Train the model
history = model.fit(X_train, y_train, epochs=200, batch_size=32, validation_data=(X_test, y_test), callbacks=[early_stopping])
```

Рис. 3.13 Тренування моделі

Параметри:

`X_train` — це вхідні дані для тренувальної вибірки (наприклад, ознаки, що описують об'єкти, які ви намагаєтеся класифікувати або прогнозувати).

`y_train` — це мітки або значення, відповідні кожному зразку в X_train. У вашому випадку, ймовірно, це залежна змінна, яку ви намагаєтесь передбачити (наприклад, бінарні класи для класифікації).

`epochs=200`- Кількість епох (повних ітерацій) навчання. Це вказує, скільки разів модель повинна пройти через весь тренувальний набір даних. У даному випадку, тренування буде продовжуватися до 200 епох, якщо early stopping не зупинить процес раніше.

`batch_size=32`- Кількість прикладів, які використовуються для кожного кроку оновлення ваг під час тренування.

В даному випадку, тренувальна вибірка розділяється на "пакети" (batches) по 32 зразки на кожен крок. Малий розмір партії (batch) може прискорити навчання, але також може призвести до менших коливань в оновленні ваг. Вибір розміру пакета залежить від розміру вашої вибірки і ресурсів.

`validation_data=(X_test, y_test)` - це дані, що використовуються для оцінки моделі після кожної епохи тренування.

`X_test` — вхідні дані для тестової вибірки (вони не використовуються для тренування, а тільки для оцінки результатів).

`y_test` — це правильні мітки для тестової вибірки. Під час тренування після кожної епохи модель буде оцінювати свою продуктивність на тестових даних, що дозволяє стежити за тим, чи покращується модель на нових, невідомих даних.

`callbacks=[early_stopping]`- Це список зворотних викликів, який в даному випадку містить один елемент: early_stopping.

Як уже пояснювалося, early_stopping зупинить тренування, якщо модель не показує покращення на валідаційних даних протягом кількох епох, щоб запобігти перенавчанню.

Результат:

Виклик model.fit() розпочне процес тренування нейронної мережі. Він буде виконувати тренування на тренувальних даних (X_train, y_train) впродовж до 200 епох або до того часу, поки early stopping не зупинить навчання. Кожен цикл навчання (епоха) модель оцінюється на тестових даних (X_test, y_test). Всі метрики (зокрема, точність) та значення втрат будуть зберігатися у змінній history.

Після цього рядка history міститиме історію тренування, включаючи значення функції втрат (loss) та метрики (наприклад, точність) для кожної епохи.

Після тренування моделі проводиться порівння значень втрат перевірки та навчання (рис. 3.14, 3.15).

```python
# Summary of the process and results
import matplotlib.pyplot as plt

# Plot training & validation loss values
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['Train', 'Validation'], loc='upper left')
```

Рис. 3.14 Код для побудови графіку порівняння втрати перевірки та навчання.

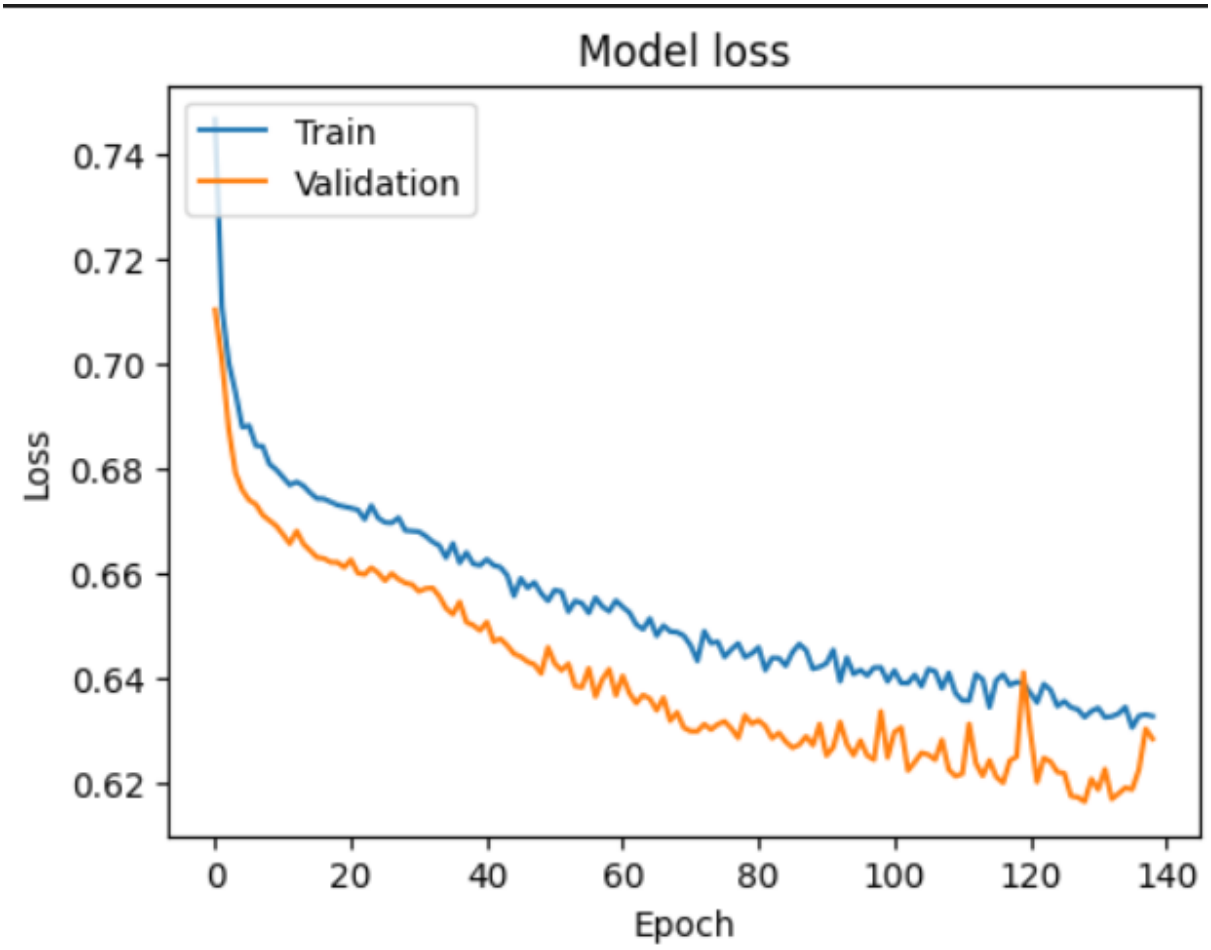

Рис. 3.15 Графік порівння втрат навчання та перевірки

Як можна бачити з отриманих графіків втрати навчання та на перевірку мають незначні розбізножсті, що показує праивльність роботи моделі.

Далі проведемо прогнозування за дпомогою навченої моделі та проаналізуємо ефективність роботи моделі за допомогою матриці плутання (рис. 3.16, 3.17).

```
#  make some predictions
predictions = model.predict(X_test)

# convert the predictions to binary
predictions = np.round(predictions)

from sklearn.metrics import classification_report, confusion_matrix

print(classification_report(y_test, predictions))
print(confusion_matrix(y_test, predictions))

from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy}")

from sklearn.metrics import precision_score
precision = precision_score(y_test, predictions)
print(f"Precision: {precision}")
```

Рис. 3.17 Код для реалізації оцінкм ефективності роботи моделей

`predictions` - отримує прогнозовані значення для тестових даних (X_test). В результаті ви отримуєте масив значень між 0 і 1 (оскільки вихідний шар вашої моделі має функцію активації sigmoid). Ці значення показують ймовірність того, що кожен зразок належить до позитивного класу (якщо клас бінарний).

`np.round(predictions)` - перетворює прогнозовані ймовірності в бінарні значення, використовуючи функцію np.round().

Якщо значення прогнозу більше або дорівнює 0.5, воно округляється до 1 (позитивний клас); якщо менше 0.5 — округляється до 0 (негативний клас).

`classification_report` генерує звіт, що містить кілька метрик для оцінки класифікації: precision (точність) — це частка правильних позитивних передбачень серед усіх передбачених позитивних класів; recall (відгук) — це частка правильних позитивних передбачень серед усіх реальних позитивних класів; f1-score — середнє гармонійне між точністю та відгуком; support — кількість справжніх зразків кожного класу.

`confusion_matrix` виводить матрицю сплутаних результатів, яка показує, скільки разів модель правильно або неправильно передбачила кожен клас.

`accuracy_score` обчислює загальну точність моделі — відношення правильних прогнозів (як для позитивних, так і для негативних класів) до загальної кількості зразків. Це дуже важлива метрика, що дає уявлення про загальну ефективність моделі.

`precision_score` обчислює точність моделі, тобто частку правильних позитивних прогнозів серед усіх передбачених позитивних класів.

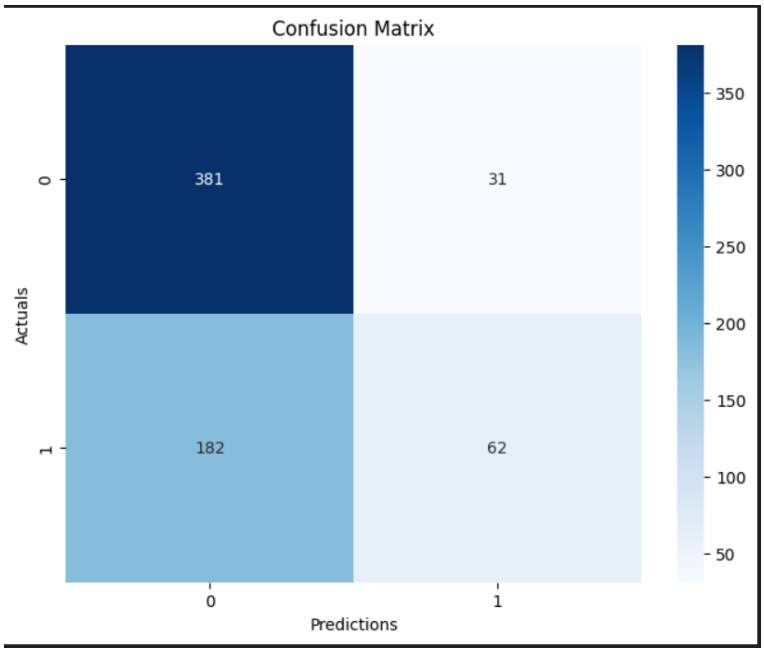

Рис. 3.18 Матриця плутання

True Positive (нижній правий вугол): прогнозовані значення спрогнозовані правильно як позитивні.

False Posititve (верхній правий вугол): прогнозовані значення спрогнозовані невірно (від'ємні значення спрогнозовані як додатні).

False Negative (нижній лівий вугол):: додатні значення спрогнозовані як від'ємні.

True Negative (верхній лівий вугол): прогнозовані значення правильно спрогнозовані як від'ємні.

Проаналізувавши отриману матрицю ми може констатувати що модель працює коректно, так як кількість правильно спрогонозованих значень перевищує кількість невірно спрогнозованих значень.

Виснововк до розділу 3

У даному розділі була проведена повна розробка моделі машинного навчання. Були проаналізовані та підготовлені дані, також була реалізована т модель машинного навчання яка була протестована. Була проведена оцінка роботи моделі за результатами якої можна свідчити що модель працю коректно.

РОЗДІЛ 4

ОХОРОНА ПРАЦІ ТА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

4.1 Охорона праці при роботі з комп'ютерною технікою

Сьогодні персональні комп'ютери (ПК) займають ключову роль у різних сферах діяльності людини. Вони виступають або як об'єкт праці, або як основний інструмент для виконання завдань. Завдяки своїй універсальності та функціональності, ПК стрімко впроваджуються:

На виробництві вони забезпечують ефективність роботи, зменшуючи час і ресурси, необхідні для виконання завдань.

В адміністративно-громадських активно використовують ПК для доступу до інформації, навчання та вирішення повсякденних завдань.

Завдяки таким впровадженням, ПК стають важливою частиною сучасного способу життя, розширюючи можливості людини як у професійній, так і в особистій діяльності.

Інтернет поступово стає основним засобом спілкування між людьми, значно розширюючи можливості взаємодії у професійній та особистій сферах. У перші роки впровадження персональних комп'ютерів (ПК) їх основним призначенням було вирішення складних задач у програмуванні, управління великими базами даних та робота у видавничих системах. На той час комп'ютери сприймалися як зручні та ефективні інструменти для автоматизації і оптимізації праці.

Однак, питання впливу ПК на здоров'я користувачів залишалося поза увагою. Лише з 1990-х років почали з'являтися дослідження, які вказували на те, що інтенсивна робота з ПК може викликати різноманітні захворювання.

В умовах масового використання ПК проблема впливу на здоров'я набуває все більшої актуальності. Вирішення цієї проблеми вимагає комплексного підходу: впровадження сучасних ергономічних стандартів,

інформування користувачів про правила організації праці та регулярного контролю умов роботи з ПК.

Інтенсивна робота за персональним комп'ютером може стати причиною виникнення ряду захворювань, пов'язаних із впливом несприятливих умов праці та недотриманням ергономічних вимог. Основні фактори, що спричиняють такі відхилення у здоров'ї користувачів: низька якість зображення, неправильна яскравість та контрастність можуть викликати перевтому очей та інші порушення зору, невідповідність висоти столу чи стільця, неправильне розташування монітора та інших периферійних пристроїв може призводити до м'язово-скелетних порушень, відсутність достатнього освітлення, неякісний мікроклімат (наприклад, недостатня вентиляція) сприяють загальному погіршенню самопочуття.

Серед захворювань, які виникають внаслідок тривалої роботи за ПК, найпоширенішими є:

Порушення зору. Втома очей, сухість, почервоніння, погіршення гостроти зору.

Кістково-м'язові порушення. біль у спині, шиї, плечах, синдром зап'ястного каналу.

Захворювання шкіри. Реакції на пил, що накопичується на поверхні обладнання, або алергічні реакції на матеріали робочого місця.

Порушення пов'язані зі стресом та нервово-емоційним навантаженням: підвищена тривожність, безсоння, головний біль, дратівливість.

Для зменшення негативного впливу ПК на здоров'я важливо дотримуватись правил ергономіки, забезпечувати регулярні перерви в роботі, правильно облаштовувати робоче місце та контролювати санітарно-гігієнічні умови.

Особливості праці користувача ПК

Стан організму користувачів персональних комп'ютерів значною мірою залежить від типу роботи та умов її виконання. Встановлено, що це

визначається як суб’єктивними показниками (скаргами на самопочуття), так і об’єктивними (функціональними показниками організму).

Користувачів ПК можна умовно розділити на такі групи

Постійні користувачі — ті, хто працює за комп’ютером регулярно через професійні обов’язки (програмісти, дизайнери, офісні працівники тощо).

Періодичні користувачі — наприклад, учні та студенти, які використовують ПК для навчання.

Користувачі, які працюють час від часу — наприклад, люди, які використовують ПК для розваг, пошуку інформації чи спілкування.

Робота за ПК супроводжується такими негативними факторами як тривале сидіння із зафіксованим положенням тіла та обмеженням загальної м’язової активності. Велике навантаження на дрібні м’язи кистей, що може призводити до м’язово-скелетних порушень, наприклад, синдрому зап’ястного каналу. Робота з монітором створює значне навантаження на очі, викликаючи втому, подразнення, сухість і погіршення зору. Стрес від складних завдань чи роботи в обмежені терміни впливає на психоемоційний стан користувачів.

Ці фактори особливо важливі для постійних користувачів ПК, оскільки їхній вплив накопичується, що може призводити до хронічних захворювань. Регулярне дотримання ергономічних, санітарно-гігієнічних вимог та фізичної активності допомагає зменшити ризики, пов’язані з роботою за комп’ютером.

Порушення зору є одними з основних професійних захворювань, пов'язаних з роботою за комп'ютером, і часто виникають через нераціональне освітлення, специфіку світлотехнічного оформлення робочих місць з ПК, а також недотримання режиму праці.

Світлотехнічна специфіка робочих місць з ПК визначається різноманітністю об'єктів зорової роботи, які мають різний контраст, розташовані на різних відстанях і в різних площинах зору. Ці особливості

вимагають постійного переміщення лінії зору від одного об'єкта до іншого. Наприклад:

Робоча документація часто розміщується на столі в горизонтальній площині на відстані 350 мм від очей користувача, має негативний контраст (темні об'єкти на світлому фоні).

Об'єкти на клавіатурі більші за розміром і знаходяться під певним нахилом, що також змінює напрямок зору.

Яскраві елементи на моніторі розташовані на відстані 450-600 мм і мають позитивний контраст на темному фоні, що вимагає незвичної орієнтації лінії зору в горизонтальній площині.

Це призводить до переадаптації зору, коли користувач постійно переключається між об'єктами з різним контрастом і відстанями. За восьмигодинний робочий день користувач може здійснити до 30,000 поглядів на екран. Око працює з перевантаженням, що ускладнює його адаптацію та спричиняє астенопічні явища — відчуття різі, болю, ломоти в очах і голові, а також розпливчастість зображення.

Постійний погляд на матове скло монітора знижує частоту кліпання, що призводить до сухості очей і погіршення зору (синдром Сікка). Монітори з пульсуючим самосвітним екраном (який не відповідає нормативним вимогам щодо пульсації) можуть викликати втому та дискомфорт, зокрема зорову.

Якщо екран має дзеркальну поверхню або неплоский зовнішній вигляд, це може призвести до появи відблисків, що створює додаткове навантаження на зір, викликаючи астенопічні явища.

Несправедливий розподіл яскравості в полі зору (наприклад, коли фон навколо екрану є набагато яскравішим за сам екран) порушує основні функції зору, що може спричиняти дискомфорт, втому та зниження зорової здатності.

Таким чином, організація робочого місця з ПК повинна враховувати освітлення, налаштування екрану та відстань до нього, щоб зменшити ризики розвитку зорових порушень.

Засліплююча дія світильників у приміщенні, на робочому місці з ПК більша, ніж на інших, тому що лінія зору користувача при роботі з екраном майже горизонтальна, що призводить до зменшення захисного кута дії різних засліплюючих джерел світильники, вікна тощо). Це викликає не тільки астенопічні явища, але й функціональні порушення очей користувача.

Кольоровий шрифт збільшує навантаження на зір, оскільки складові кольорів мають різні довжини хвиль і видимі на різній віддалі. Око потребує точнішої адаптації, ніж при чорно-білому зображенні.

Робота за персональним комп'ютером, незважаючи на всі її переваги, має серйозні наслідки для здоров'я користувачів, особливо якщо робоче місце не належним чином організовано з ергономічної точки зору.

Тривала робота за комп'ютером спричиняє статичне напруження м'язів спини, шиї, рук і ніг, що призводить до втоми та специфічних скарг. Ушкодження хребта часто виникають через неправильне налаштування робочого місця, зокрема, якщо крісло неправильно підтримує згин хребта. Погана постава під час роботи може спричиняти напруження в плечах і шиї, що, у свою чергу, викликає біль в ділянці шиї, спини і голови. Враховуючи, що працівник може сидіти за комп'ютером до 80,000 годин протягом свого життя, це створює високий ризик розвитку міжхребцевих захворювань.

Не менш важливим є неправильне положення рук під час введення даних за допомогою клавіатури. Якщо зап'ястя підняті під час набору тексту, це може призвести до перетискання нервів у вузьких місцях зап'ястя, що викликає тунельний синдром (синдром Карпального каналу). Також розвивається синдром RSI (Repetitive Strain Injury) — хронічне розтягнення зв'язок через постійне напруження м'язів кистей рук. Це захворювання може непомітно розвиватися протягом кількох років, що часто ускладнює його діагностику та лікування.

Робота за ПК є надзвичайно монотонною, адже часто люди виконують понад 600 однакових дій протягом години, що складає близько 75% робочого часу. Така монотонність, у поєднанні з неергономічними умовами праці і електромагнітними випромінюваннями, може призвести до загальних невротичних порушень.

Ці порушення можуть бути спричинені електромагнітними хвилями, які випромінює монітор. Вивченість впливу електромагнітного випромінювання на організм людини все ще залишається неповною, але вже відомо, що вплив хвиль низьких і наднизьких частот може викликати серйозні біологічні зміни, зокрема утворення пухлин.

Захворювання шкіри

Робота за персональним комп'ютером може мати серйозні наслідки для здоров'я через вплив різних шкідливих факторів. Окрім вже згаданих ергономічних і психоемоційних аспектів, важливо звернути увагу на інші небезпеки, пов'язані з моніторами та периферійними пристроями, які можуть спричиняти шкідливий вплив на шкіру та органи дихання користувача.

Один з шкідливих факторів, який не завжди враховують, — це електростатичне поле монітора, яке притягує частинки пилу з повітря. Це може стати проблемою для людей з чутливою шкірою. Експозиція до пилу, який осідає на екрані, може викликати подразнення шкіри, висипки та навіть запалення, особливо при тривалій роботі за монітором.

Також необхідно звернути увагу на хімічні токсини, що можуть виділятися від матеріалів, з яких виготовляються корпуси ПК і моніторів. Діоксини та фуран — це токсичні сполуки, які можуть бути присутніми в атмосфери через виділення з корпусів комп'ютерної техніки. Ці речовини мають канцерогенний ефект і є дуже небезпечними навіть у незначних кількостях. Вони можуть потрапляти в повітря не тільки при високих температурах (наприклад, під час горіння), але й при звичайних умовах.

Також важливо зазначити, що ці токсини належать до групи протипожежних засобів, що використовуються в матеріалах корпусу.

Ще одним небезпечним чинником є озон, який утворюється при роботі з лазерними принтерами. Це відбувається через вплив електричних зарядів на кисень в повітрі. Озон є сильним подразником слизових оболонок носа, очей та горла і може спричиняти проблеми з диханням, запалення або навіть рак, якщо виявляється в значних кількостях. Хоча нові принтери оснащуються фільтрами для озону, вони з часом втрачають свою ефективність, що може спричиняти накопичення озону в приміщенні.

Вибір монітора залежить від багатьох технічних характеристик (табл. 4.1), які мають великий вплив на комфорт роботи та здоров'я користувача. Якщо на ці характеристики не звертати увагу, можна зіткнутися з проблемами з зором, болями в спині чи шиї, а також іншими фізіологічними порушеннями. Правильний монітор з оптимальними параметрами сприяє зручності роботи та зменшує ризик виникнення професійних захворювань.

Розміри екрану монітора (дюйм)	Рекомендована роздільча здатність
14	640x480
15	800x600
17	1024x768
19	1280x1024
21	1600x1200

Табл. 4.1 Розмір єкрану монітора та рекомендована роздільна здатність

Монітори є критично важливими пристроями для роботи з комп'ютером, і неправильний вибір або невірне встановлення можуть негативно вплинути на зір і здоров'я в цілому. Ось деякі технічні характеристики моніторів, на які важливо звертати увагу при виборі, щоб мінімізувати потенційні ризики:

Розмір екрана визначається по діагоналі монітора. Стандартні розміри моніторів варіюються від 14 до 21 дюймів. Реальний розмір зображення зазвичай дещо менший через технологічні особливості виготовлення ЕПТ.

Роздільна здатність визначає чіткість зображення. Чим вища роздільна здатність, тим більше точок (пікселів) відображається на екрані, що забезпечує високу деталізацію зображення.

Зернистість зображення залежить від роздільної здатності та розміру екрану. Чим більша піксельна щільність, тим чіткіше зображення. Для комфортної роботи важливо, щоб зернистість була достатньо високою, щоб уникнути пікселізації зображення, що може призвести до навантаження на зір.

Вертикальна розгортка визначає, скільки разів за секунду зображення оновлюється по вертикалі. Зазвичай вимірюється в герцах (Гц). Вищі значення забезпечують більш плавне зображення і зменшують навантаження на очі.

Горизонтальна розгортка вказує на кількість пікселів, які можуть бути відображені по горизонталі за одну секунду. Вища горизонтальна розгортка дозволяє відображати більше інформації на екрані без втрат у якості.

Смуга пропускання відеосигналу - це здатність монітора обробляти та відображати відеосигнал з високою швидкістю. Важлива для роботи з графічними додатками, іграми чи відео. Низька смуга пропускання може спричинити спотворення зображення або затримку.

Важливі такі регулювання, як нахил екрана, висота та поворот, оскільки неправильне розташування екрана може викликати дискомфорт і напругу в шийному відділі та спині.

Монітори з мікропроцесорним управлінням можуть автоматично регулювати яскравість, контрастність та інші параметри для оптимального відображення в залежності від умов освітлення та зміни зображення.

Динамічне фокусування - це технологія, яка дозволяє автоматично коригувати фокус для чіткішого зображення на різних частинах екрану. Важлива для зменшення спотворення зображення і поліпшення якості картинки.

Однак поганий монітор (наприклад, з низьким розширенням, частотою оновлення або з проблемами з кольоровою передачею) може призвести до значних проблем з зором та викликати астенопічні симптоми, такі як головний біль, біль в очах, зниження чіткості зображення. Окрім цього, робота з низькоякісними моніторами може спричинити швидке висихання очей та інші проблеми з органами зору через неправильне освітлення і часті зміни фокусу.

Профілактика:

Правильне розташування монітора - монітор має бути розміщений на оптимальній відстані від очей, а також у правильному куті нахилу, щоб уникнути напруги в очах і шиї.

Вибір екологічно чистих пристроїв - використання техніки з сертифікатами безпеки і низьким рівнем викидів шкідливих речовин (наприклад, діоксинів і озону).

Фільтри та очищення повітря - для зниження рівня озону і пилу в приміщенні необхідно використовувати очисники повітря або регулярно провітрювати кімнату.

Перерви в роботі - регулярні перерви для розслаблення очей і тіла є важливими для запобігання розвитку астенопії та хронічних захворювань.

Таким чином, для мінімізації негативного впливу техніки на здоров'я користувачів важливо враховувати як ергономічні, так і хімічні та фізичні аспекти роботи з комп'ютером.

Розмір діагоналі екрану монітора слід вибирати виходячи з призначення. Найбільш поширені моделі з діагоналями від 15 до 21 дюйма.

4.2 Охорона довкілля

Робота за комп'ютером має вплив не лише на здоров'я користувачів, але й на довкілля. Охорона навколишнього середовища в контексті роботи з ПК включає різні аспекти, які стосуються енергоспоживання, управління відходами, вибору техніки та її утилізації. Розглянемо основні заходи з охорони довкілля, пов'язані з використанням комп'ютерної техніки.

Споживання електроенергії комп'ютерами та периферійними пристроями істотно впливає на стан довкілля через підвищені викиди вуглекислого газу, що утворюються при генерації електроенергії. Щоб зменшити цей негативний вплив, необхідно дотримуватись низки рекомендацій: використовувати енергоефективну техніку, яка відповідає сучасним стандартам, наприклад, сертифіковану за екологічними програмами на кшталт Energy Star чи аналогічними міжнародними стандартами; активувати функції енергозбереження: налаштовувати сплячий режим або автоматичне відключення монітора та інших компонентів після періоду бездіяльності; вимикати пристрої, що не використовуються, наприклад, принтери, сканери або зовнішні жорсткі диски, а також уникати роботи техніки в режимі очікування, якщо її використання не планується найближчим часом; надавати перевагу ноутбукам замість стаціонарних ПК у випадках, коли це можливо, адже ноутбуки споживають менше електроенергії; застосовувати розумні розетки та таймери для автоматичного вимикання пристроїв у неробочий час; встановлювати LED-екрани замість застарілих технологій (наприклад, LCD з CCFL-підсвіткою), адже вони споживають менше енергії та мають більший строк служби; оптимізувати налаштування ПК, наприклад, знижуючи яскравість екрана до комфортного рівня, що додатково зменшить споживання енергії; навчати співробітників та

користувачів правилам енергоефективного використання техніки, щоб знизити зайве навантаження на електромережу.

Ці заходи дозволяють не лише скоротити обсяги викидів парникових газів, але й зменшити витрати на електроенергію, сприяючи економічній ефективності та екологічній відповідальності.Комп’ютери та електронні пристрої містять токсичні речовини (свинець, ртуть, кадмій), які можуть потрапити у довкілля, якщо їх неправильно утилізувати.

Висновоки за розділом 4

В ході роботи було проаналізовано основні негативні чинники для здорв’я людини прри роботі за ПК. Також був проаналізований екологічний аспект робти за ПК, зокрема можливі можливі фактори забруднення навколишнього середовища.

ВИСНОВКИ

В ході виконання дипломної роботи було проведено дослідження основних етапів створення моделі машинного навчання для прогнозування стану водного середовища. Також був проведений аналіз основних методів машинного навчання та алгоритмів.

Завданням кваліфікаційної роботи була розробка інформаційно-вимірювальної системи оперативного екологічного моніторингу водного середовища з прогнозом стану якості води на основі машинного навчання. Був проведений аналіз існуючих рішень застосування машинного навчання у сфері екологічного моніторингу.

Модель машинного навчання розроблена на Python.

В роботі також розглянуто питання охорони праці та навколишнього довкілля.

ПЕРЕЛІК ПОСИЛАНЬ

1. В. О. Харченко Оснви машинного навчання: навчальний посібник, 2023

2.Andriy Burkov The Hundred-Page Machine Learning Book, 2019 с. 30-60.

3.10 найкращих інструментів ШІ для моніторингу довкілля https://www.unite.ai/uk/найкращі-інструменти-штучного-інтелекту-для-моніторингу-навколишнього-середовища/

4. А.Ю. Дорошенкоа, В.М. Шпиг, Р.В. Кушніренко Застосування машинного навчання для уточнення чисельних метеорологічних прогнозів, 2020.

5. Кононова К. Ю. Машинне навчання: методи та моделі підручник, 2020

6. Introduction to Machine Learning Nils J. Nilsson, 2005 с. 30-41.

7. The Elements of Statistical Learning. Data Mining, Inference and Prediction» Trevor Hastie, Robert Tibshirani, Jerome Friedman, 2014.

8. Основи моделювання ринкових ситуацій https://web.posibnyky.vntu.edu.ua/fksa/12kocubynsky,kyslycia_osn_model_rynk_sytuac/zmist.html

9. Bengio, Y. and LeCun, Y. (2007). Scaling learning algorithms towards AI. In Large Scale Kernel Machines.

10. Brett Lantz. Machine Learning with R. Packt Publishing, Birmingham – Mumbai, 2013.

11. David J. C. MacKay (2003). Information Theory, Inference and Learning Algorithms. Cambridge University Press.

12. Документація мови Python https://docs.python.org/3/

13. Документація бібліотеки Scikit-Learn https://scikit-learn.org/stable/supervised_learning.html#

14. Документація TensorFlow https://www.tensorflow.org/api_docs/python/tf/all_symbols

15 . Donald M. Machine learning, neural and statistical classification / M. Donald, D. J. Spiegelhalter, C. C. Taylor. – New York : Ellis Horwood, 1994.

16. Deisenroth M. P. Mathematics for machine learning / M. P. Deisenroth, A. A. Faisal, C. S. Ong. – New York : Cambridge University Press, 2020.

17 . Schapire R. E. Boosting: Foundations and Algorithms / R. E. Schapire, Y. Freund. – Cambridge : MIT Press, 2012.

18. Ян Гудфеллоу Иошуа Бенджио Аарон Курвилль Глубокое обучение, 2018

19. Breiman L. Random forests. Machine Learning. Vol. 45, N 1. P. 5–32, 2001

20. Shalev-Shwartz S. Understanding Machine Learning: From Theory to Algorithms / S. Ben-David, S. Shalev-Shwartz. – New York : Cambridge University Press, 2014. – 449 p.

21. Hastie, T., Tibshirani, R., and Friedman, J. (2001). The elements of statistical learning: data mining, inference and prediction. Springer Series in Statistics. Springer Verlag.

22. Классификация и кластер / под ред. Дж. Вэн-Райзина. М.: Мир, 1980. 390 с.

23. Флах П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных. М.: **ДМК** Пресс, 2015. 400 с.

24. Саттон Ричард С., Барто Эндрю Г. Обучение с подкреплением, Reinforcement Learning. — М.: БИНОМ. Лаборатория знаний, 2017.

25. Дэви Силен, Арно Мейсман Основы Data Science и Big Data. Python и наука о данных, 2019

ДОДАТОК А

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
from sklearn.model_selection import train_test_split
# load the data that is needed into the application
df = pd.read_csv('water_potability.csv')
df.head()
sns.heatmap(df.corr(), cbar=True)
df.describe()
df.info()
df.isnull().sum()

missing_data = df.isnull().sum()
total = df.isnull().count()
percent = (missing_data/total) * 100
missing_data = pd.concat([missing_data, percent], axis=1, keys=['Total', 'Percent'])
missing_data
df['ph'] = df['ph'].fillna(df['ph'].mean())
df['Sulfate'] = df['Sulfate'].fillna(df['Sulfate'].mean())
df['Trihalomethanes'] = df['Trihalomethanes'].fillna(df['Trihalomethanes'].mean())

df.info()
X = df.drop('Potability', axis=1)
y = df['Potability']
# scale the data
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()

X = scaler.fit_transform(X)

# split the data into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
# build the model
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(128, kernel_regularizer=tf.keras.regularizers.l2(0.001),
activation='relu', input_dim=X_train.shape[1]),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(64, kernel_regularizer=tf.keras.regularizers.l2(0.001),
activation='relu'),
```

```
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

# Compile the model
optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)
model.compile(optimizer=optimizer, loss='binary_crossentropy',
metrics=['accuracy'])

# callbacks
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)

# Train the model
history = model.fit(X_train, y_train, epochs=200, batch_size=32,
validation_data=(X_test, y_test), callbacks=[early_stopping])

# Model Evaluation
evaluation = model.evaluate(X_test, y_test)
print(f"Test Loss: {evaluation[0]}, Test Accuracy: {evaluation[1]}")
# Summary of the process and results
import matplotlib.pyplot as plt

# Plot training & validation loss values
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['Train', 'Validation'], loc='upper left')
#  make some predictions
predictions = model.predict(X_test)

# convert the predictions to binary
predictions = np.round(predictions)

from sklearn.metrics import classification_report, confusion_matrix

print(classification_report(y_test, predictions))
print(confusion_matrix(y_test, predictions))

from sklearn.metrics import accuracy_score
```

```
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy}")

from sklearn.metrics import precision_score
precision = precision_score(y_test, predictions)
print(f"Precision: {precision}")
#  plot the graphs of the confusion matrix
cm = confusion_matrix(y_test, predictions)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predictions')
plt.ylabel('Actuals')
plt.title('Confusion Matrix')
plt.show()
```