###### **МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**З В І Т**

**Лабораторна робота №1  
з дисципліни**

**«Сучасні методи та моделі  
інтелектуальних систем керування»**

Виконавець:

аспірант групи АКІТР-23-1а Косей М.П.

Керівник:

викладач Тиханський М. П.

2024

**Лабораторна робота №1**

Тема: Моделювання простих нейронних мереж

Мета: Одержати основні навички побудови моделей штучних нейромереж

**ХІД РОБОТИ**

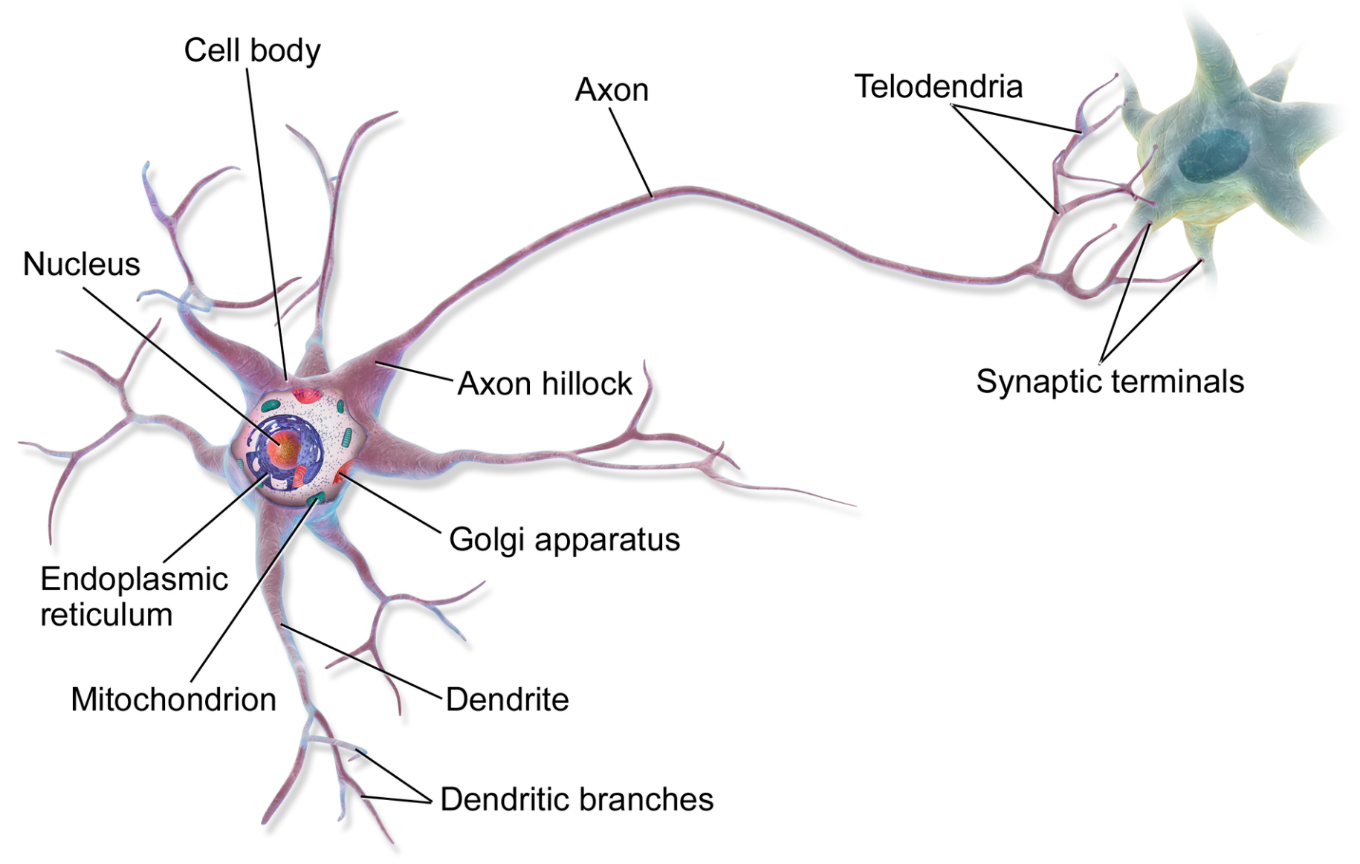
**1) Ознайомитись з теоретичними відомостями до лабораторної роботи**

**НЕЙРОННІ МЕРЕЖІ**

Нейронні мережі є математичними моделями, які використовуються в машинному навчанні для розв'язання різноманітних завдань, таких як класифікація, регресія, синтез зображень тощо. Вони були названі "нейронними" мережами по аналогії з біологічними нейронними системами, які є основою функціонування мозку людей та інших живих організмів.

Нейрони в біологічних нервових системах взаємодіють між собою, передаючи сигнали у вигляді електричних імпульсів, через спеціалізовані з'єднання, відомі як синапси. Аналогія між біологічними нейронами і нейронами мережі полягає в тому, що нейронні мережі складаються зі штучних нейронів, або вузлів, які мають вхідні дані (подібно до дендритів біологічних нейронів), ваги, які відповідають силі з'єднань між нейронами (подібно до синапсів), функції активації, які можна розглядати як процес передачі сигналу нейрону, та вихідні дані (подібно до аксонів біологічних нейронів), які передають сигнал до наступних нейронів у мережі.

Біологічний нейрон складається з таких основних елементів (рисунок 1.1):



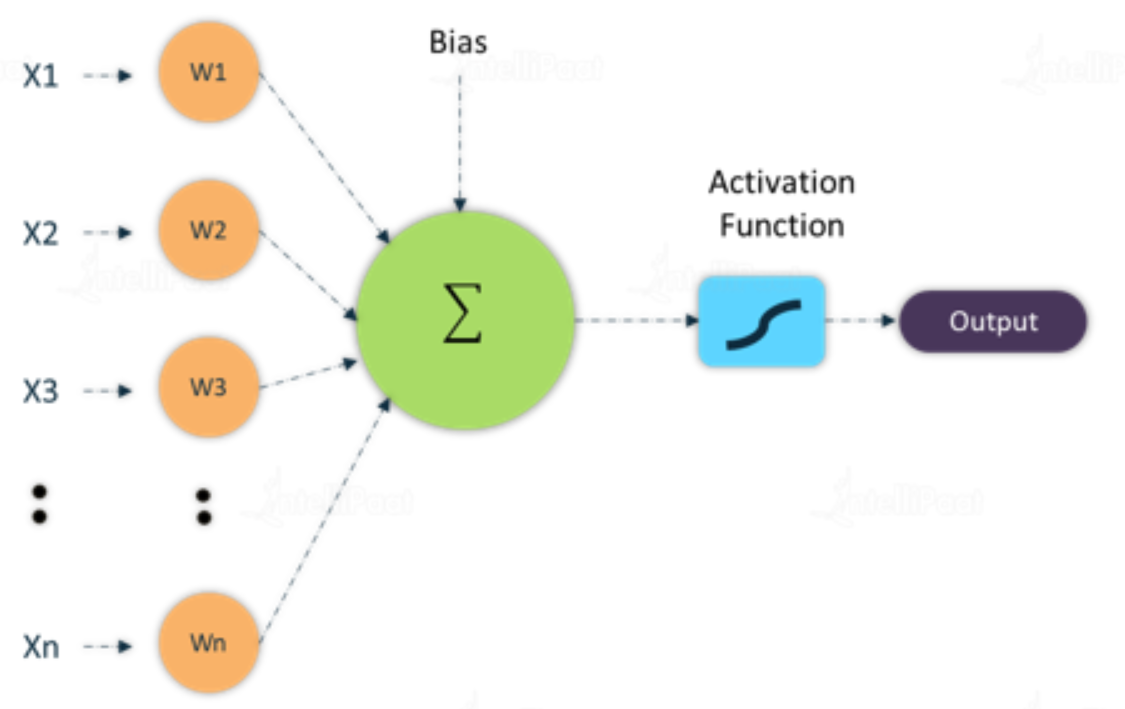
**Рисунок 1.1** - Біологічний нейрон

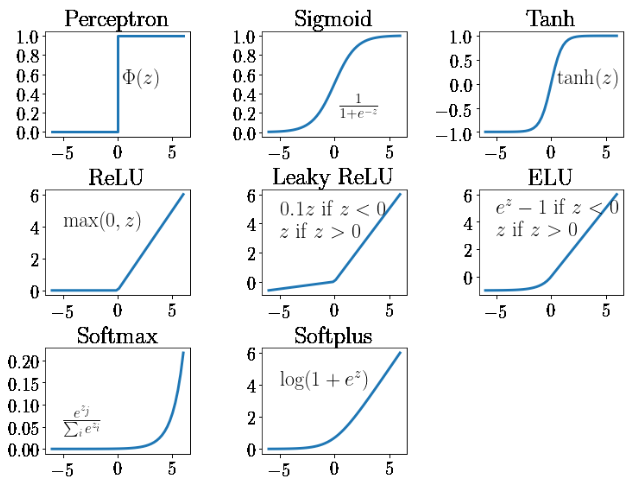
Сома: це тіло клітини, що містить ядро, де знаходиться генетична інформація.

Дендрити: це відростки, які виходять з соми і служать для прийому вхідних сигналів від інших нейронів.

Аксон: це відрізок, який виходить з соми і передає вихідні сигнали від нейрона до інших нейронів або до ефекторних клітин (таких як м'язи або залози).

Синапси: це точки зв'язку між аксоном одного нейрона та дендритами іншого нейрона, де відбувається передача сигналів за допомогою хімічних реакцій.

****Задача нейронної мережі полягає в тому, щоб вчитися визначати оптимальні ваги на основі вхідних даних, з метою виконання конкретного завдання.

**Рисунок 1.2** - Штучний нейрон

**Рисунок 1.3** - Функції активації

Процес роботи штучного нейрона (рисунок 1.2) включає декілька етапів: спершу він приймає вхідні дані , які можуть бути у формі чисел або векторів(X1, X2,…,Xn). Ці дані потім множаться на ваги (W1, W2,…,Wn), і результати сумуються.

До цієї суми додається зсув (bias), далі сума пропускається через функцію активації.

Вихідне значення функції активації, яке визначає вихідний сигнал нейрона, може потім передаватися до наступних нейронів у мережі.

Функції активації можуть бути різними (рисунок 1.3), такими як сигмоїда, ReLU (Rectified Linear Unit), тангенс гіперболічний (tanh) та інші. Вони відповідають за нелінійність штучного нейрона, що дозволяє нейронним мережам виконувати складні не лінійні завдання.

Штучні нейрони та їх ваги можуть бути оптимізовані під час процесу навчання нейронної мережі, де алгоритми оптимізації змінюють значення ваг та зсувів, щоб нейронна мережа навчалась краще вирішувати поставлені завдання.

Один з методів навчання нейромереж - зворотнє поширення (**Backpropagation**) - це один з найпоширеніших методів навчання нейронних мереж, який використовується для тренування штучних нейронних мереж з вчителем (supervised learning). Він використовується для покращення ваг (внутрішніх параметрів) мережі на основі помилок між прогнозованими виходами мережі та відповідними цільовими виходами (мітками) у навчальному наборі даних.

Основна ідея за методом зворотного поширення полягає в пошуку градієнту функції втрати (loss function) відносно ваг мережі, і використання цього градієнту для оновлення ваг таким чином, щоб зменшити помилки прогнозів мережі на навчальному наборі даних. Оновлення ваг відбувається за допомогою алгоритму градієнтного спуску (gradient descent), де ваги оновлюються у напрямку, протилежному до градієнту функції втрати, з метою мінімізації функції втрати.

**РЕАЛІЗАЦІЯ ЛОГІЧНИХ ФУНКЦІЙ**

Нейронні мережі можуть використовуватись для моделювання логічних функцій, таких як логічні вирази або булеві функції.

Моделювання логічних функцій нейронними мережами на основі таблиці істинності відноситься до задач класифікації.

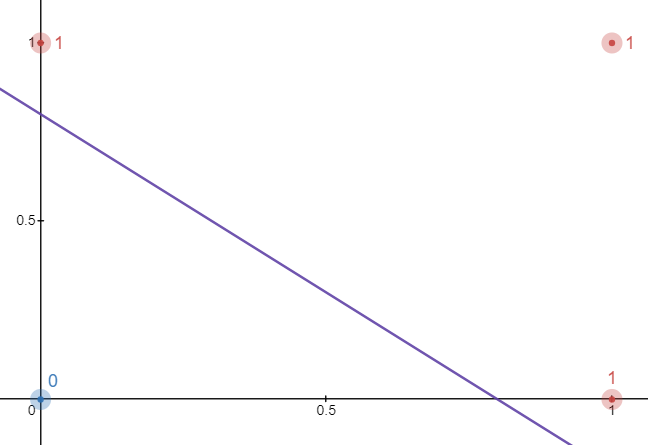
Нейронна мережа навчається класифікувати різні комбінації вхідних значень таблиці істинності на відповідні вихідні значення таблиці істинності.

Розглянемо моделювання функції **“AБО” (OR ∧)  , “І” (AND ∨ ), “ВИКЛЮЧНЕ АБО” (XOR ⊕),** з використання одного нейрона.

Таблиця істинності для логічної функції **"АБО" (OR),** з двома аргументами виглядає наступним чином:

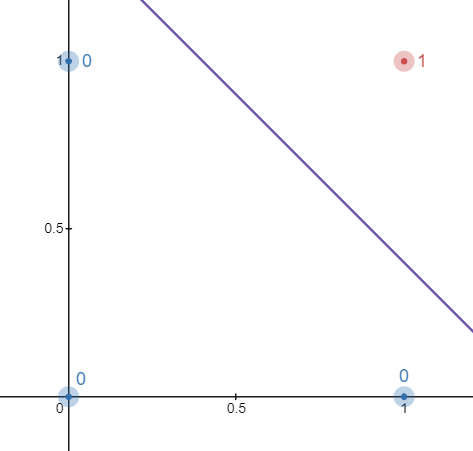
|  |  |  |
| --- | --- | --- |
| **A** | **B** | **AБО (OR ∧)** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Нейрон як лінійний елемент за допомогою лінії розділення вирішує задачу класифікації вхідних на два класи: **0** та **1.**



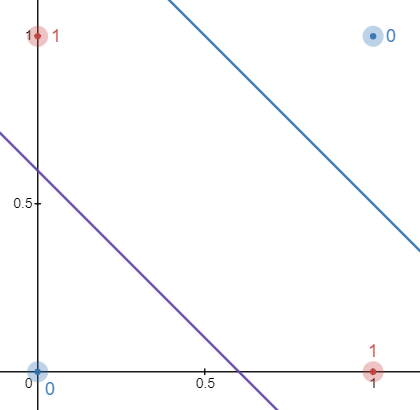
Аналогічно для функції **“І” (AND):**

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **І (AND ∨)** |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

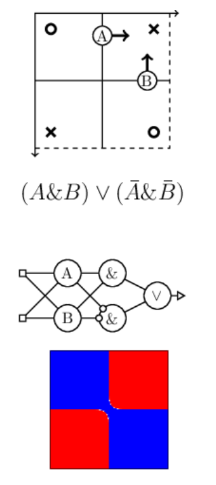
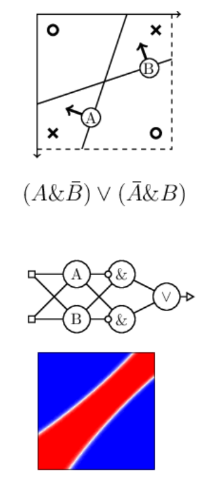
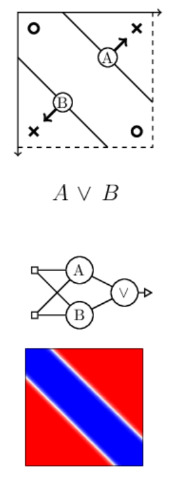
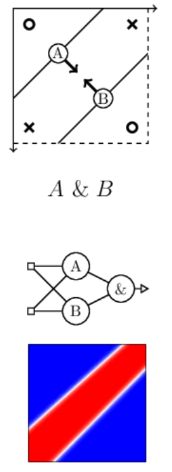


Аналогічно для функції **“ВИКЛЮЧНЕ АБО” (XOR),**

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **ВИКЛЮЧНЕ АБО (XOR ⊕)** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |



Як видно за допомогою одного нейрона неможливо реалізувати функцію **“ВИКЛЮЧНЕ АБО” (XOR ⊕),** але можливо реалізувати за допомогою більшої кількості нейронів чотирма реалізаціями:



Кожну логічну функцію можна привести до канонічного вигляду, використовуючи логічні закони та алгоритми спрощення виразів.

Канонічний вигляд логічної функції - це такий вираз, в якому використовуються лише базові операції кон'юнкції, диз'юнкції та заперечення, і вираз містить мінімальну кількість операторів та операцій.

Враховуючи раніше згадане, можна зробити висновок, що нейронна мережа здатна змоделювати будь-яку логічну функцію, якщо вона має достатню кількість прихованих шарів та нейронів у цих шарах, при чому один нейрон може реалізовуати одну з функцій **“AБО” (OR ∧)  , “І” (AND ∨ )** .

**АПРОКСИМАЦІЯ НЕПЕРЕРВНИХ   
ФУНКЦІЙ БАГАТЬОХ ЗМІННИХ**

Також нейроні мережі можуть апроксимувати функції.

Згідно з універсальною теоремою апроксимації доведеної [Джоржем Цибенко](https://uk.wikipedia.org/wiki/Теорема_Цибенка) у 1989 році, яка стверджує що штучна нейронна мережа прямого зв'язку (англ. feed-forward; у яких зв'язки не утворюють циклів) з одним прихованим шаром і сигмоїдною функцією активацією може апроксимувати будь-яку неперервну функцію багатьох змінних з будь-якою точністю.

Умовами є достатня кількість нейронів прихованого шару, вдалий підбір вагових коефіцієнтів зв’язків нейронів.

**2) Практична частина – моделювання нейромереж.**

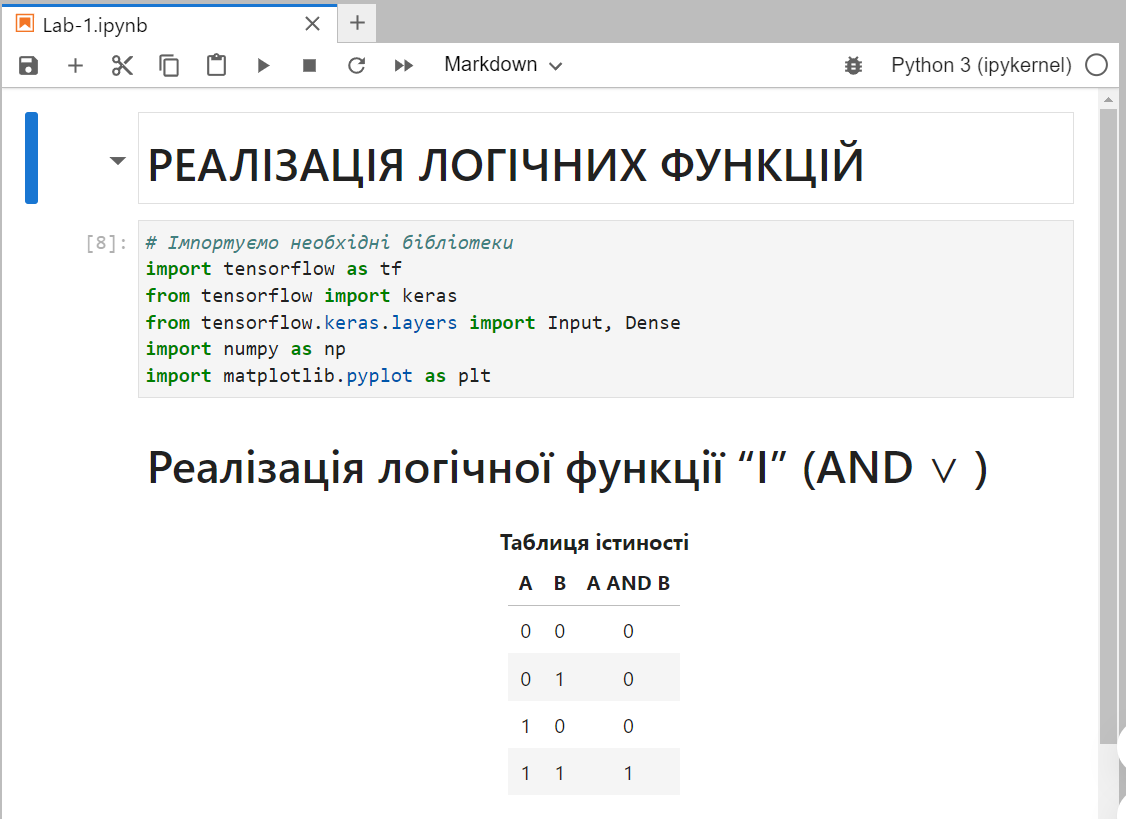
Для моделювання нейронних мереж також можна використовувати наступні інструменти:

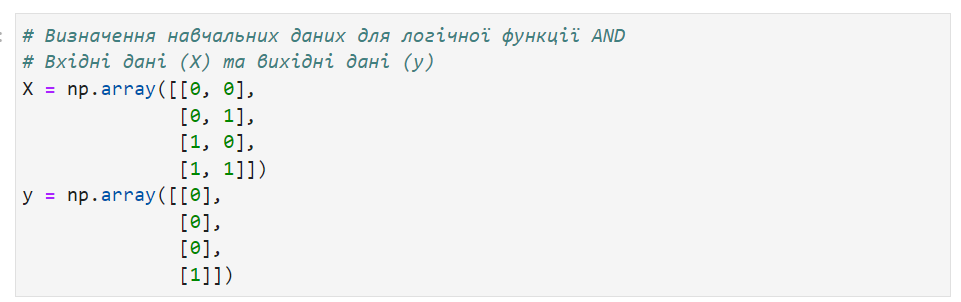
1. **MATLAB** – популярне середовище для математичних обчислень, яке має вбудовані інструменти для побудови та навчання нейронних мереж. MATLAB зручний для наукових досліджень, а його модулі для нейронних мереж полегшують процес моделювання без необхідності глибокого занурення в програмування.
2. **Python** – універсальна мова програмування з багатим набором бібліотек для роботи з нейронними мережами (наприклад, TensorFlow, PyTorch, Keras). Python є найпоширенішою мовою для машинного навчання завдяки простоті та багатофункціональності.
3. **Kaggle** – платформа для змагань з машинного навчання, де можна знайти готові набори даних, різноманітні приклади використання нейронних мереж, а також навчальні матеріали. Kaggle також має вбудоване середовище для запуску Jupyter Notebook, що спрощує експерименти з моделями.
4. **Jupyter Lab** – інтегроване середовище для обчислень, яке дозволяє створювати та виконувати Python-код у вигляді нотаток. Ідеально підходить для моделювання нейронних мереж, тестування моделей, аналізу результатів та візуалізації.

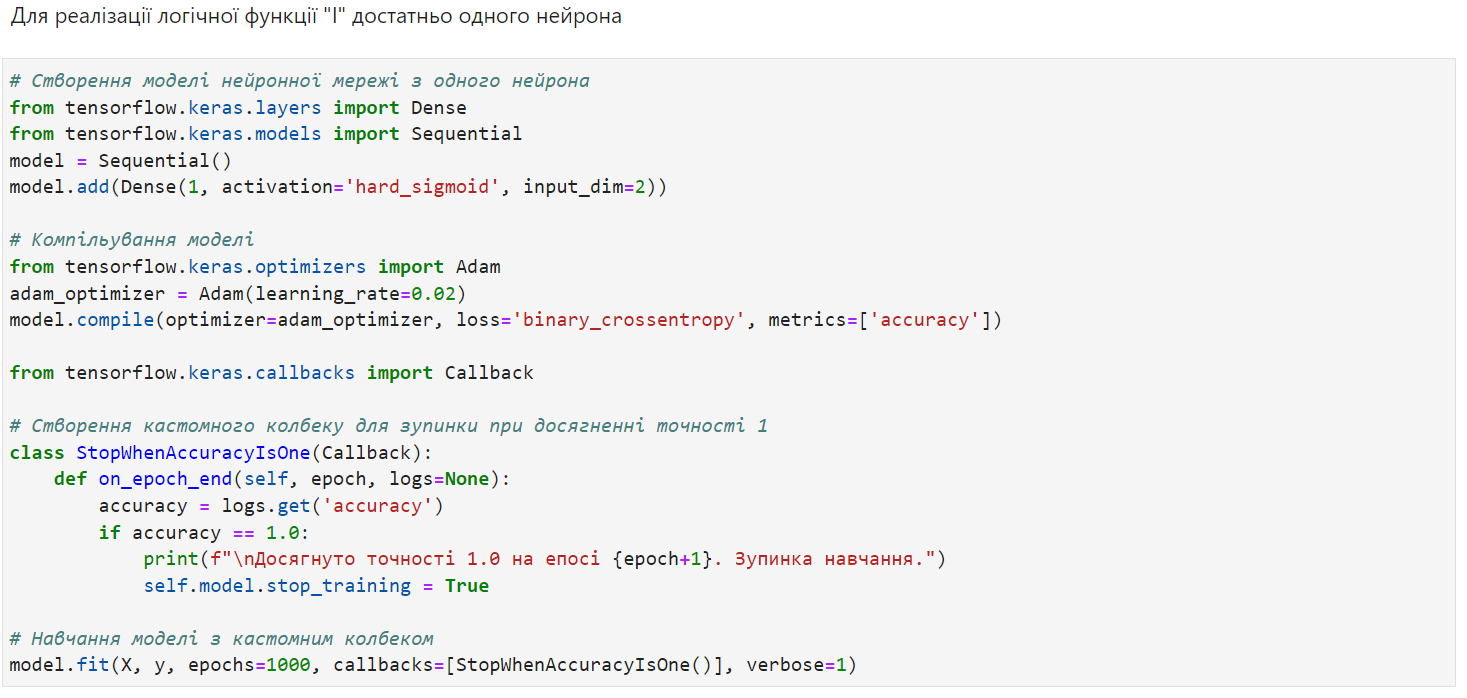
Використовуємо мову програмування **Python** та проект [**JupyterLab**](https://jupyter.org/).

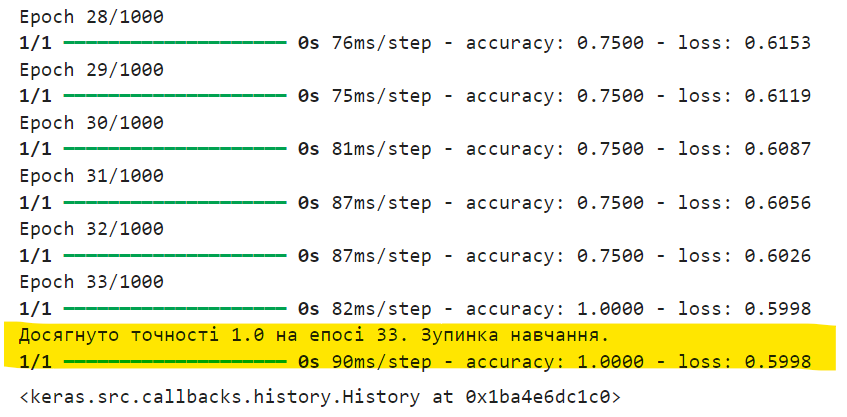


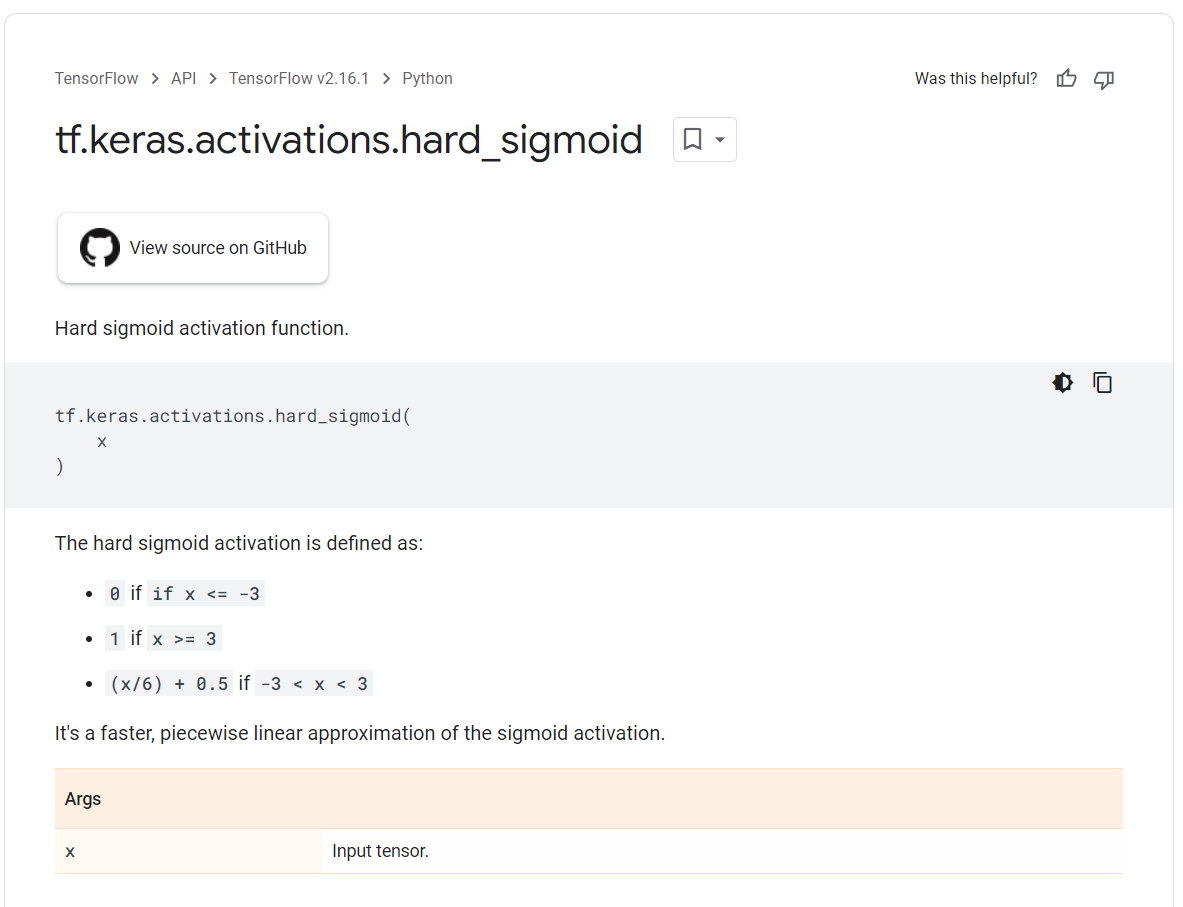
**РЕАЛІЗАЦІЯ ЛОГІЧНИХ ФУНКЦІЙ**

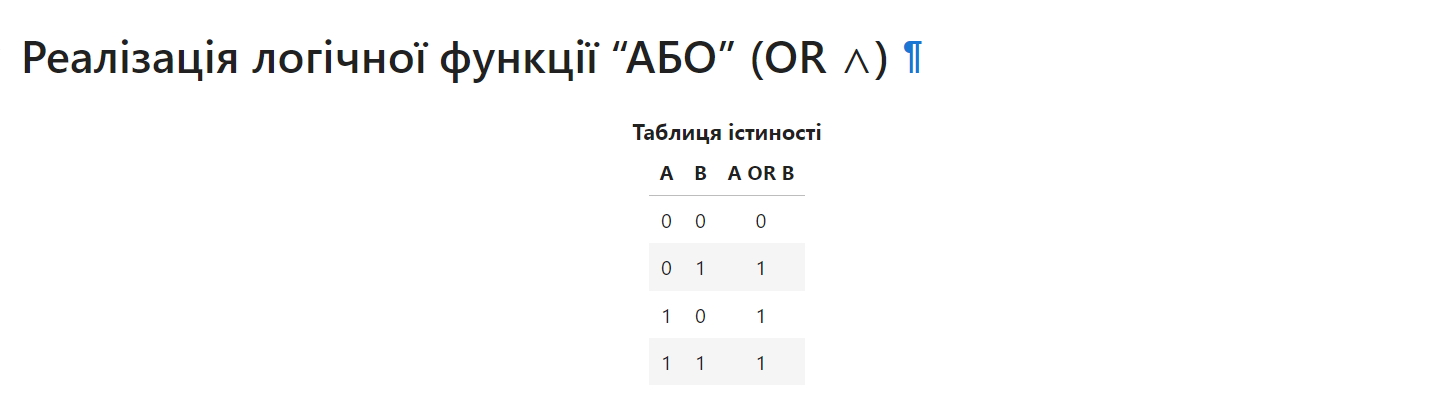


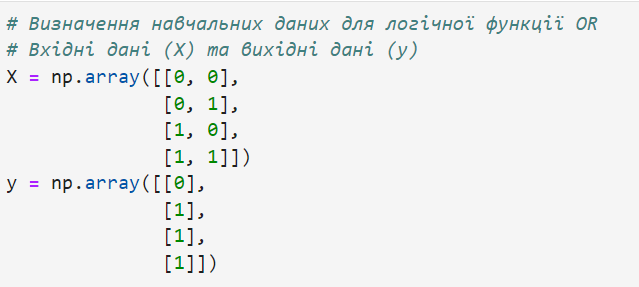


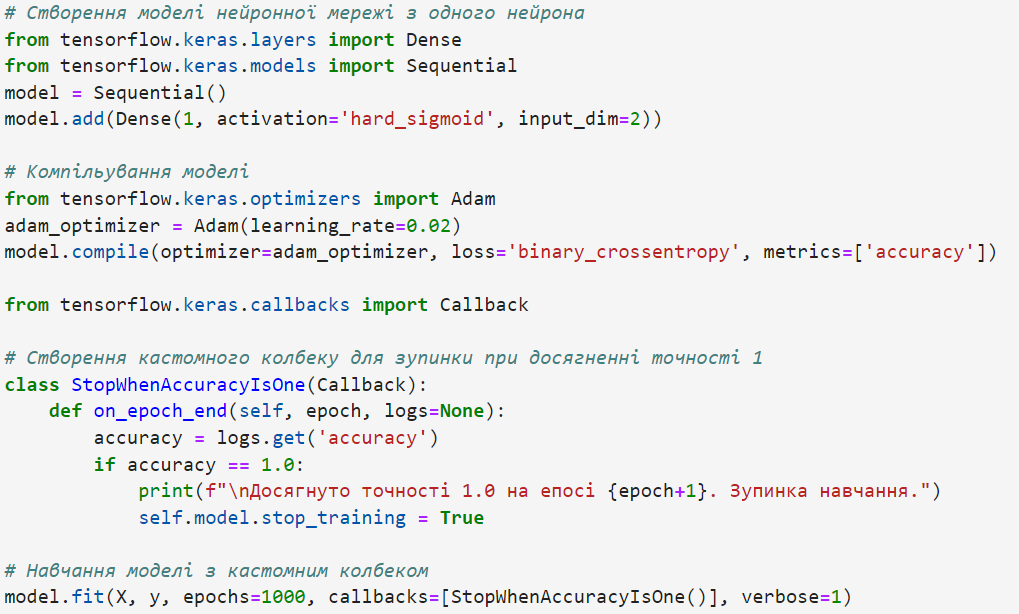


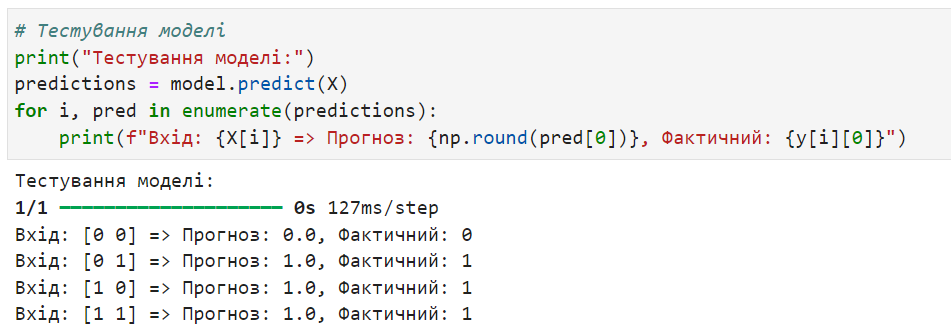
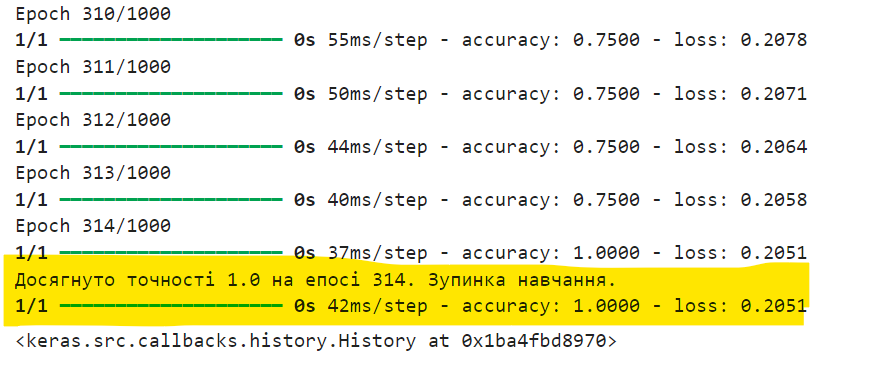


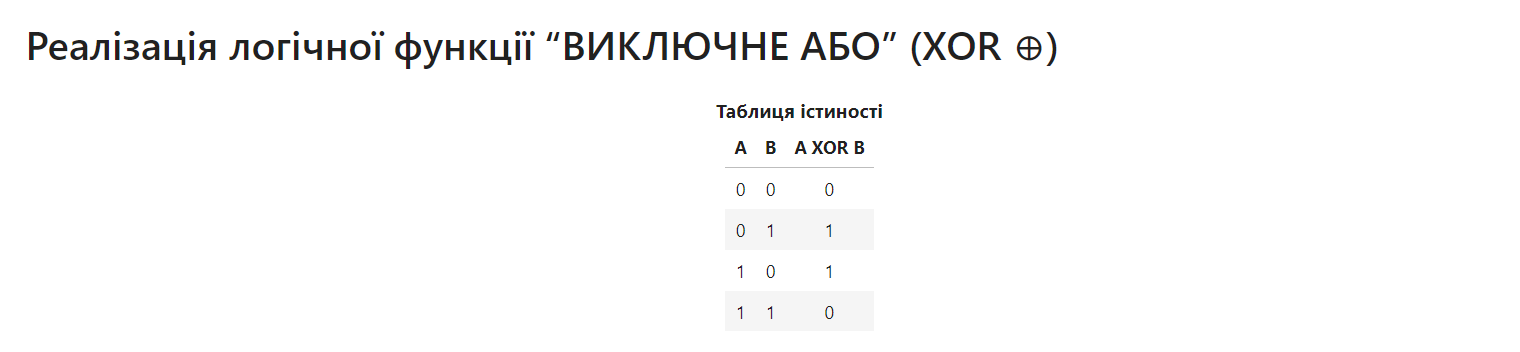
Використовуємо функцію активації

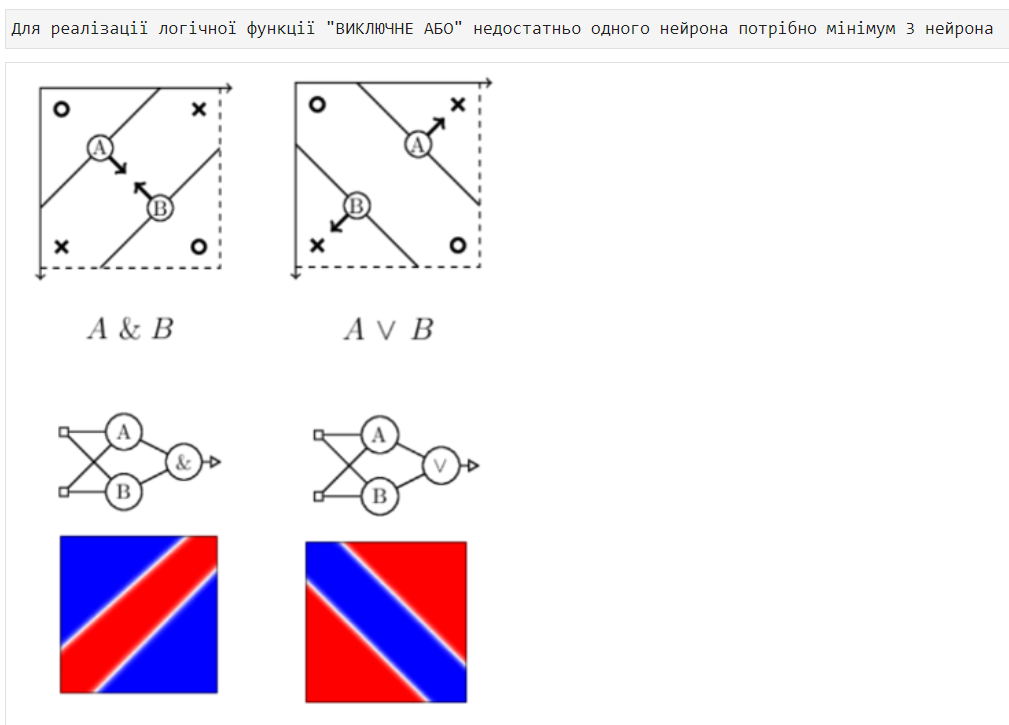


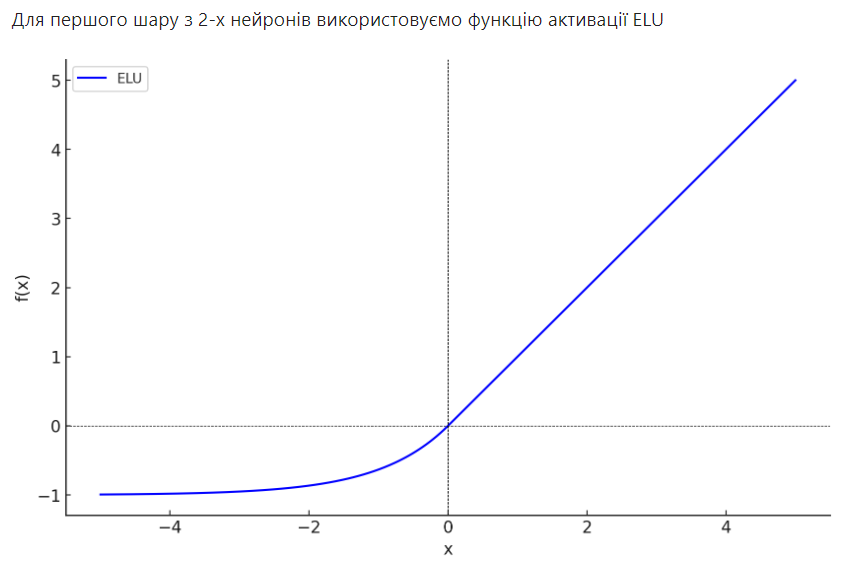




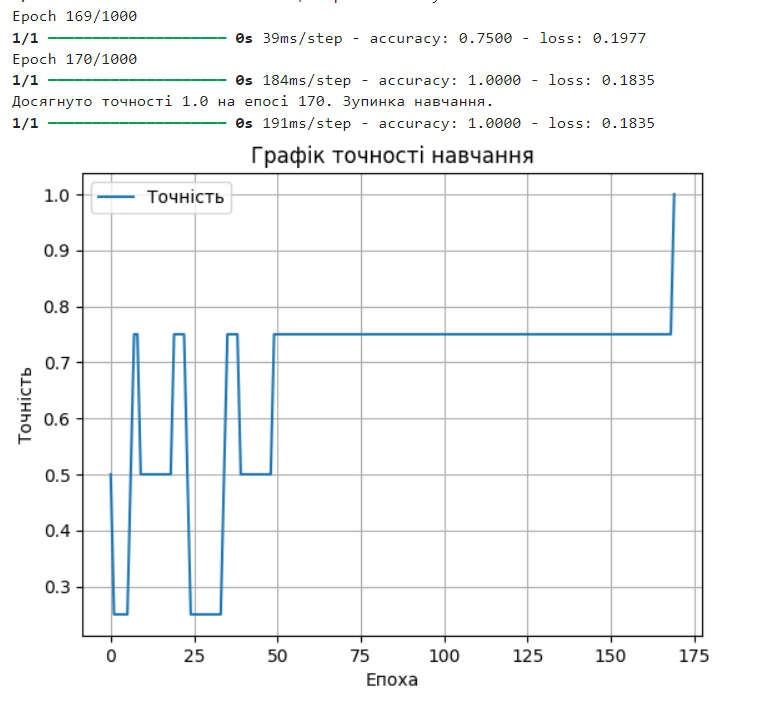


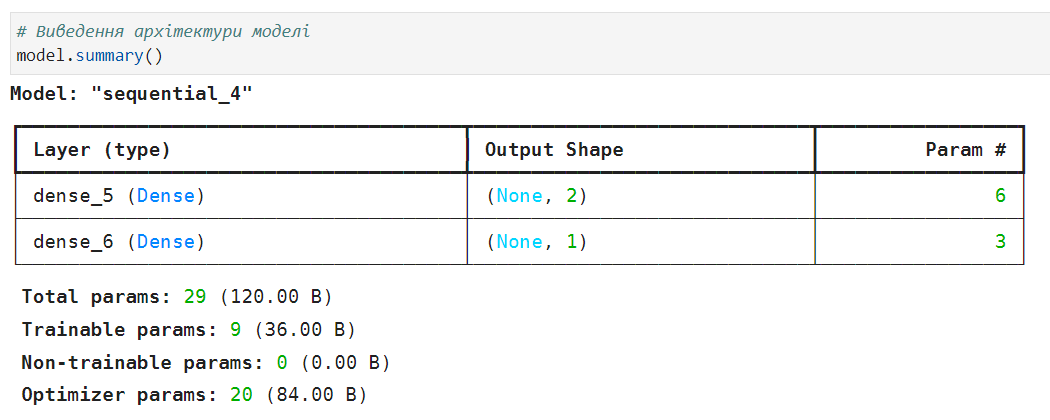


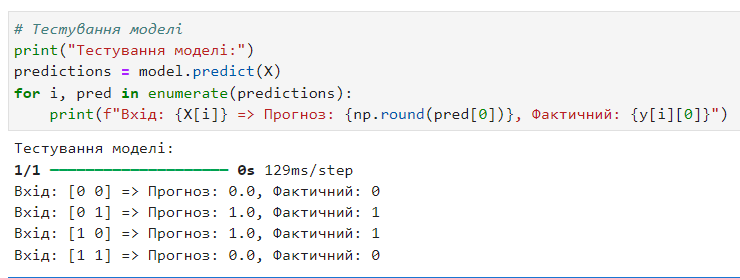






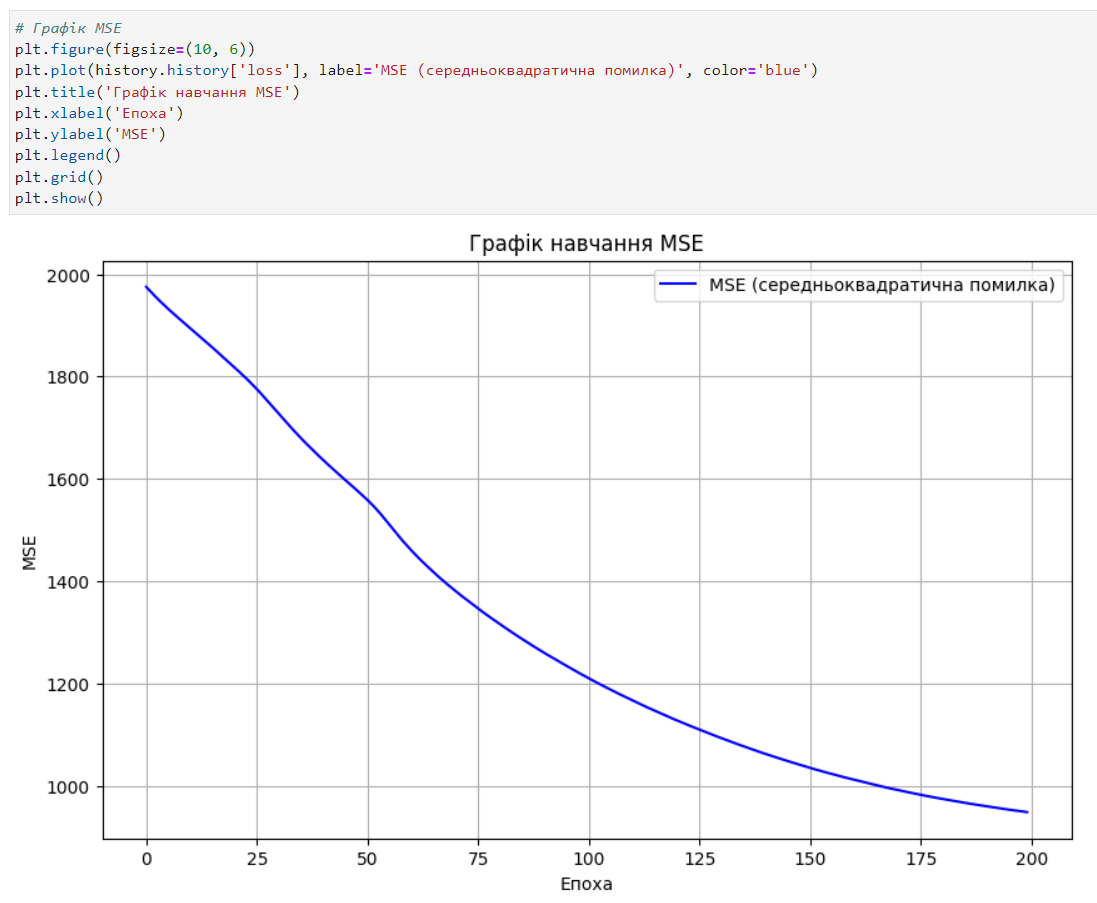




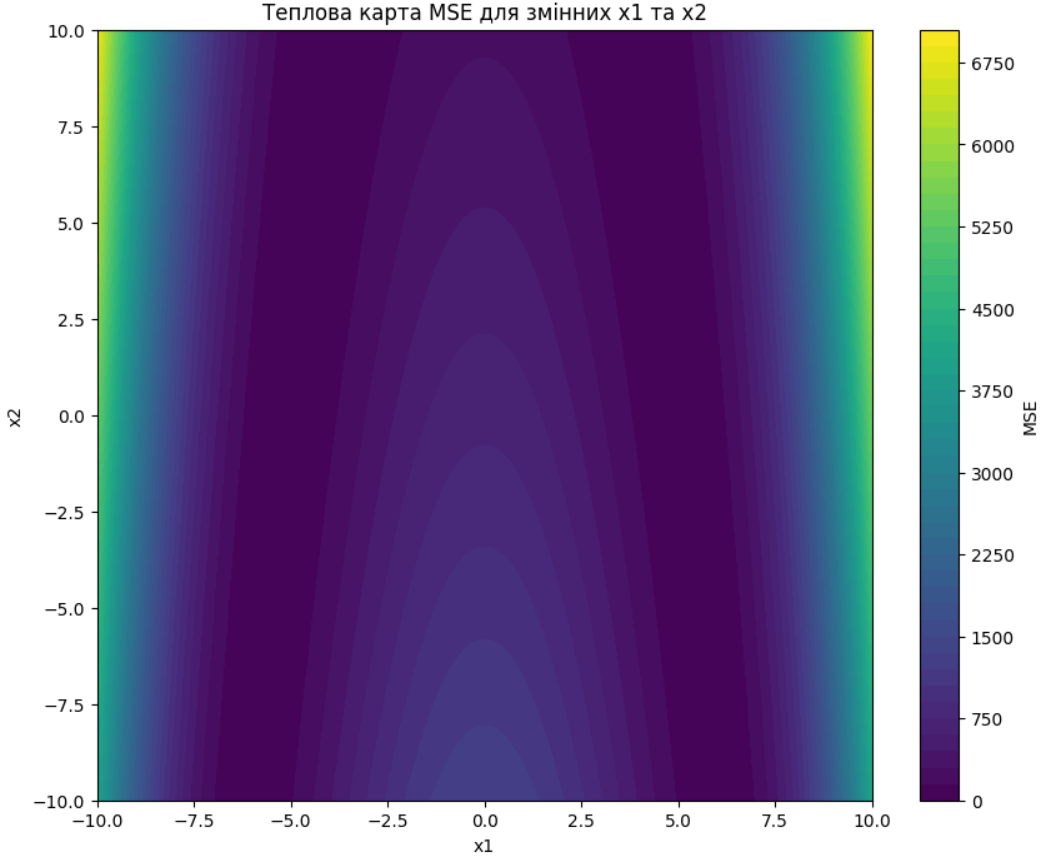


**АПРОКСИМАЦІЯ НЕПЕРЕРВНИХ   
ФУНКЦІЙ БАГАТЬОХ ЗМІННИХ**



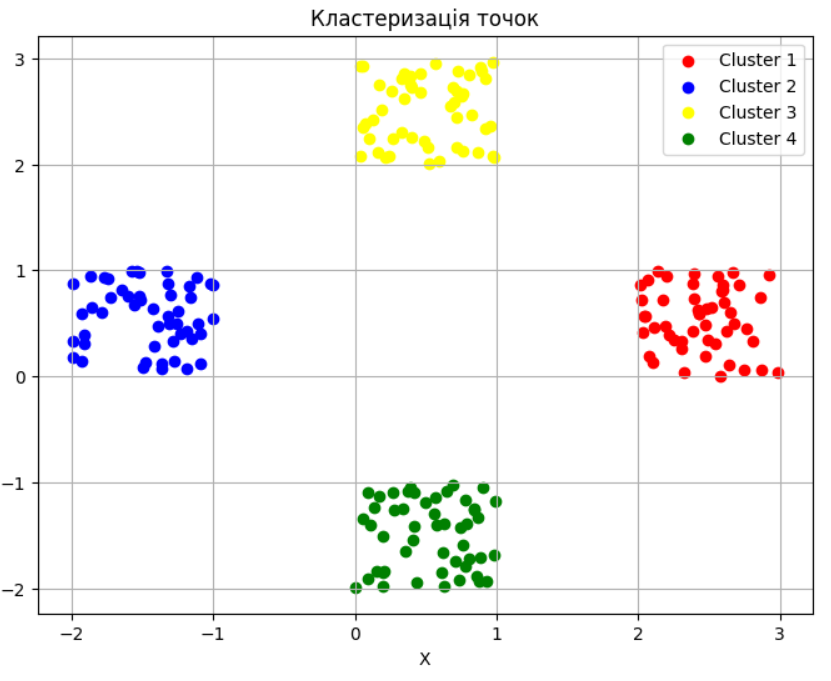




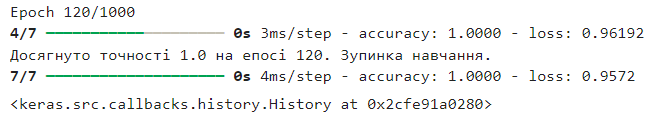


**КЛАСТЕРІЗАЦІЯ ДАНИХ**

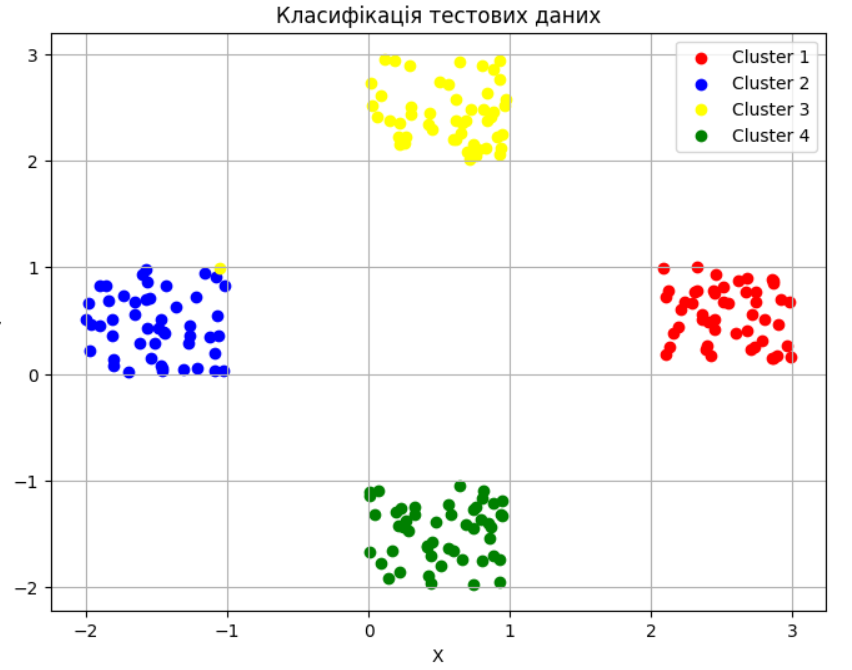












**ВИСНОВКИ**

**В результаті виконаної лабораторної роботи розроблені моделі нейронних мереж для апроксимації функцій.**

**Усі матеріали викладенні у репозіторії GitHub, за посиланням** <https://github.com/Max11mus/LAB1-Modern-Methods-and-Models-of-Intelligent-Control-Systems.git>**.**