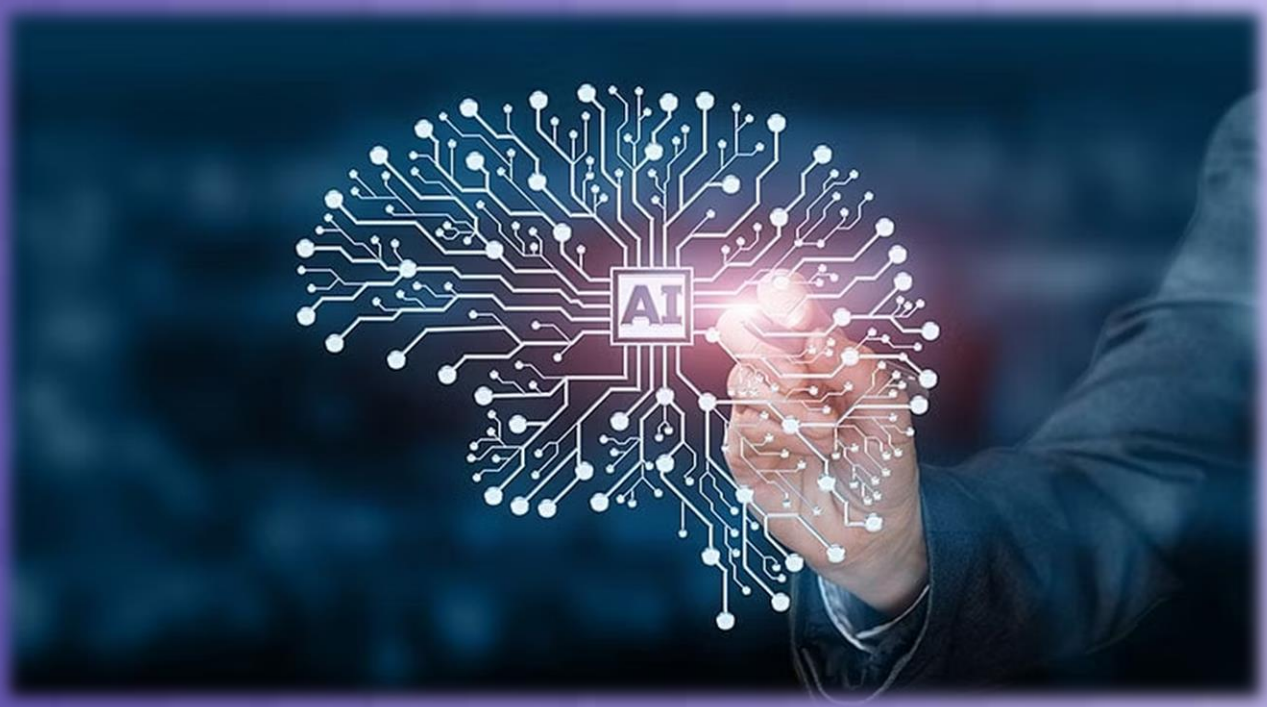
Міністерство освіти і науки України
Державний університет телекомунікацій

Кисіль Т.М., Звенігородський О.С., Фесенко М.А.

ОСНОВИ ШТУЧНОГО ІНТЕЛЕКТУ



Методичні рекомендації до виконання практичних завдань для здобувачів ступеня бакалавра освітньої програми «Штучний інтелект» за спеціальністю 122 «Комп'ютерні науки»

Київ - 2022

Міністерство освіти і науки України
Державний університет телекомунікацій

Кисіль Т.М., Звенігородський О.С., Фесенко М.А.

ОСНОВИ ШТУЧНОГО ІНТЕЛЕКТУ

Методичні рекомендації до виконання практичних завдань для здобувачів ступеня бакалавра освітньої програми «Штучний інтелект» за спеціальністю 122 «Комп’ютерні науки»

Київ – 2022

УДК 004.832

Рецензенти: **Савченко В.А**., доктор технічних наук, професор, директор Навчально-наукового інституту Захисту інформації Державного університету телекомунікацій.
Корнага Я.І., доктор технічних наук, доцент, доцент кафедри технічної безпеки Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

Кисіль Т.М., Звенігородський О.С., Фесенко М.А.
Основи штучного інтелекту. – Методичні рекомендації до виконання практичних завдань для здобувачів ступеня бакалавра освітньої програми «Штучний інтелект» за спеціальністю 122 «Комп'ютерні науки». – К: ДУТ, 2022. – 112 с.

Методичні рекомендації містять практичні завдання основ штучного інтелекту: приклади виконання завдань, тексти лабораторних та практичних робіт (контрольні питання, завдання до самостійного виконання, вимоги до захисту робіт). Методичні рекомендації використовуються на заняттях дисципліни «Основи штучного інтелекту» для студентів спеціальності 122 «Комп'ютерні науки» освітньої програми «Штучний інтелект»

Рекомендовано на засіданні вченої ради Навчально-наукового інституту інформаційних технологій (Протокол № 2 від 12.09.2022 року)

ЗМІСТ

ПЕРЕДМОВА

Штучний інтелект трактує здатність автоматизованих систем брати на себе окремі інтелектуальні функції людини, приймати оптимальні рішення на основі раніше одержаного досвіду і раціонального аналізу зовнішній їх впливів. Фахівці у галузі штучного інтелекту намагаються не лише з'ясувати природу інтелекту, але й створювати інтелектуальні інформаційні системи. Навчання основам штучного інтелекту передбачає освоєння теоретичних засад та практичних навичок фахового спрямування.

В ході даних методичних вказівок охоплено теми від базових принципів машинного навчання до практичних аспектів розробки концептуальних моделей штучних нейронних мереж. Тематика методичних рекомендацій охоплює не тільки теоретичні аспекти, а й прикладні завдання, які стосуються задач обробки природної мови, комп'ютерного зору, розпізнаванню зображень, рукописних текстів та інших застосувань штучного інтелекту.

Методичні матеріали рекомендовані студентам з метою набуття практичних навичок та закріплення ними теоретичних знань за напрямком штучний інтелект з можливістю:

- знайомства з принципами побудови продукційних та експертних систем мовою функціонально-логічного програмування Prolog;
- організації нейронних мереж та особливостями семантичного моделювання в системах штучного інтелекту мовою програмування Python;

Методичні матеріали «Основи штучного інтелекту» забезпечують комплекс теоретичних та практичних знань, в яких наведено:

- завдання до виконання лабораторних, практичних та самостійних робіт;
- теоретичні відомості до кожної теми навчальної дисципліни;
- приклади виконання типових завдань, що сприяє цілеспрямованому вивченню та поглибленому засвоєнню навчального матеріалу.

Тематики практичних завдань відповідають змісту теоретичного курсу навчальної дисципліни «Основи штучного інтелекту». Вивчення матеріалів дисципліни побудовано відповідно до вимог кредитно-модульної системи оцінювання знань. Програмою дисципліни передбачено виконання чотирьох лабораторних та практичних робіт першого модулю та чотирьох робіт другого. Подані в методичних рекомендаціях практичні завдання охоплюють три рівні складності та стимулюють студентів до накопичення бонусних балів, а також розвитку особистісних навичок за фаховим напрямком.

На виконання та захист кожної роботи відводиться чотири академічні години. За відведений на практичне виконання час студент повинен:

− отримати у викладача варіант завдання для індивідуального виконання;
− відповісти на поставлені контрольні запитання;

− підготувати звіт по виконаній роботі.

Звіт про виконання лабораторної роботи має містити: титульний аркуш, тему та мету роботи, основні теоретичні відомості, порядок виконання роботи та висновки. До оформлення звіту виносяться наступні вимоги:

− робота оформлюється на аркушах формату А4 відповідно до затвердженого стандарту «ДСТУ 4163:2020»;

− титульний аркуш містить назву дисципліни, тема роботи, ким виконано роботу (прізвище, ім’я, по батькові, навчально-науковий інститут та номер групи), ким прийнято роботу;

− основна частина звіту має містити вхідні дані, текст програми, отримані результати та висновки проведеного дослідження.

Методичні рекомендації складено на основі лекційних матеріалів, які викладаються студентам відповідно до навчальної програми дисципліни «Основи штучного інтелекту» напряму підготовки 122 «Комп’ютерні науки» освітньої програми «Штучний інтелект». Дисципліна «Основи штучного інтелекту» є базовою та надає підґрунтя для вивчення дисциплін такі як, «Штучний інтелект», «Штучні нейронні мережі», «Основи Big Data», «Аналіз даних Data Science», «Теорія розпізнавання зображень» на старших курсах освітньої програми бакалаврського рівня освіти в Державному університеті телекомунікацій.

Практичне завдання 1
Створення віртуальних помічників за допомогою онлайн-конструктора Engati.

Мета. Набути практичних навичок по створенню віртуальних помічників з імітацією інтелектуальних діалогів на платформі **Engati** та їх інтеграції на веб-ресурсах, в соціальних мережах, мобільних додатках.

Завдання 1. Створити чат-бот засобами онлайн-конструктора Engati:

1. Створити чат-бота в конструкторі **https://www.engati.com/** :
   - задати ім'я віртуального помічника (**Create a bot - Bot name**)
   - вибрати шаблон (**Blank Bot Canvas**)
   - створити команди чат-бота за заданою архітектурою (табл. 1), забезпечивши привітання, невідомі дії користувача, діалогове меню та прощання за альтернативними відповідями.
2. Правила для створення чат-бота (розділ **Paths):**
   - команди для бота **Create Path** (задати назву правила та **Random Messages** з альтернативними відповідями);
   - 1 та 2 пункти меню (**Send Message**);
   - 3 пункт меню (**Send Message** та **Random Messages**);
   - влаштовані стандартні правила:
     - ✓ для нового користувача (**Send Message** з привітанням та Trigger **a path**);
     - ✓ для постійного користувача **Greeting Message** (**Send Message** з привітанням та **Trigger a path** з меню);
     - ✓ невідома команда **Default Message** (**Trigger a path** з меню)
3. Запити користувача (розділ **FAQ**)
   - варіанти можливих запитів користувача **User Question;**
   - відповідні правила бота **Bot Response.**
4. Інтеграція чат-бота на веб-ресурсі (**Deployment - Website Chatbot**).
   - вказати **Website URL** ресурсу;
   - встановити відповідні налаштування аватара чат-бота;
   - отриманий **Script** впровадити на веб-ресурсі;
5. Проаналізувати статистику використання чат-бота:
   - користувачів та відповідні запити;
   - аналітика даних за взаємодією користувач/чат-бот.

Таблиця 1 – Приблизна архітектура команд чат-бота

Команди користувача	Дії чат-бота
Привіт / Вітаю / Добрий ранок / Добрий день / Добрий вечір	Привіт! /Вітаю! /Вітаю! Раді бачити Вас!
Невідома дія	Я не знаю / Не зрозуміло мені / Перепрошую
Прощавай / До побачення / Всього найкращого / До зустрічі	Прощавай / До побачення / До зустрічі / Всього найкращого / Приходьте ще до нас
МЕНЮ ДІЙ:	Виберіть пункт для переходу: 1-лекції 2-практичні 3-додаткові джерела
Пункт меню 1	***Лекції:*** Лекція 1 Лекція 2
Пункт меню 2	***Практичні:*** Практична 1 Практична 2
Пункт меню 3	***Рекомендуємо ознайомитись:*** Бородянський, Руденко “Нейронні мережі” Братко “Алгоритми штучного інтелекту на мові Пролог” Коцовський “Методи та системи штучного інтелекту” Ручкін, Фулін “Універсальний штучний інтелект”

Завдання 2. Засобами онлайн-конструктора Engati, необхідно модернізувати створеного в завданні 1 чат-бота за модернізованою структурою.

1. Додати карту головного меню (**Send Carousel**) та забезпечити (рис. 1):
   - кнопки переходу до відповідних розділів (**Add Button**);
   - логотип меню (**Add Image**);

- назву (**Heading**) та короткий опис меню (**Subheading**).
- до відповідних розділів вказати певні повідомлення взаємодії (**Trigger a path**).

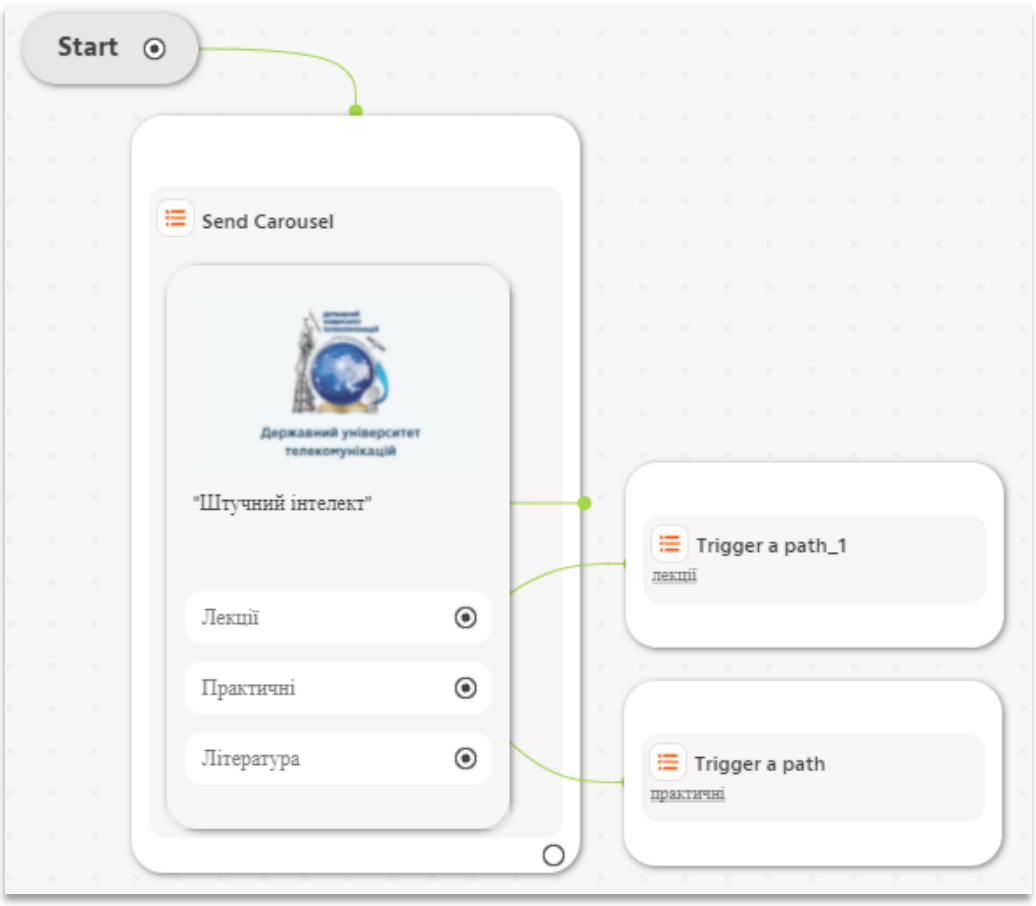


Рисунок 1. – Приблизна структура головного меню

2. Модернізувати правила ***Лекції/Практичні/Література*** в розділі **Paths** за складною структурою (рис. 2), додавши**:**
   - повідомлення **Send Message (with options)**, в якому вказати **Message**, **Attribute Name**, **Option 1** та **Option 2** з параметрами **Option title, Option Value** і **Connection**;
   - повідомлення (**Send Message)** з посиланням на задані змінні величини;
   - повідомлення (**Trigger a path**) з посиланням на відповідні правила.
3. Модернізувати стандартні правила в розділі **Paths**:
   - для нового користувача (**Welcome New User**) задати:
     - запит (**Request User Data)** на введення даних від користувача імені, електронної адреси та ін. даних (рис. 3);
     - повідомлення (**Send Message)** з використанням відповідних змінних;
   - для постійного користувача (**Greeting Message)** відповідні запити та повідомлення з використанням стандартних/користувацьких змінних;
   - команди невідомих дій (**Default Message)** за довільною структурою.

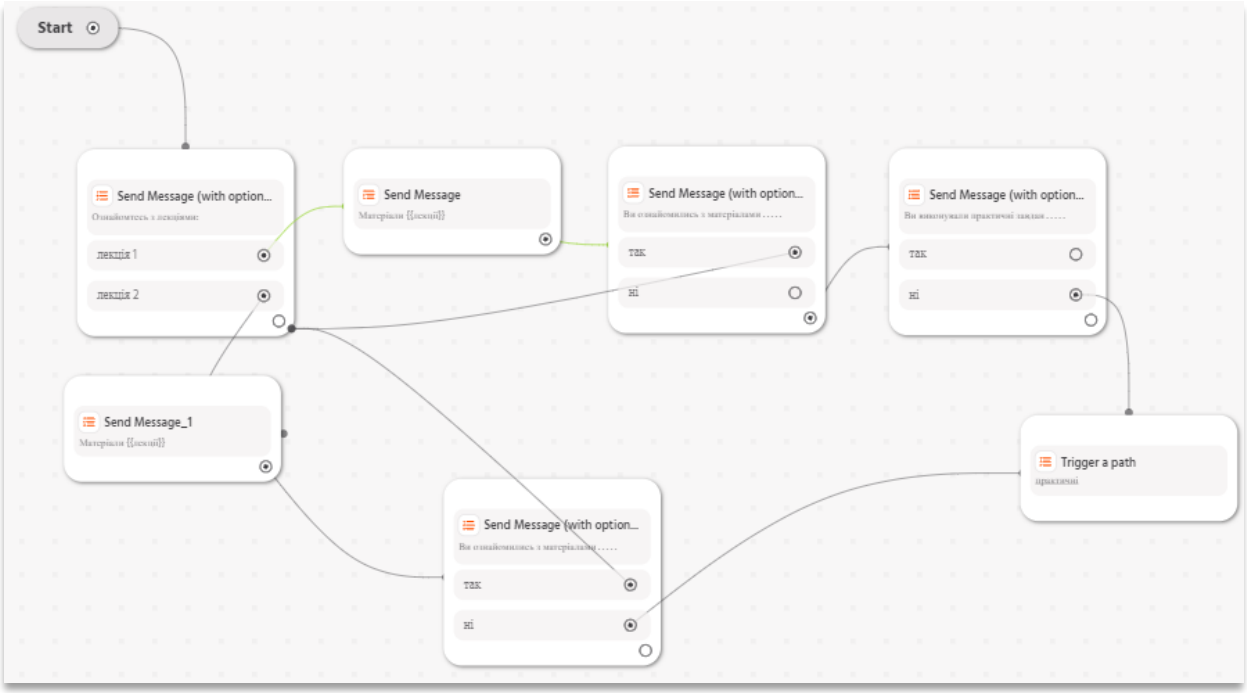


Рисунок 2. – Приблизна структура модернізованих правил.

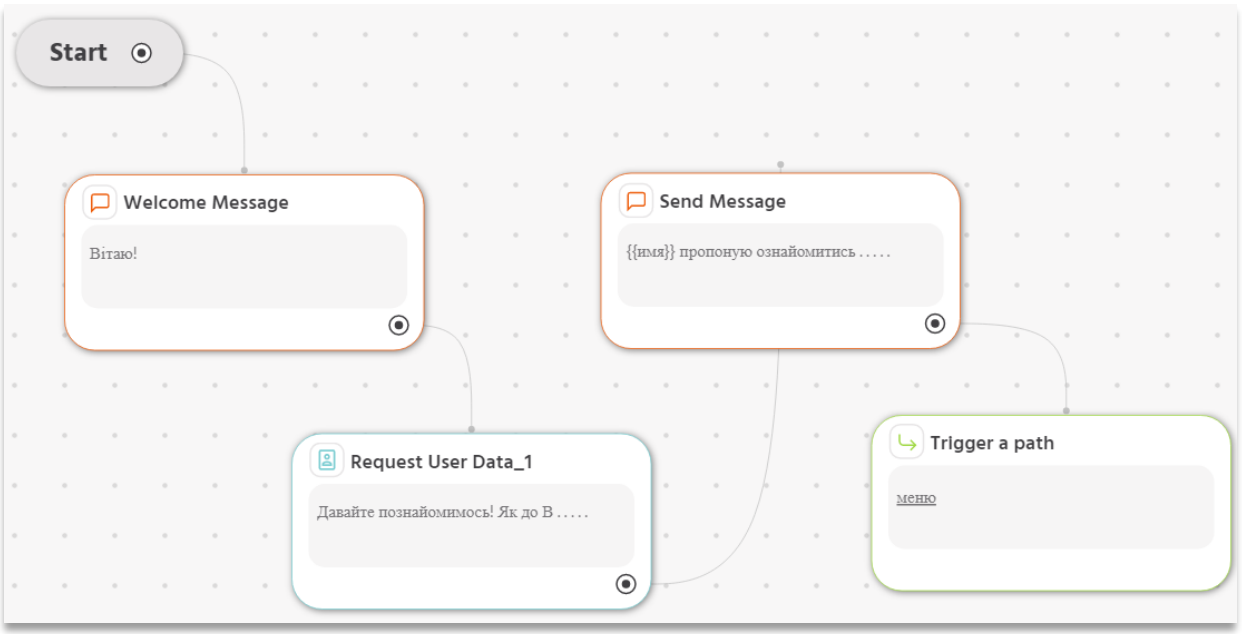


Рисунок 3. – Приблизна структура модернізованого правила **Welcome New User**.

4. Модернізувати створені запити користувача в розділ **FAQ**:
   - організувати діалоговий режим у відповідних розділах, додавши 5-10 можливих запитань/відповідей (**User Question**);
   - протестувати організований діалог у відповідних запитах-правилах;
   - за необхідністю натренувати чат-бота на відповідний діалог.
5. Проаналізувати статистику використання чат-бота:
   - користувачів та відповідні запити;
   - аналітика даних за взаємодією користувач/чат-бот.

<u>Завдання 3</u>. Створити чат-бот засобами онлайн-конструктора Engati відповідно до варіанту (табл. 2):

1. Створити архітектуру чат-бота з відповідними правилами та запитами (табл. 3);
2. Інтегрувати чат-бот в месенджері ***Telegram*** (рис. 4);

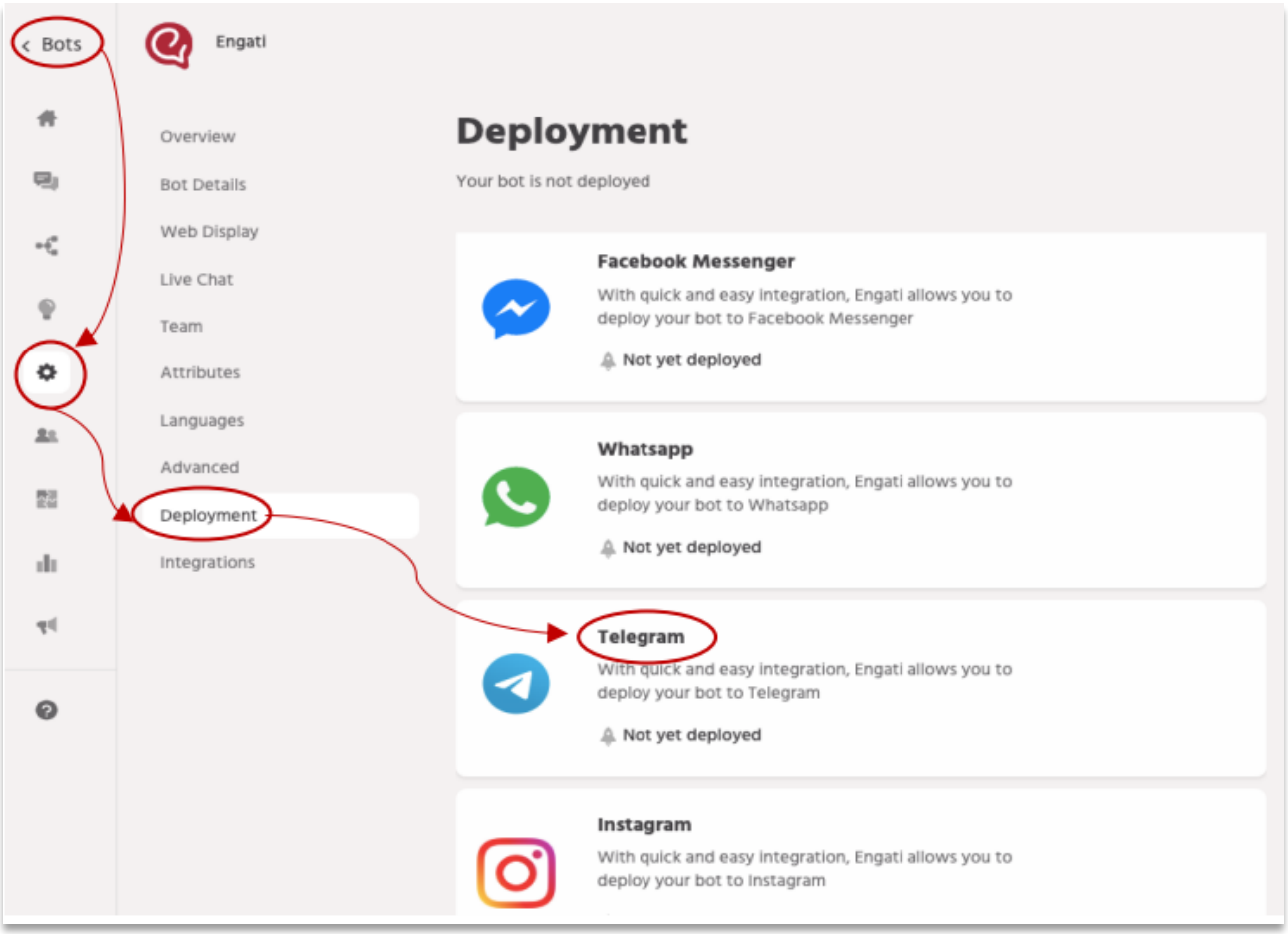


Рисунок 4. – Інтеграція чат-бота в месенджері ***Telegram***.

3. Створення бота в Telegram:
   - необхідно в Telegram знайти "***BotFather***" у полі "***Пошук повідомлень або користувачів***" (рис. 5);

Рисунок 5. – Створення чат-бота в ***Telegram***-месенджері.

   - із **списку** команд знайти та/або вибрати команду *"**/newbot***" для створення бота;
   - вказати ім’я (лат.) чат-бота, наприклад «***MyDemo***»;

- вказати користувацьке ім'я чат-бота (лат.), яке повинне закінчуватись "bot". Наприклад, "***MyDemoBot***" або "***My_Demo_Bot***".
- виконати команду ***"/mybots"*** для перегляду/редагування параметрів створеного бота.

4. Отримання токена для авторизації бота:
   - виконати в "***BotFather***" команду ***"/token"*** для отримання токена доступу (рис. 6)

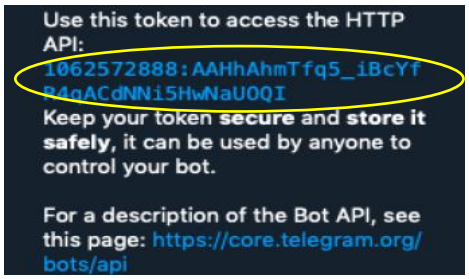


Рисунок 6. – Визначення токена чат-бота в ***Telegram***-месенджері.

   - скопіювати отриманий токен бота та вставити його дані в полі сервісу ***Engati*** (рис. 7)

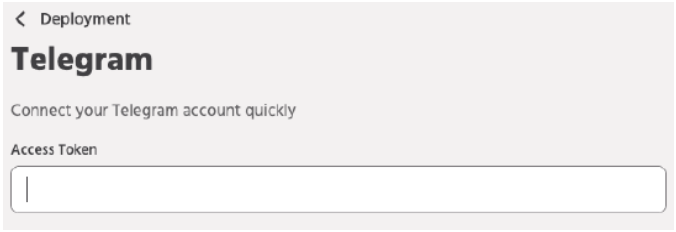


Рисунок 7. – Визначення токена чат-бота в ***Telegram***-месенджері.

   - протестувати бота в Telegram-месенджері.

5. Проаналізувати статистику чат-бота (за необхідністю додати відповідні правила та запити, внести зміни).

Таблиця 2 – Перелік рекомендованих предметних областей для створення віртуальних асистентів.

№ з/п	Предметна область
1.	Замовлення піци з ресторану
2.	Замовлення косметики в мережі магазинів Єва
3.	Замовлення гарячих напоїв
4.	Замовлення верхнього одягу
5.	Замовлення суші з суші-бару
6.	Замовлення взуття
7.	Замовлення продуктових товарів
8.	Замовлення побутової техніки
9.	Замовлення холодних напоїв

№ з/п	Предметна область
10.	Замовлення відеопродукції
11.	Замовлення продукції Рошен
12.	Замовлення мобільних телефонів
13.	Замовлення корму для тварин
14.	Продаж та оренда житла
15.	Замовлення телепродукції
16.	Замовлення корму для птахів
17.	Замовлення та продаж букетів з цукерок
18.	Замовлення корму для декоративних риб
19.	Замовлення та продаж букетів
20.	Замовлення спортивного одягу
21.	Замовлення відео-послуг
22.	Реалізація послуг по ремонту автомобілів
23.	Замовлення комп'ютерної техніки
24.	Замовлення фото-послуг
25.	Реалізація послуг по прокату автомобілів
26.	Замовлення продукції гіпермаркету Епіцентр
27.	Продаж та реалізація канцелярських товарів
28.	Замовлення та продаж аксесуарів
29.	Бронювання концертних квитків
30.	Замовлення та продаж меблів

Таблиця 3 – Перелік команд та архітектура віртуального помічника за обраною предметною областю.

Ім'я команди	Тип повідомлення	Текст повідомлення/ альтернативні відповіді
Правила (Paths)		
…		
Запити (FAQ)		
…		

Зміст звіту

1. Титульна сторінка.
2. Тема та мета практичного заняття
3. Архітектуру проектованих віртуальних помічників

4. Опис виконання завдань по пунктам з наведеними рисунками та скріншотами.
5. Посилання на веб-ресурси з доданим віртуальними помічниками
6. Висновки, відповідно до проведеного аналізу.

Теоретичні відомості

Платформа ***Engati*** дозволяє проектувати та конфігурувати чат-боти відповідного спрямування, залежно від напрямку використання. Завдяки даній платформі можна вибирати з низки варіантів відповідні процеси автоматизації.

Створення чат-бота

Щоб створити бота на платформі, необхідно вказати команду «***Створити бота***» після реєстрації та входу. Влаштований в системі майстер (рис. 8), допоможе спростити процес створення чат-бота.

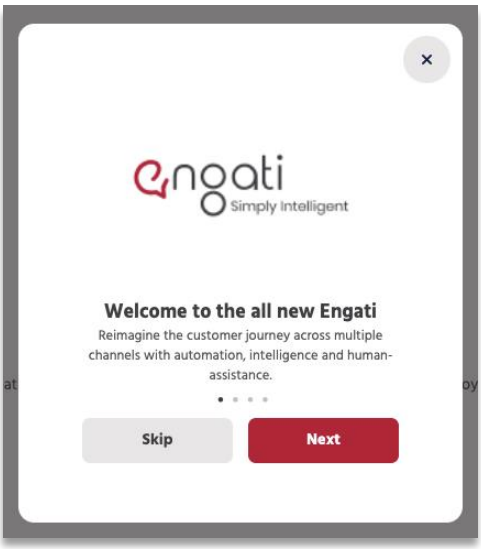


Рисунок 8. – Процес створення чат-бота на платформі ***Engati***.

Після вибору команди «***Створити бота***» відкривається діалогове вікно, в якому необхідно вказати ім'я бота (на кирил. або лат.) та підтвердити процес створення кнопкою «***Створити бота***» (рис. 9).

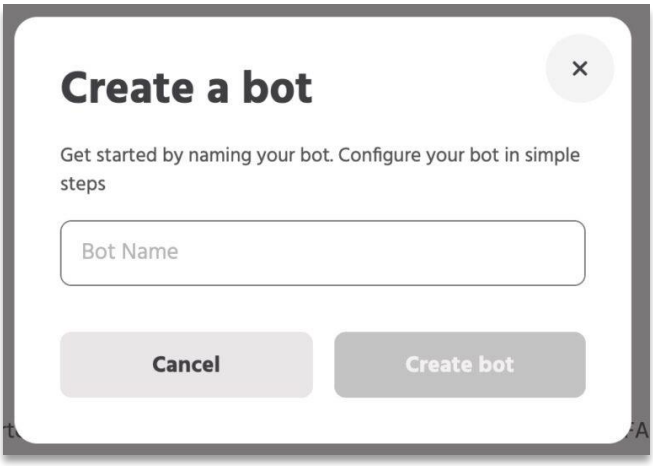


Рисунок 9. – Процес створення чат-бота на платформі ***Engati***.

На наступному етапі відображаються стандартні шаблони, які запропоновані платформою ***Engati*** (рис. 10). Рекомендується створювати чат-боти за пустим шаблоном «***Blank Bot Canvas***».

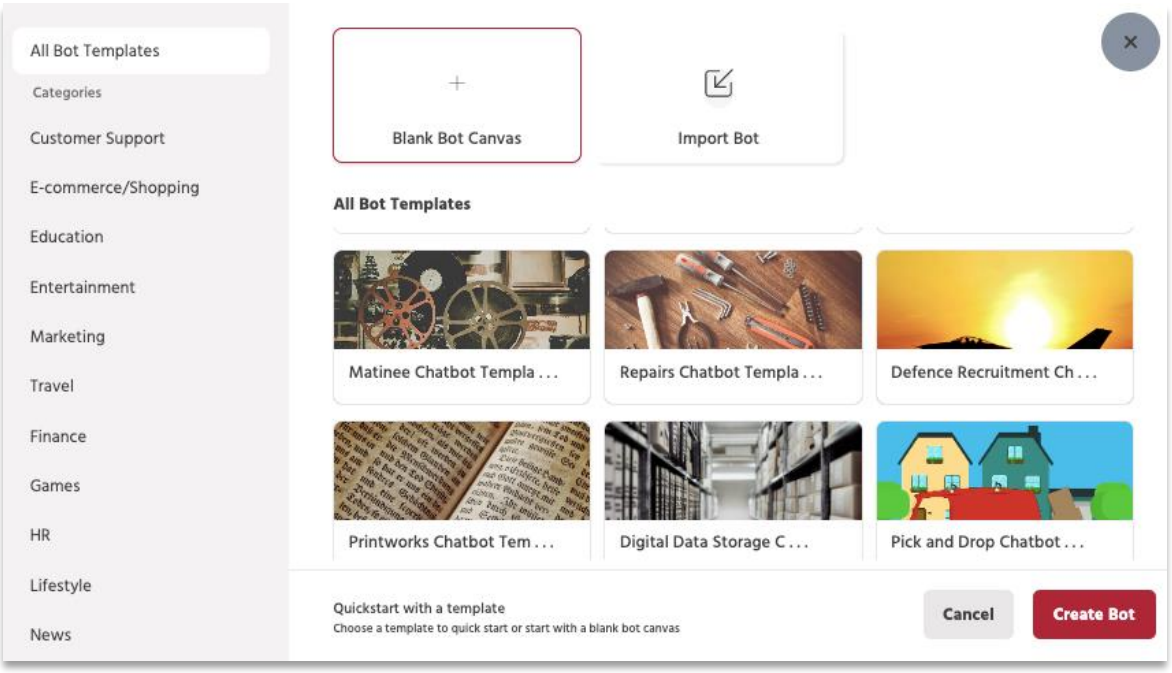


Рисунок 10. – Список стандартних шаблонів чат-ботів платформи ***Engati***.

Налаштування чат-бота

Після створення бота необхідно перейдете на сторінку налаштувань, яка містить вказівки щодо наступних кроків (рис. 11):

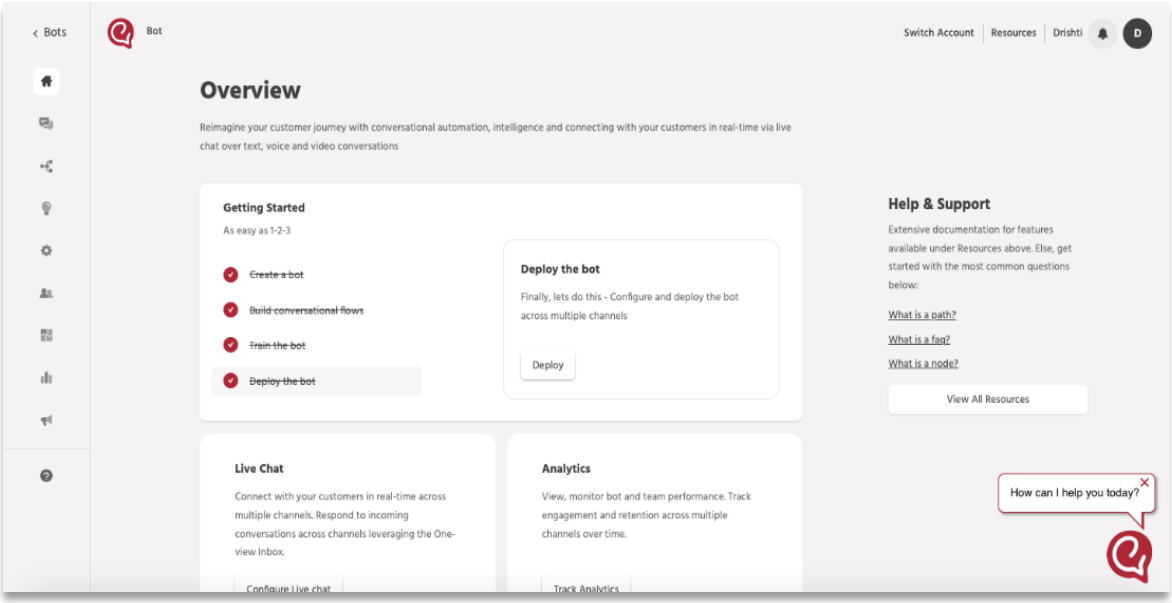


Рисунок 11. – Налаштування чат-бота на платформі ***Engati***.

- *Створення розмовних потоків* – необхідно вибрати опцію «***Побудувати розмовні потоки***» або перейти до пункту «***Побудувати***» в лівому навігаційному меню.

- *Навчання бота* – необхідно перейти до розділу «***Навчання»*** в навігаційному меню, щоб додати поширені запитання, сутності та документи для когнітивного пошуку. Бот посилається на поширені запитання, засновані на знаннях, та відповідає на вхідні запити користувачів.
- *Розгортання бота* – розгорнути чат-бот можна на 15+ каналах за допомогою простого процесу розгортання. Щоб розгорнути бота, необхідно перейти до пункту меню ***Налаштування → Розгортання***.

Більш детально ознайомитись з платформою ***Engati*** та процесами проектування чат-ботів можна на офіційному сайті [6].

Контрольні запитання

1. Що являє собою програмний агент?
2. Поясніть та проаналізуйте властивості агентів: автономність, адаптивність, комунікативність, здатність до співробітництва, персоніфікованість, мобільність.
3. З якою метою застосовуються агенти, використання яких залежить від завдань?
4. Що таке розважальні агенти?
5. Які методи використовуються для надання інтелектуальності програмним агентам?
6. Проаналізуйте основні функції обраного інструментарію розробки віртуального агента: внутрішня структура, параметри, основні змінні, їх призначення та використання.
7. Що є предметом вивчення штучного інтелекту?
8. Дайте визначення інтелекту і поясність його сутність.
9. Дайте визначення інтелектуального агента.
10. Яку структуру має інтелектуальний агент? Наведіть приклад.
11. Наведіть приклади різних типів агентів. Який із перерахованих типів є, на вашу думку, найефективнішим?
12. Які типи проблемних середовищ ви знаєте? У чому полягають їх відмінності?

Практичне завдання 2
Розробка Telegram – ботів по адмініструванню інтернет-ресурсів на платформі SendPulse

Мета. Набути практичних навичок по створенню віртуальних помічників з імітацією інтелектуальних діалогів для ***telegram-каналів***; їх інтеграції на веб-ресурсах, в соціальних мережах та мобільних додатках.

Завдання 1. Створити чат-бот в telegram – месенджері по адмініструванню інтернет-ресурсів:

1. Створити чат-бот в ***Telegram-месенджері*** (див. теоретичні відомості практичного завдання 1, 2)**.**
2. Інтегрувати токен створеного чат-бота в конструкторі ***SendPulse https://sendpulse.com/ua***
3. Відповідно до варіанту наведеного в ***табл. 4***, створити архітектуру чат-бота. Передбачити в архітектурі відповідні факти, правила, змінні, в результаті чого, користувачам буде надаватись можливість замовлення відповідних послуг (консультації, замовлення товарів, запис на прийом, рекламні розсилки та інше).
4. Перевірити роботу створеного чат-бота в ***Telegram-месенджері*** (за необхідністю змінити архітектуру: додати відповідні факти, правила, запити, зв'язки, тощо).
5. Проаналізувати статистику використання чат-бота:
   - кількість користувачів, відповідні запити/діалоги від користувачів;
   - аналітику даних за взаємодією користувач/чат-бот.

Таблиця 4 – Перелік рекомендованих предметних областей для створення віртуальних помічників.

№ з/п	Предметна область
1.	Замовлення піци з ресторану
2.	Замовлення косметики в мережі магазинів Єва
3.	Замовлення гарячих напоїв
4.	Замовлення верхнього одягу
5.	Замовлення суші з суші-бару
6.	Замовлення взуття
7.	Замовлення продуктових товарів
8.	Замовлення побутової техніки
9.	Замовлення холодних напоїв
10.	Замовлення відеопродукції

№ з/п	Предметна область
11.	Замовлення продукції Рошен
12.	Замовлення мобільних телефонів
13.	Замовлення корму для тварин
14.	Продаж та оренда житла
15.	Замовлення телепродукції
16.	Замовлення корму для птахів
17.	Замовлення та продаж букетів з цукерок
18.	Замовлення корму для декоративних риб
19.	Замовлення та продаж букетів
20.	Замовлення спортивного одягу
21.	Замовлення відео-послуг
22.	Реалізація послуг по ремонту автомобілів
23.	Замовлення комп'ютерної техніки
24.	Замовлення фото-послуг
25.	Реалізація послуг по прокату автомобілів
26.	Замовлення продукції гіпермаркету Епіцентр
27.	Продаж та реалізація канцелярських товарів
28.	Замовлення та продаж аксесуарів
29.	Бронювання концертних квитків
30.	Замовлення та продаж меблів

Завдання 2. Створити чат-бот в одному із конструкторів: ***ManyChat, Chatfuel, ChattyPeople, FlowXO, Botsify, HelpCrunch, Kwizbot, Boost.ai, Tidio, Verloop, Giosg, Fasttrack, Dexter, Gerabot, Chatforma, Chatbot, Salebot, Konvertbot***. Інтегрувати створеного чат-бота на веб-ресурсах та/або в месенджерах, соціальних мережах. Проаналізувати статистику використання чат-бота: кількість користувачів, відповідні запити/діалоги від користувачів, аналітику даних за взаємодією користувач/чат-бот.

Зміст звіту

1. Титульна сторінка.
2. Тема та мета практичного заняття
3. Архітектуру проектованих віртуальних помічників
4. Опис виконання завдань по пунктам з наведеними рисунками та скріншотами.
5. Посилання на веб-ресурси з доданим віртуальними помічниками
6. Висновки щодо виконаних практичних завдань.

Теоретичні відомості

Створення чат-бота в Telegram – месенджері. Для створення чат-бота в Telegram необхідно:

- в Telegram знайти "***BotFather***" у полі "***Пошук повідомлень або користувачів***";
- із списку команд знайти або вибрати команду ***"/newbot***" для створення чат-бота;
- вказати ім'я (лат.) вашого бота, наприклад «***MyDemo***»;
- вказати користувацьке ім'я чат-бота (лат.), яке повинно закінчуватись "bot" (наприклад, "***MyDemoBot***" Або "***My_Demo_Bot***");
- виконати команду ***"/mybots***" для перегляду/редагування параметрів створеного бота (рис. 12);

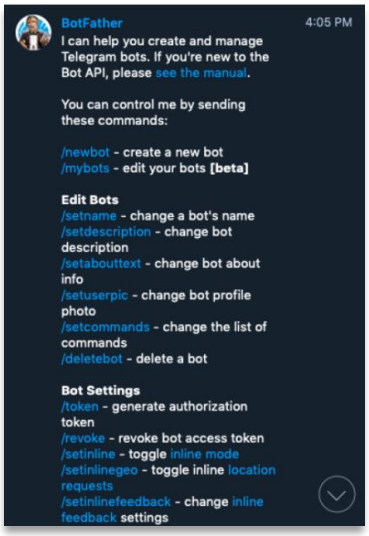


Рисунок 12. – Перелік команд по налаштуванню чат-бота в ***Telegram – месенджері***.

- виконати в "***BotFather***" команду "***/token***" для отримання доступу до бота;
- скопіювати отриманий токен чат-бота та вставити його дані в полі сервісу ***SendPulse*** (рис. 13)

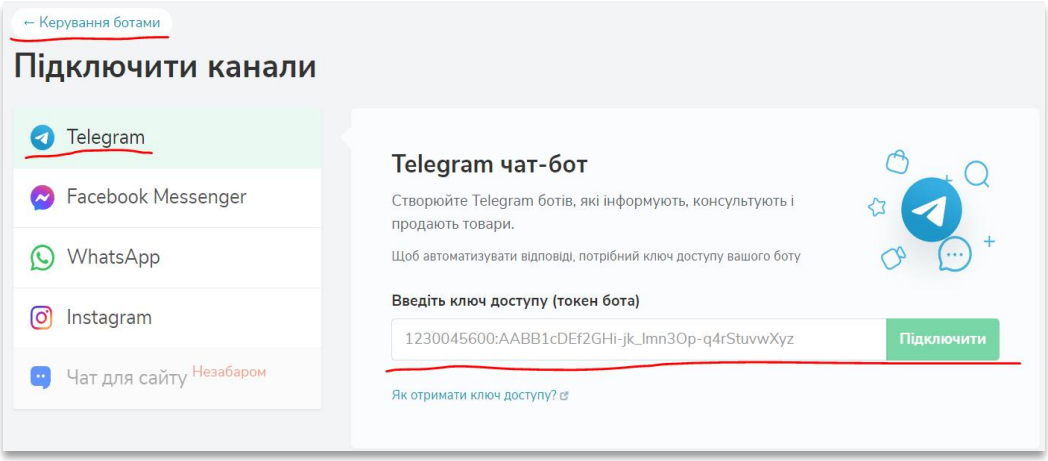


Рисунок 13. – Налаштування чат-бота на платформі ***SendPulse***.

- протестувати створеного чат-бота в Telegram-месенджері.

Створення архітектури чат-бота в конструкторі SendPulse. Для створення архітектури чат-бота на платформі ***SendPulse*** необхідно створити сценарії стандартних ланцюжків, які розглянемо нижче.

Створення сценарій-ланцюжка для чат-бота. Конструктор ланцюжків — головний інструмент налаштування вашого бота, який може вітати нових підписників, консультувати, продавати товари та повідомляти менеджера про питання підписника в чат-боті. А також збирати та фільтрувати дані та відправляти їх до вас у систему для подальшої обробки.

Достатньо вказати ключові слова, визначити структуру бота та перемістити елементи, налаштовуючи їх один за одним.

Вибір старту ланцюжка. За замовчуванням доступні наступні ланцюжки: «***Вітальна серія***» (рис. 17), «***Стандартна відповідь***» (рис. 16), «***Відписка від бота***» та швидкі реакції на історію у вигляді емодзі. Також можна створювати власні тригери.

Достатньо вибрати створеного чат-бота, перейти на вкладку «***Структура бота***» та вибрати тип запуску для редагування (рис. 14).

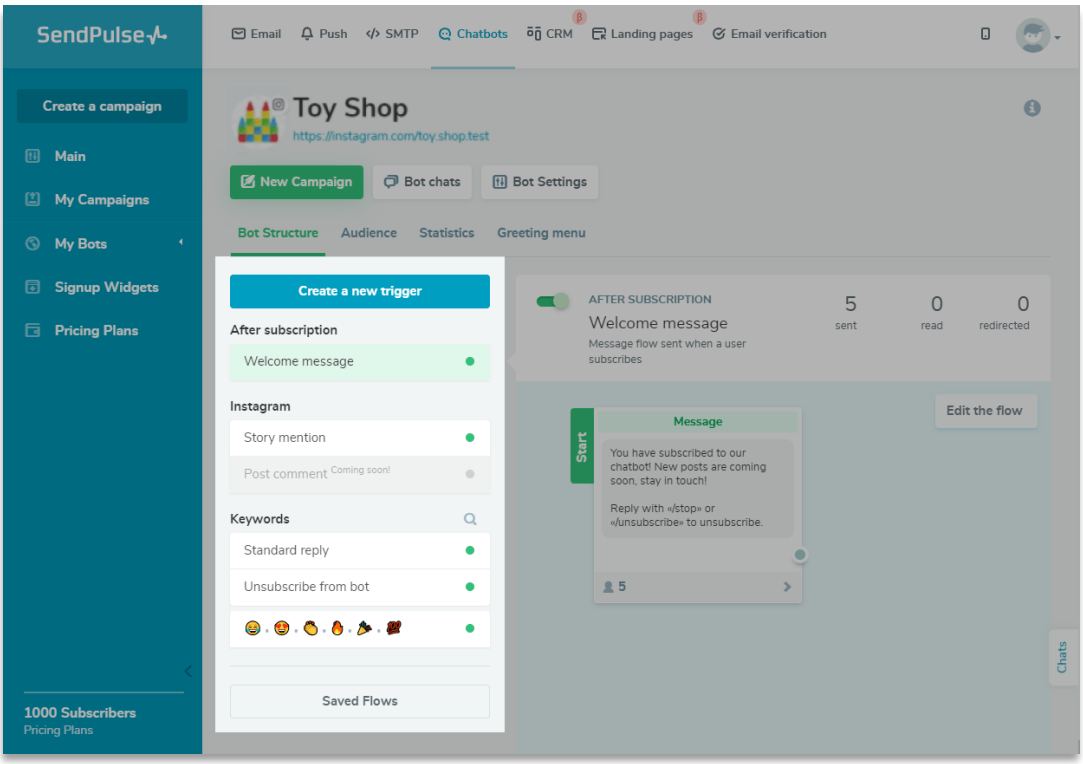


Рисунок 14. – Редагування структури чат-бота на платформі ***SendPulse***.

Вітальний ланцюжок. Даний ланцюжок (рис. 15) знайомить користувачів зі створеним чат-ботом, в ньому можна розповісти про бота і його можливості — чим чат-бот буде корисним, яку інформацію може надавати, як часто буде надсилати розсилки відповідного змісту, тощо.

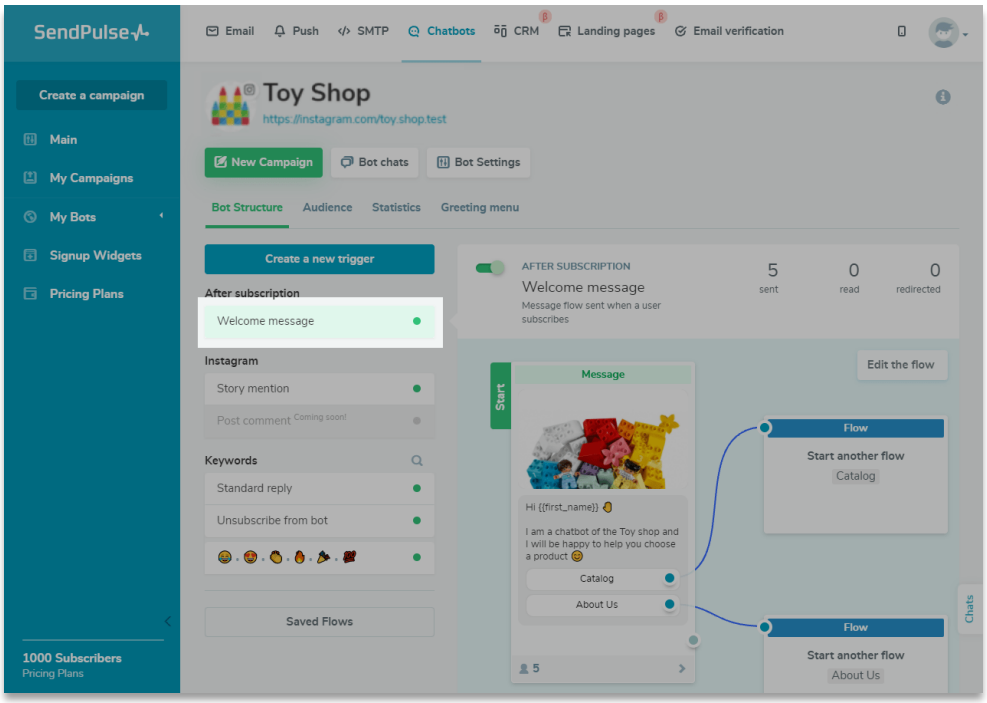


Рисунок 15. – Налаштування чат-бота на платформі ***SendPulse***.

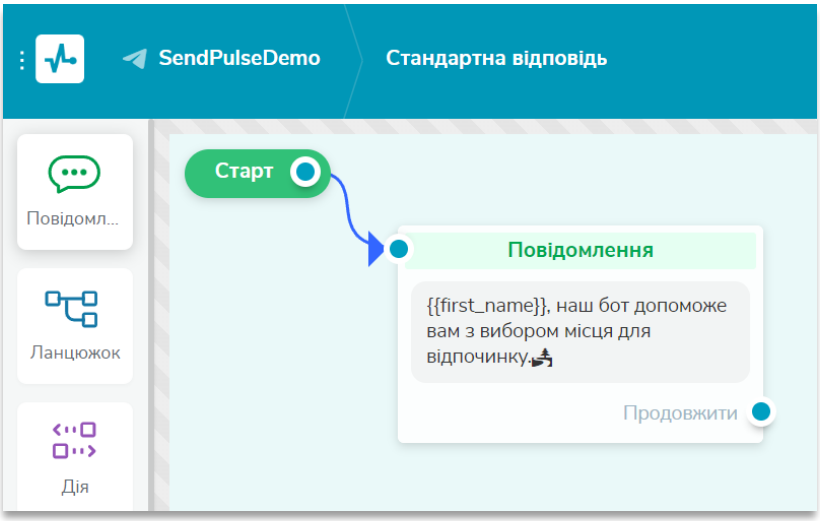


Рисунок 16. – Налаштування чат-бота на платформі ***SendPulse***.

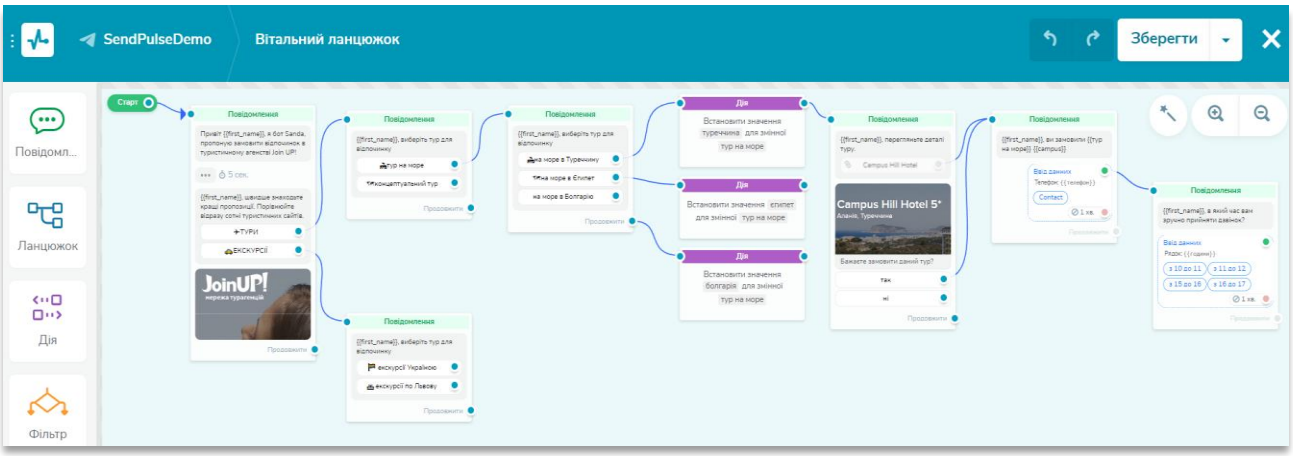


Рисунок 17. – Налаштування чат-бота на платформі ***SendPulse***.

Більш детально ознайомитись з платформою ***SendPulse*** та процесами проектування чат-ботів можна на офіційному сайті [8].

Контрольні запитання

1. З якою метою застосовуються чат-боти, які завдання вони можуть виконувати?
2. Назвіть основні призначення чат-ботів?
3. Назвіть основні функції запровадження чат-ботів.
4. Які методи використовуються для надання інтелектуальності програмним агентам?
5. Проаналізуйте основні інструментарію розробки чат-ботів на різних платформах: внутрішня структура, параметри, основні змінні, їх призначення та використання.
6. Дайте визначення інтелектуальності чат-боту, поясність його сутність.
7. Наведіть приклади інтелектуальних чат-ботів та їх переваги/недоліки при створенні на різних платформах.
8. Назвіть платформи на яких створені інтелектуальні чат-боти.
9. Яку структуру має інтелектуальний агент? Наведіть приклад.
10. Назвіть платформи створення різних типів агентів. Яка платформа є, на вашу думку, найефективнішою?
11. Які типи проблемних середовищ ви знаєте? У чому полягають їх переваги та відмінності?

Практичне завдання 3
Створення інтелектуальних віртуальних асистентів на платформі Dialogflow.

Мета. Набути практичних навичок по створенню інтелектуальних віртуальних асистентів з адаптацією на різних платформах та підтримкою мови програмування **Python**; їх інтеграції на веб-ресурсах, в соціальних мережах та мобільних додатках.

Завдання 1. Створити чат-бот на базі платформи Dialogflow.

1. Створити чат-бот в сервісі ***https://dialogflow.cloud.google.com/*** командою (**Create a new agent**):
   - вказати назву асистента (лат. або кир.);
   - вибрати основну мову та робочу зону асистента (рис. 18);
   - підтвердження створення нового Google-проекту.

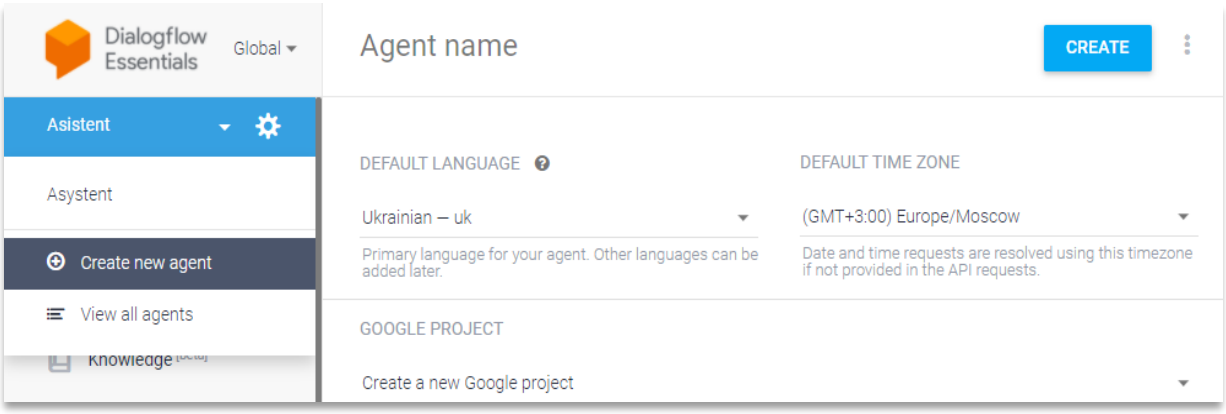


Рисунок 18. – Створення віртуального асистента на платформі **Dialogflow.**

2. Створити контекст стандартних правил:
   - **Default Welcome Intent** – привітання користувача, вказавши відповідні значення в розділах: **Contexts, Training phrases** (рис. 19)**, Events, Action (**рис. 20**), Response** (рис. 21);
   - **Default Fallback Intent** – контекст невідомих команд користувача.
3. Створити правила віртуального асистента (розділ **Intents)** відповідно до варіанту (**табл. 5**)**:**
   - для кожного правила задати розгалужену структуру (за вказаними відповідями **так/ні**);
   - для альтернативних відповідей правила вказати **Contexts, Training phrases, Events, Action, Response** з використанням локальних змінних.

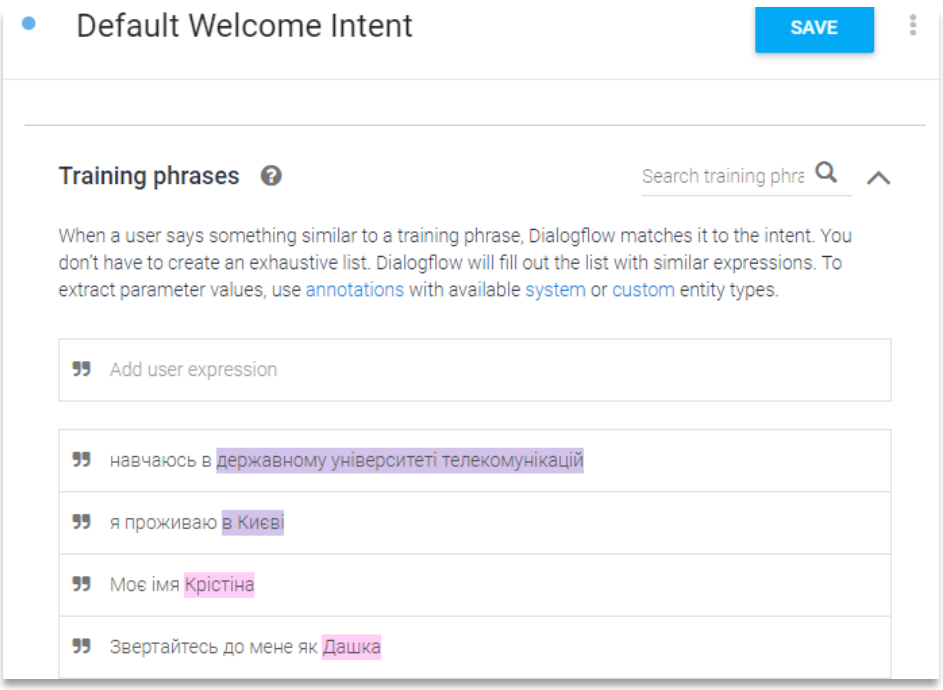


Рисунок 19. – Налаштування діалогу з користувачем в стандартному правилі

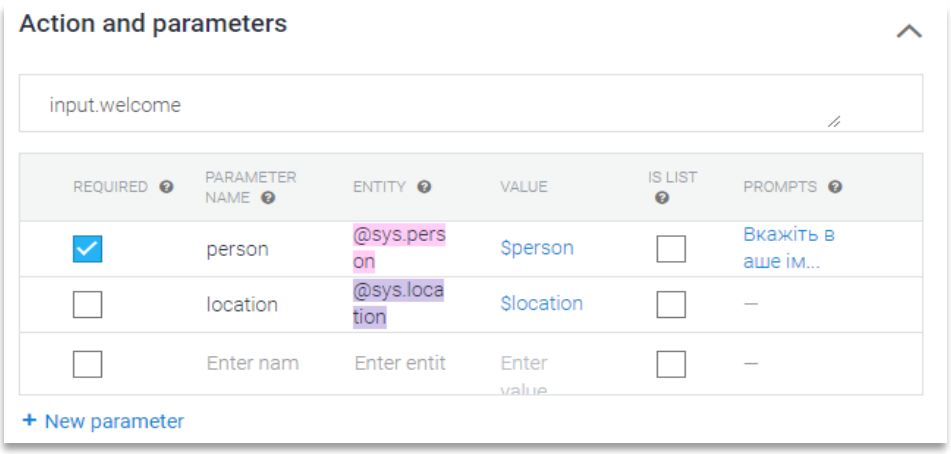


Рисунок 20. – Визначення локальних змінних в діалозі з користувачем.

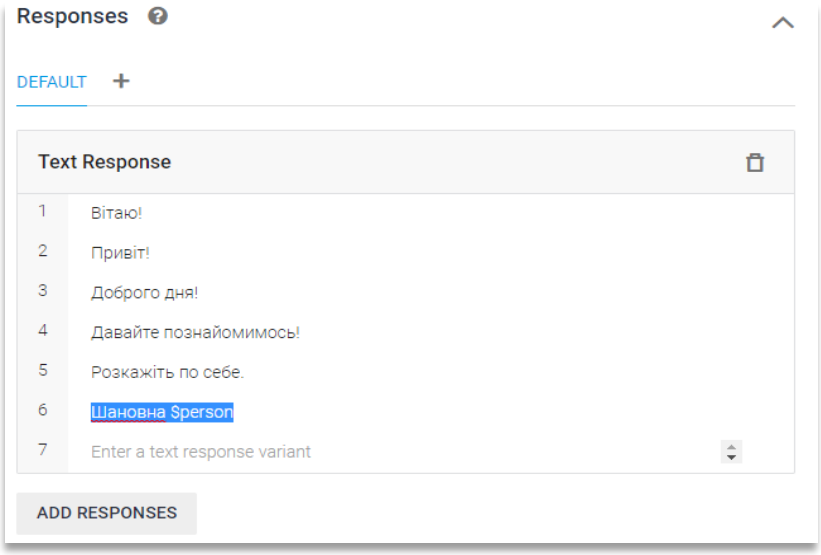


Рисунок 21. – Налаштування альтернативних відповідей чат-бота в стандартному правилі.

Таблиця 5 – Перелік рекомендованих предметних областей для створення віртуальних асистентів.

№ з/п	Предметна область
1.	Замовлення піци з ресторану
2.	Замовлення косметики в мережі магазинів Watson
3.	Замовлення гарячих напоїв
4.	Замовлення верхнього одягу
5.	Замовлення суші з суші-бару
6.	Замовлення взуття
7.	Замовлення продуктових товарів
8.	Замовлення побутової техніки
9.	Замовлення холодних напоїв
10.	Замовлення відеопродукції
11.	Замовлення продукції Рошен
12.	Замовлення мобільних телефонів
13.	Замовлення корму для тварин
14.	Продаж та оренда житла
15.	Замовлення телепродукції
16.	Замовлення корму для птахів
17.	Замовлення та продаж букетів з цукерок
18.	Замовлення корму для декоративних риб
19.	Замовлення та продаж букетів
20.	Замовлення спортивного одягу
21.	Замовлення відео-послуг
22.	Реалізація послуг по ремонту автомобілів
23.	Замовлення комп'ютерної техніки
24.	Замовлення фото-послуг
25.	Реалізація послуг по прокату автомобілів
26.	Замовлення продукції гіпермаркету Епіцентр
27.	Продаж та реалізація канцелярських товарів
28.	Замовлення та продаж аксесуарів
29.	Бронювання концертних квитків
30.	Замовлення та продаж меблів

4. Натренувати чат-бота на відповідний діалог (розділ **Training**) за обраною темою:
   - передбачити альтернативні відповіді з врахуванням відмінків для кожного правила;
   - протестувати створені правила в **Dialogflow.**
5. Створити сутності чат-бота за відповідними властивостями (розділ **Entities**)
6. Інтеграція чат-бота в телеграм месенджері (**Integrations - Telegram).**
   - в **Telegram**-месенджері знайти **BotFather;**
   - запустити на виконання чат-бота командою **/start**;
   - створити нового бота командою ***/newbot*** задати ім'я бота (лат. або кир.);
   - коротке ім'я бота (лат.) з обов'язковою приписком в кінці імені **_bot** або **bot;**
   - згенерований токен додати в текстовому полі **Telegram-token** (рис. 22);
   - задати налаштування та властивості створеного телеграм-бота командою ***/mybots***:
     - кнопка **Bot Settings** – опція **Inline Mode - Turn on** для включення повідомлень користувачів.
     - перейти по опції **Back to Settings** та зайти в **Group Privacy - Turn off** для відключення приватності груп.

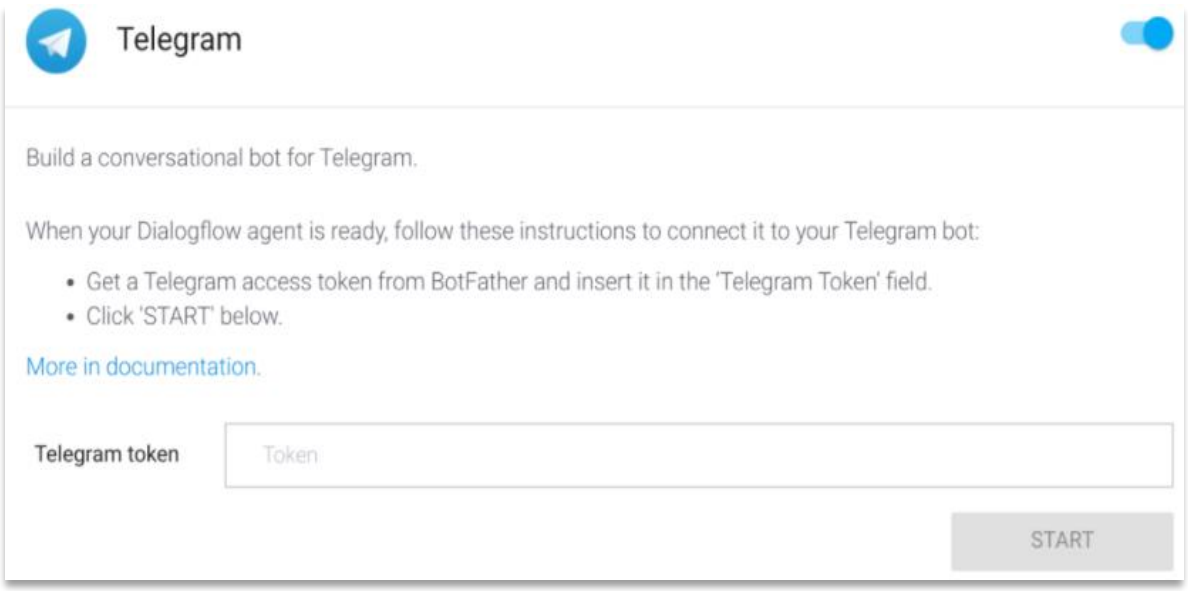


Рисунок 22. – Підключення телеграм-токена в ***Dialogflow***

7. Переглянути статистику використання чат-бота в ***Dialogflow***:
   - користувачів за відповідними діалогами (розділ History);
   - аналітика даних за взаємодією користувач/чат-бот.

Завдання 2[1]. Створити чат-бот програмними засобами відповідно до варіанту (табл. 5):

- мовою програмування **Python2** створити архітектуру чат-бота (інструкцію по встановленню та налаштуванню мови програмування **Python** 3.7.2 наведено в теоретичних відомостях даного практичного завдання);
- інтегрувати чат-бот на веб-ресурсах, месенджерах, соціальних мережах, тощо;
- проаналізувати статистику використання створеного чат-бота.

Зміст звіту

1. Титульна сторінка.
2. Тема та мета практичного заняття.
3. Архітектуру проектованих віртуальних асистентів.
4. Опис виконання завдань по пунктам з наведеними рисунками та скріншотами.
5. Посилання на веб-ресурси з доданим віртуальними асистентами.
6. Висновки щодо виконаного практичного завдання.

Теоретичні відомості

Dialogflow: загальні відомості. Dialogflow служба Google, яка працює на платформі Google Cloud. Dialogflow — це інтуїтивно зрозумілий і зручний інструмент, який включає продукти Google Cloud Speech-to-Text і досвід машинного навчання. В основному Dialogflow використовується для створення Google Assistant для більшості пристроїв.

Іншими словами, Dialogflow визначається як платформа NLP (обробка природної мови) , яка використовується для розробки діалогів для клієнтів компаній різними мовами на численних платформах. Використовуючи технології Google, розробники можуть створювати текстові та голосові інтерфейси. Наприклад, різні компанії використовують Dialogflow для створення ботів для обміну повідомленнями, які відповідають на запити клієнтів на платформах ***Google Assistant, Slack, Facebook Messenger, Alexa Voice Search (AVS)*** тощо**.**

Dialogflow, потребує передбачуваних і структурованих вхідних даних для належного виконання функцій, і іноді це робить використання інтерфейсів складним і неприродним. В ідеалі інтерфейси здатні визначити, що саме хоче

---

1 ***Дане завдання не є обов'язковим для виконання. Виконується за бажанням студента для накопичення бонусних балів.***

2 ***За необхідності встановити компілятор мови Python відповідно до інструкції наведеної в теоретичних відомостях.***

кінцевий користувач, на основі природної мови. Як приклад можна розглянути запит користувача: "*Яка погода сьогодні*"?, тоді як інші користувачі також можуть поставити запитання:

- *Яка зараз погода?*
- *Яка температура в Лондоні?*
- *Якою буде погода у жовтні?*

Як відомо розмовний діалог складно реалізувати за таких простих запитань. Для обробки та інтерпретації природної мови потрібен ***парсер*** надійної мови програмування. Саме Dialogflow надає такий тип синтаксичного аналізатора для реалізації якісних діалогів кінцевим користувачам.

Агент Dialogflow визначається як віртуальний агент, завданням якого є керування діалогами користувачів . ***Агент*** —модуль, який може зрозуміти складність природньої мови. Під час розмови Dialogflow перетворює текст та/або аудіо користувача на структуровані дані. Агент Dialogflow призначений для керування типами діалогів в інтелектуальних інформаційних системах.

Інтент-аналізи текстів. Інтент (правило) класифікує користувача для ведення ходу діалогу. Для кожного агента можна визначити різних користувачів, і об'єднаний агент здатний обробити весь діалог. У той час, коли користувач говорить або пише щось, тоді Dialogflow перевіряє та збігає заданий вираз користувача з правилом створеного агента. Порівняння правил прийнято називати ***класифікацією інтентів***.

Наприклад: для створення агента погоди, який має можливість ідентифікувати та відповідати на запити користувачів, визначається правило, в якому зберігаються запитання по прогнозу погоди. Коли користувач надає запит: "***Який прогноз***?" Dialogflow порівнює запит користувача з правилом прогнозу. Якщо необхідно уточнити інформацію за запитом користувача, таку як місцезнаходження, час для прогнозу погоди та інше, то, в такому випадку, необхідно створити інше правило. Надана інформація передається в систему Dialogflow для обробки та виконання пов'язаних запитів (рис. 23).

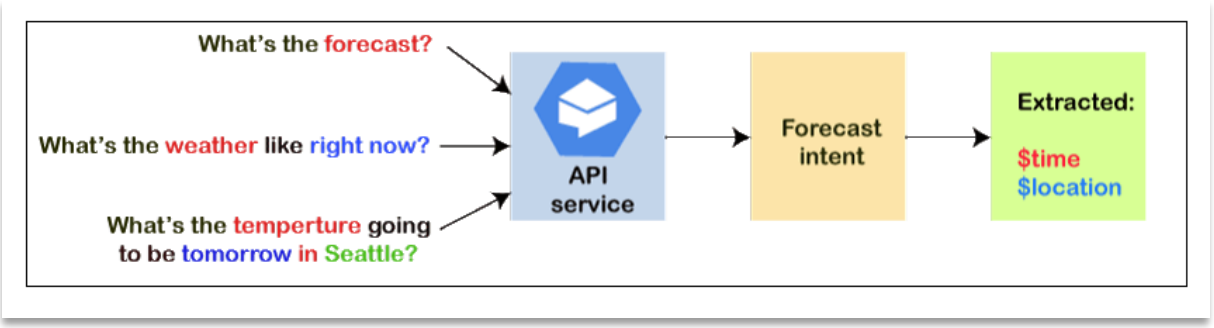


Рисунок 23. – Процес обробки даних в системі ***Dialogflow***

Кожен інтент містить наступні складові частини:

- Тренувальні фрази

- Дії
- Параметри
- Відповіді

Тренувальні фрази задаються, як приклади допустимих фраз від користувачів. Якщо одна з цих фраз нагадує запит користувача, тоді Dialogflow відповідає за певним правилом. Не потрібно передбачати всі можливі приклади запитів користувача, оскільки вбудоване машинне навчання Dialogflow розширюється іншими пов'язаними фразами у діалозі.

Дії можна використовувати для запуску різних подій, які попередньо визначені в системі. Для відповідного агента потрібно вказати певні дії, які, надалі, порівнюються з правилами в системі Dialogflow.

Параметри задаються в Dialogflow для виявлення значень запитів користувача. Кожен параметр містить тип (сутність), яка точно визначає спосіб отримання даних. Параметри не схожі на вихідні дані кінцевого користувача. Параметр означає структуровані дані, які використовуються для виконання логіки або створення відповідей.

Відповіді можна виділити мовні, візуальні або текстові для повернення базового потоку користувачу. Відповіді здатні передати відповідну інформацію користувачу, а також з'ясувати додаткову інформацію. На рис. 24 наведено базовий потік для порівняння інтентів та відповідей користувачеві.

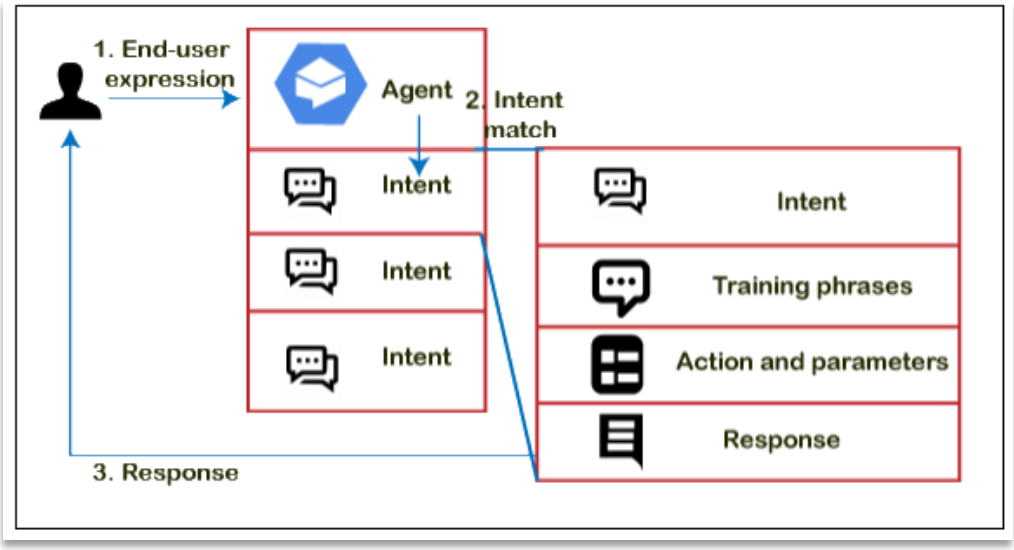


Рисунок 24. – Передача базового потоку в ***Dialogflow***

Сутності. Існує тип кожного параметра інтенту, відомий як тип сутності. Dialogflow пропонує різні попередньо визначені системні сутності (змінні), які можуть відповідати різним типам загальних даних. Наприклад, існують різні типи системних об'єктів для порівняння адреси електронної пошти, сервіс, час, дата тощо. Для порівняння даних можна створювати власні об'єкти

та їх властивості. Наприклад: можна визначити властивість фруктів, яка відповідає видам фруктів продуктового магазину, тощо.

Контекст. Для керування потоком діалогів використовуються контексти. Контекст Dialogflow аналогічний природній мові. Коли користувач скаже «*вони блакитні*», такий контент повинен бути зрозумілим з різним трактуванням в діалозі. Встановлюючи контексти введення та виведення, які розпізнаються за назвами рядкових структур, можна налаштувати контексти в правилах. Якщо правило збігається з запитом, то для цього правила активується та налаштовується вихідний контекст. Коли контексти знаходяться в активному стані, Dialogflow намагається порівняти інтент, налаштований для вхідних контекстів, який відповідає контекстам, які активні в потоці. На рис. 25 наведено приклад потоку, який використовує контекст взаємодії банківський агент, а саме:

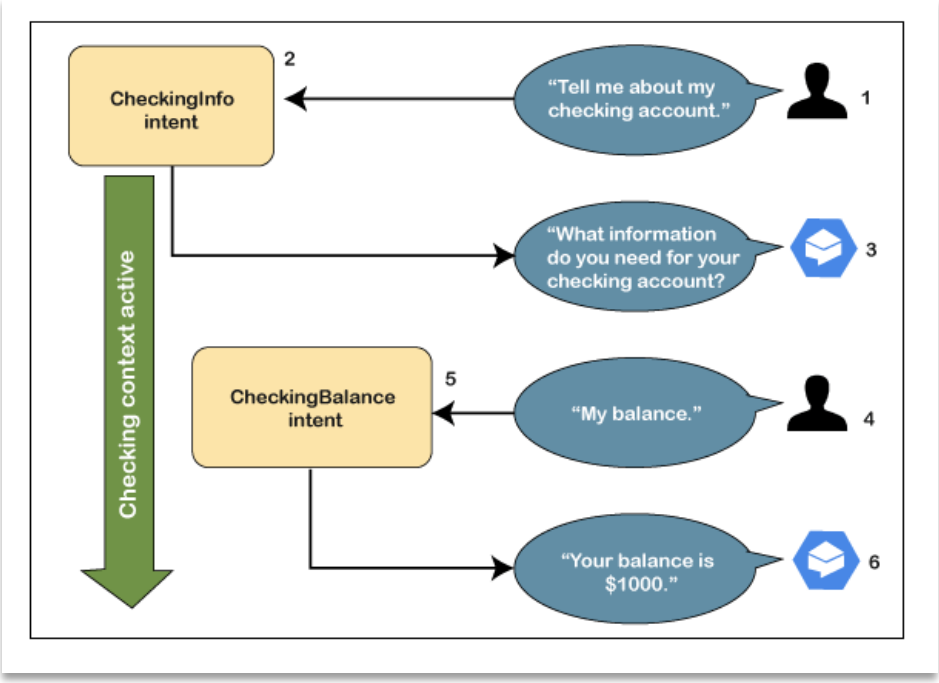


Рисунок 25. – Підключення телеграм-токена в ***Dialogflow***

- Користувач надає запит про дані поточного рахунку.
- Dialogflow порівнює запит користувача з правилом. Проводиться перевірка вихідного контексту даного інтенту та його активація за даним посиланням.
- Агент уточнює у користувача додаткову інформацію, пов'язану з поточним рахунком.
- Користувач надає відповідь по рахунку з назвою «*Мій баланс*».
- Даний запит користувача відповідає правилу перевірки балансу. Існує перевірка контексту даного інтенту, який активується при перевірці. Аналогічно проходить активація іншого правила ***SavingsBalance*** , яке відповідає на запит користувача, якщо контекст заощаджень активний.

- В результаті агент відповість на запит клієнта по перевірці балансу рахунку.

Створення чат-ботів на платформі Dialogflow. Щоб створити агента Dialogflow, потрібно виконати наступні кроки:

1. Вхід в Dialogflow за посиланням https://dialogflow.com/ та пройти безкоштовну реєстрацію на базі облікового запису Google (рис. 26).

Рисунок 26. – Процес реєстрації на платформі ***Dialogflow***

2. Створити агента (рис. 27) та задати відповідні параметри: мову, часовий пояс за замовчуванням, ім'я чат-бота.

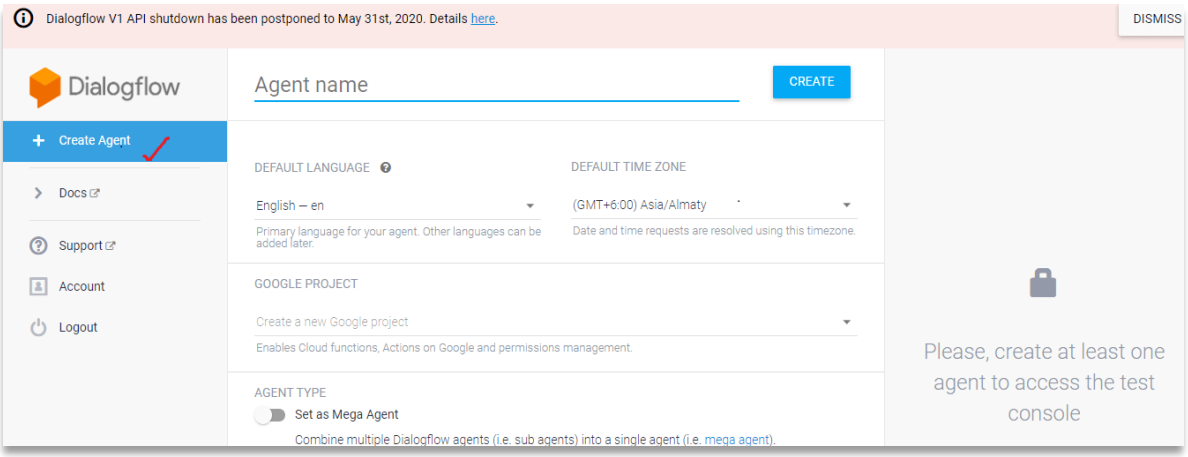


Рисунок 27. – Створення агента в ***Dialogflow***

3. Відредагувати стандартне правило привітання ***Default Welcome Intent*** (рис. 28). Після створення, бот не знає, як реагувати на запити користувача. Шлях до його навчання задається в даному інтенті. Необхідно змоделювати особистість бота, забезпечити відповіді привітання в діалозі користувач/бот.

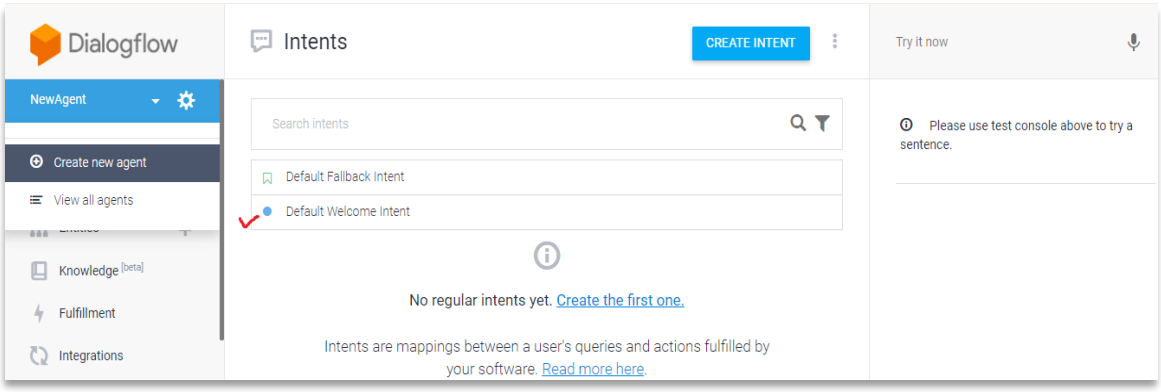


Рисунок 28. – Підключення телеграм-токена в ***Dialogflow***

4. Додати варіації фраз привітання у текстовій формі в секції тренувальних фраз (рис. 29)

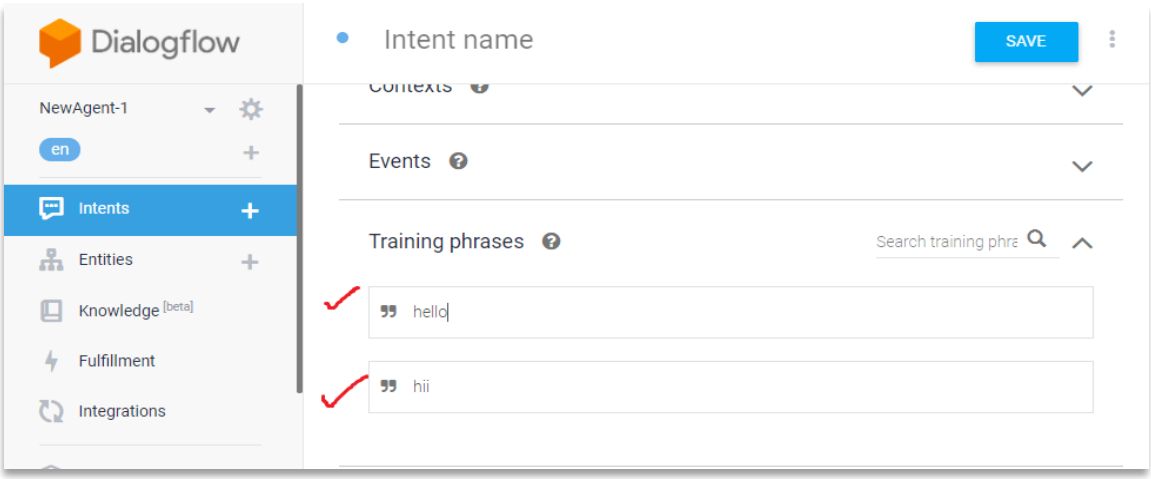


Рисунок 29. – Тренувальні фрази інтента в ***Dialogflow***

5. В опції «***Відповіді***» (рис. 30) ***Додати Відповіді*** (рис. 31) в діалозі привітання бот/користувач, які повинні мати текстовий формат відповідей.

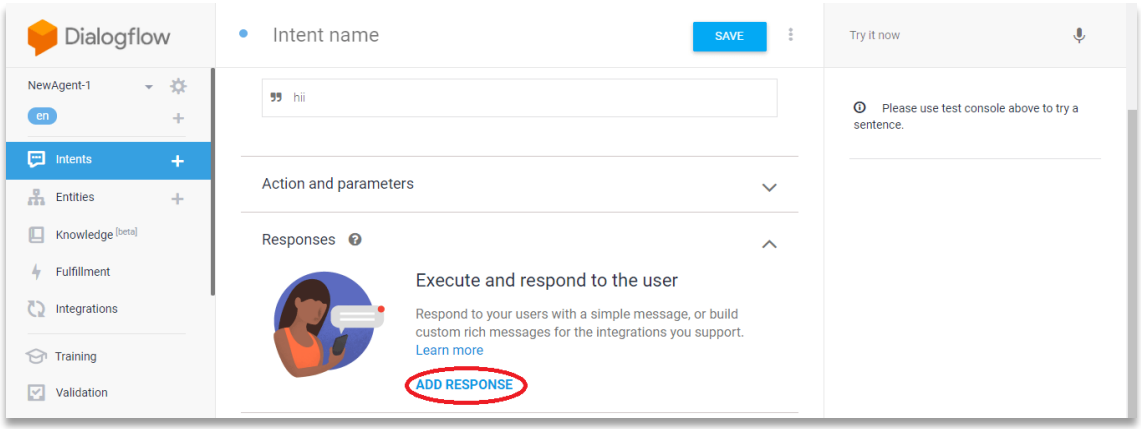


Рисунок 30. – Додавання відповідей в інтентах

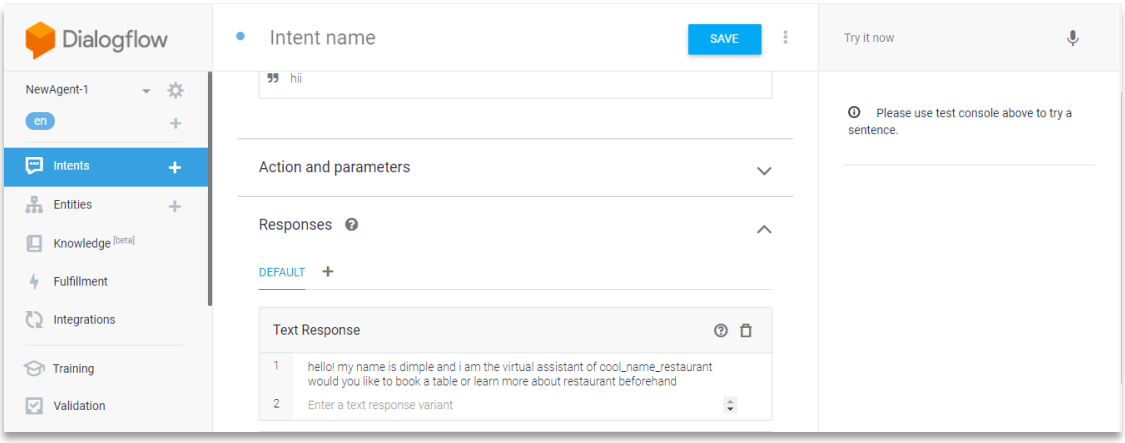


Рисунок 31. – Можливі варіанти відповідей в інтентах

6. Обов'язковим є збереження внесених змін в створених інтентах (рис. 32) кнопкою «***Зберегти***».

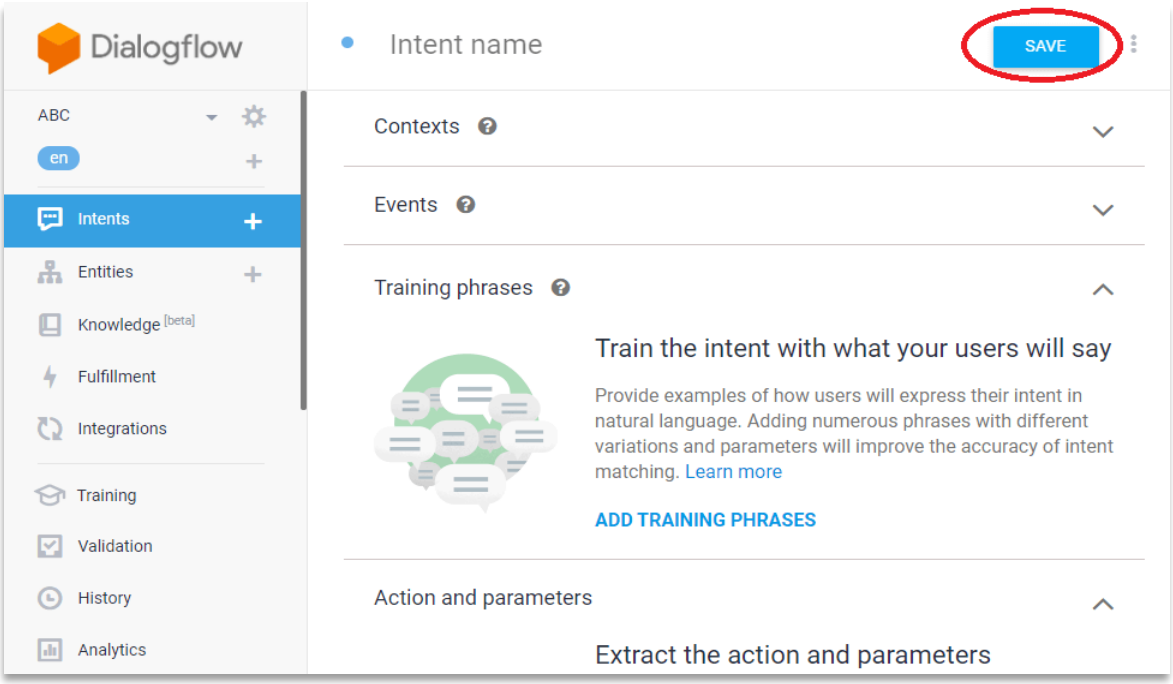


Рисунок 32. – Збереження інтентів в ***Dialogflow***

База знань в Dialogflow. В Dialogflow реалізована бази знань, в якій містяться/зберігаються діалоги користувачів. Деякі функції Dialogflow використовують концепцію баз знань під час пошуку відповідей на запити користувачів. Створити базу знань можна за наступних кроків:

1. Перейти до консолі діалогів.
2. Обрати віртуального агента.
3. Активувати опцію «**Знання**» (рис. 33) в лівій частині панелі меню.
4. Створити базу знань відповідною кнопкою (рис. 33).

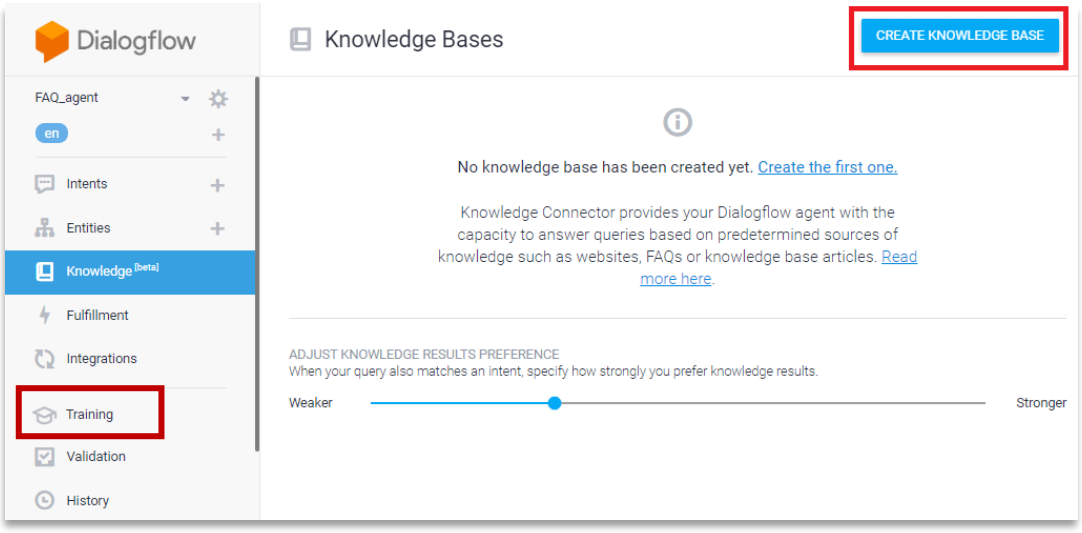


Рисунок 33. – Створення бази знань в ***Dialogflow***

5. Задати ім'я створеної бази знань та зберегти її (рис. 34)

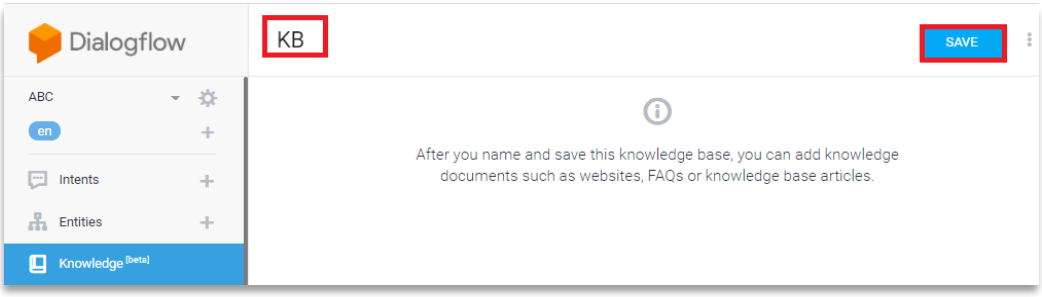


Рисунок 34. – Збереження бази знань в ***Dialogflow***

Навчання в Dialogflow. Навчання агента за методами машинного навчання в Dialogflow використовує переважно навчальні дані. Якщо потрібно задати напрям навчальних даних, то пропонується набір стандартних фраз, за допомогою яких можна створювати діалоги в правилах. В Dialogflow реалізовано навчальний інструмент, який використовується для покращення, експорту / імпорту фактичних даних діалогів, а також для аналізу навчальних даних.

В Dialogflow щоразу, після збереження агента, навчання виконується автоматично. Перш ніж тестувати агента, необхідно дочекатись завершення процесу навчання. Якщо агент містить більш 780 інтентів або опція автоматичного навчання вимкнена, то процес навчання можна виконати вручну.

Існують різні способи процесу навчання. Розглянемо один із них:

1. На панелі консолі ***Dialogflow*** необхідно активувати агента та обрати кнопку налаштувань (рис. 37).

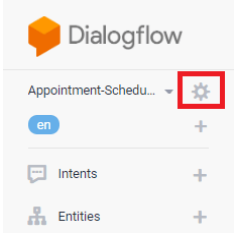


Рисунок 37. – Активація параметрів віртуального агента в ***Dialogflow***

2. На вкладці «***ML Settings***» (рис. 38) активувати кнопку «***Train***» (рис. 39)

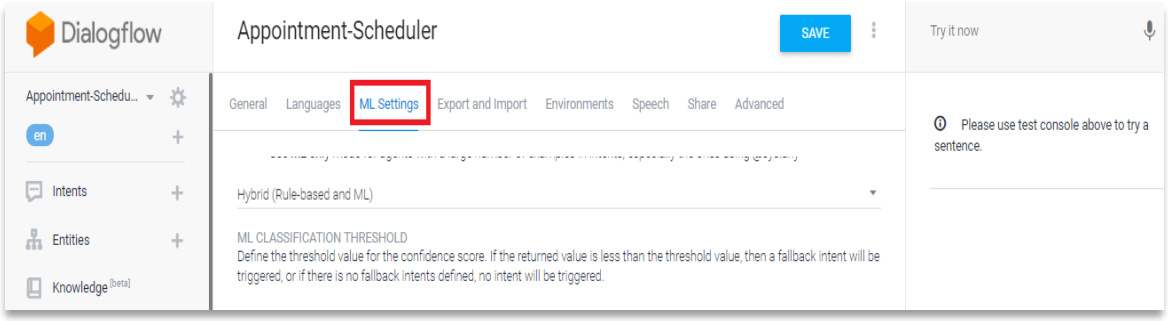


Рисунок 38. – Параметри процесу навчання в ***Dialogflow***

Якщо необхідно провести процес навчання за допомогою API, то потрібно викликати метод ***Train*** для відповідного типу агента.

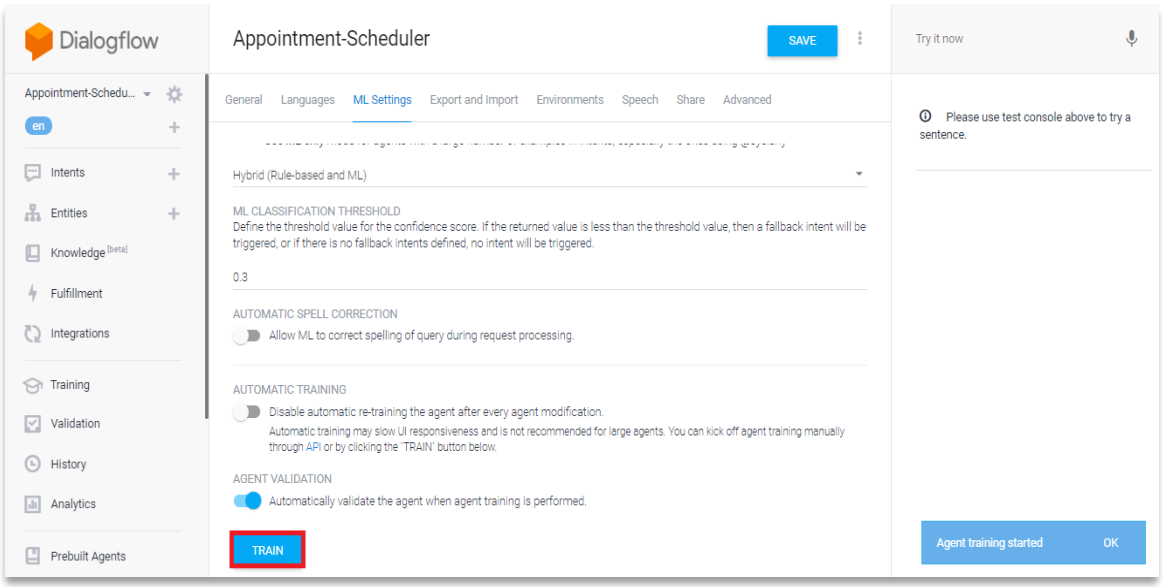


Рисунок 39. – Активація навчання віртуального агента

Інструмент навчання. За допомогою інструменту навчання можна покращити діалоги. Він використовується для перегляду розмов із користувачами, які проводились агентом. За допомогою навчального інструменту можна:

- імпортувати дані діалогів із реальних розмов, які запланували або записались;
- проаналізувати реальні розмови та інтенти, які були перевіреними в ході кожного діалогу;
- приєднати запити користувача з діалогу до навчальних фраз попередніх правил, резервних правил або навчальних правил.

Для активації інструменту навчання, необхідно ввімкнути журнал діалогів, оскільки інструмент навчання використовує історію даних агента. Інструмент навчання відображає тільки запити користувача. Для відкриття навчального інструменту необхідно активувати консоль Dialogflow, вибрати віртуального агента та включити опцію «Навчання».

Більш детально ознайомитись з платформою ***Dialogflow*** та процесами проектування інтелектуальний чат-ботів можна на офіційному сайті [7].

Інструкція по встановленню та налаштуванню мови програмування Python на прикладі версії 3.7.2:

1. Завантажити **Python** 3.7.2 (версія може бути змінена) з веб-ресурсу ***https://www.python.org/downloads/***
2. Відкрити завантажений пакет та виконати в командному рядку операційної системи наступні команди:
   - ***git clone https://github.com/eternnoir/pyTelegramBotAPI.git***
   - ***cd pyTelegramBotAPI/***
   - ***python setup.py install***
3. Рекомендується завантажити та встановити текстовий редактор **Sublime Text**, розміщеного на веб-ресурсі **https://www.sublimetext.com**
4. По завершенню, відкрити **Sublime Text** та створити робочий файл командою **File - New File** (або клавіші швидкого доступу: **Cmd+N**).
5. Написати програмний код в **Sublime Text** та запустити його на виконання командою **Tool - Build** (або клавіші швидкого доступу: **Cmd+B**).
6. Реалізований чат-бот інтегрувати в телеграм-каналі, додавши його токен.

Контрольні запитання

1. Які стандартні правила передбачені в структурі агентів Dialogflow?
2. Як налаштовати відповідне правило Dialogflow?
3. Назвіть основні призначення сутностей Dialogflow?
4. Як проходить навчання агента в Dialogflow?
5. Як інтегрувати віртуального агента, створеного в Dialogflow?
6. Назвіть основні складові бази знань.
7. Які переваги надає навчальний інструмент в Dialogflow?
8. Наведіть приклади правил та запитів віртуального агента прогнозу погоди / курсу валют.

Практичне завдання 4
Створення бази фактів, бази знань та експертних систем мовою функціонально-логічного програмування Prolog.

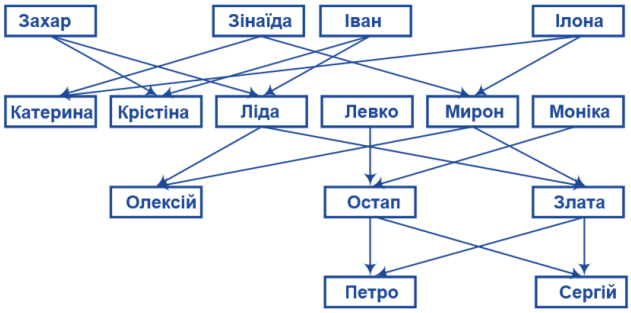
Мета. Вивчити структуру логічної експертної системи, засвоїти основи мови програмування ***Prolog*** і порядок роботи в online-середовищі ***SWI-Prolog***. Вивчити склад і основні принципи побудови експертних систем на засадах логічних моделей подання знань та навчитися будувати інтерфейс користувача експертної системи.

<u>Завдання 1.</u> Створити структуру логічної експертної системи мовою програмування Prolog:

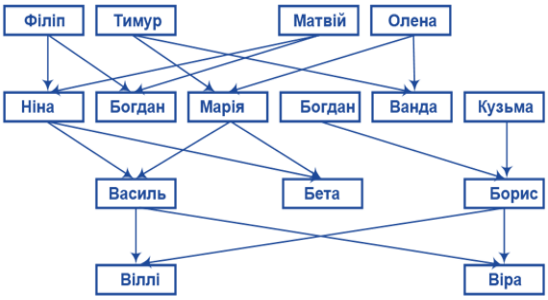
1. Згідно з варіантом, на мові ***Prolog*** створити базу фактів та відповідні правила бази знань за наведеною блок-схемою (***<u>табл. 6</u>***).
2. Навести приклади простих та складних 5-6 запитів до створеної бази знань.
3. Проаналізувати отримані результати і записати відповідні висновки.

Таблиця 6 – Індивідуальні варіанти до <u>завдання 1</u>.

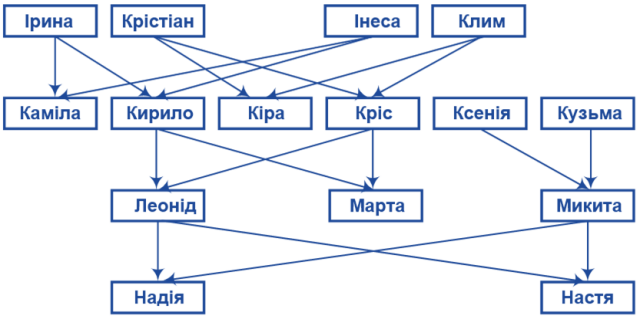
Варіант 1



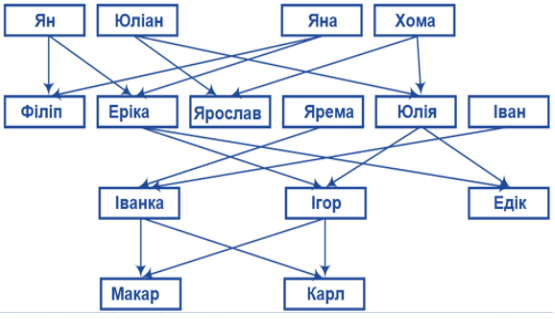
Варіант 2



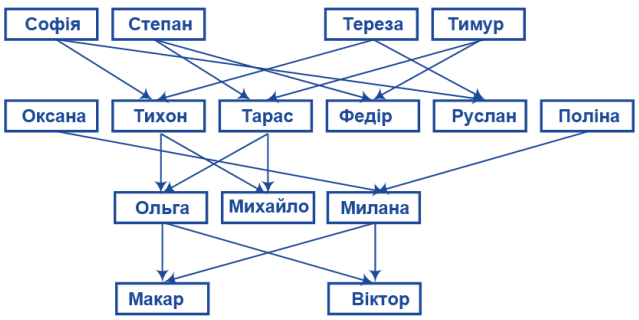
Варіант 3



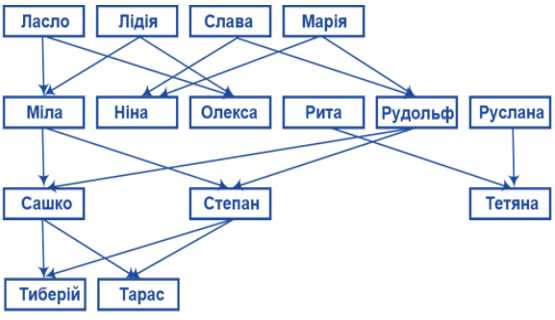
Варіант 4



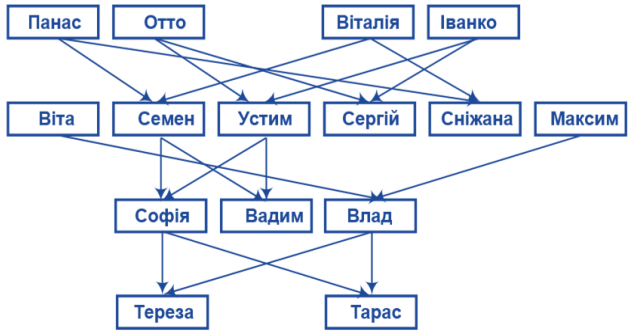
Варіант 5



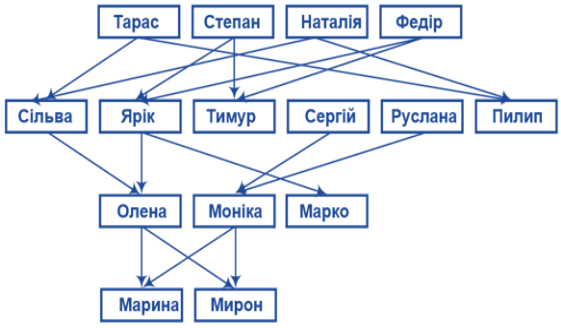
Варіант 6



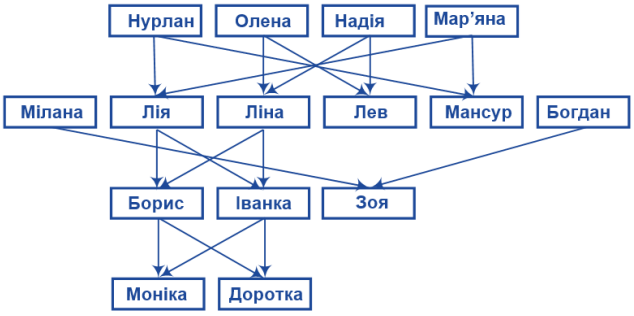
Варіант 7



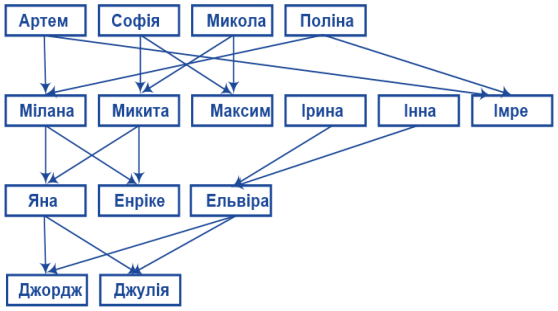
Варіант 8



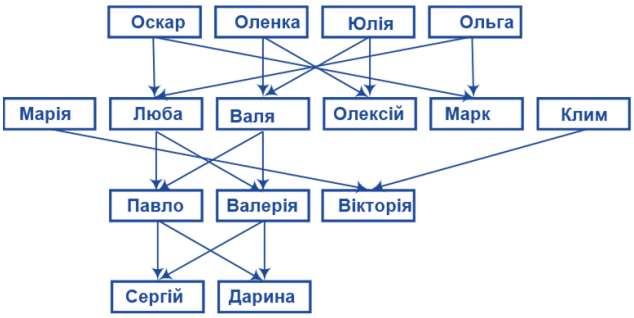
Варіант 9



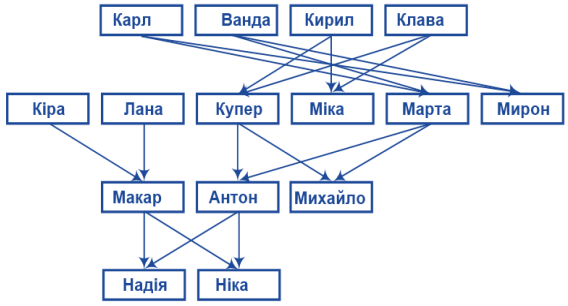
Варіант 10



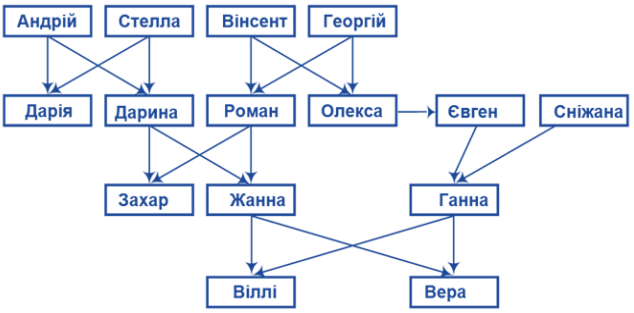
Варіант 11



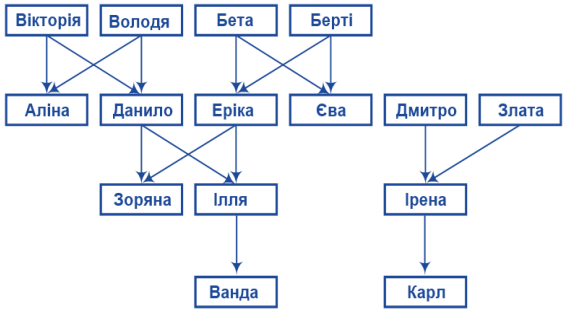
Варіант 12



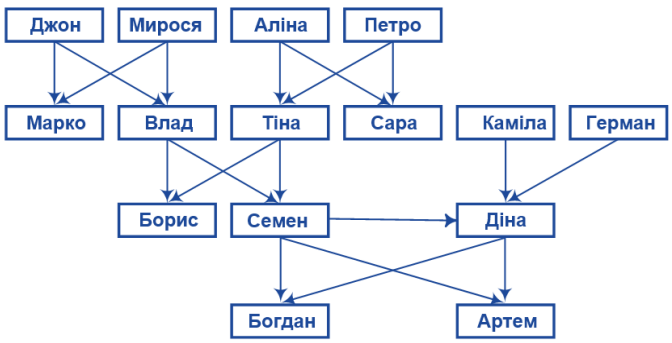
Варіант 13



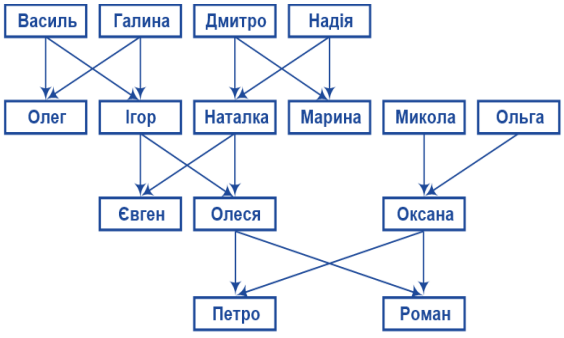
Варіант 14



Варіант 15



Варіант 16



Завдання 2. **Створити структуру логічної експертної системи мовою програмування Prolog:**

4. Згідно наведеного варіанту створити базу фактів і 1, 2 правила бази знань обраної предметної області (***табл. 7***).
5. Зв'язки предметної області задати у виді блок-схеми.
6. Створити ***SWI-Prolog*** програму.
7. Навести приклади 5-6 типів запитів до створеної бази знань.
8. Проаналізувати отримані результати і записати відповідні висновки.

Таблиця 7 – Перелік рекомендованих предметних областей для створення логічної експертної системи.

№ з/п	*Предметна область*
1.	Адміністративна структура університету
2.	Адміністративна структура інтернет-магазину
3.	Адміністративна структура підприємства
4.	Адміністративна структура навчального закладу
5.	Адміністративна структура України
6.	Асортимент товарів відеотехніки
7.	Асортимент товарів мобільних пристроїв
8.	Асортимент товарів оргтехніки
9.	Генеалогічне дерево київських князів
10.	Ієрархія військових чинів національної гвардії
11.	Ієрархія доменних імен DNS (Domain Name System)
12.	Ієрархія контенту веб-ресурсу
13.	Класифікація медичних препаратів
14.	Класифікація музичних інструментів
15.	Класифікація операційних систем
16.	Класифікація спортивних ігор
17.	Клуб за інтересами (прихильники письменників)
18.	Країни: столиця, населення
19.	Навчальний процес: викладач, асистент, лекція, практика.
20.	Навчальний процес: дисципліна, викладач, група, студент
21.	Склад медичного персоналу лікарні
22.	Склад спортивної команди Динамо-Київ
23.	Структура адміністративного правління компанії
24.	Структура адміністративного правління корпорації
25.	Структура адміністрації туристичного агентства
26.	Структура дистанційного курсу
27.	Структура спортивної команди по баскетболу

№ з/п	*Предметна область*
28.	Структура спортивної команди по волейболу
29.	Структура спортивної команди по гандболу
30.	Членство книжкового клубу

Завдання 3. Створити структуру експертної системи мовою програмування Prolog

1. В якості предметної області обрати предметну область згідно з варіантом (***табл. 8***), або з варіантом **завдання 2**.
2. Визначити мету, задачі експертної системи та питання користувача до експертної системи.
3. Визначити правила бази знань.
4. Побудувати блок-схему (дерево) логічних можливостей для вибору.
5. Побудувати схему структуру діалогу з користувачем.
6. Створити базу знань за допомогою засобів SWI-Prolog.
7. Проаналізувати отримані результати і зробити висновки.

Таблиця 8 – Перелік рекомендованих предметних областей для створення експертної системи.

№з/п	*Предметна область*
1.	Діагностика несправності автомобіля
2.	Індивідуальний підбір автомобіля
3.	Індивідуальний підбір взуття
4.	Індивідуальний підбір відеокамери
5.	Індивідуальний підбір житла
6.	Індивідуальний підбір комп’ютера
7.	Індивідуальний підбір косметики
8.	Індивідуальний підбір мобільного телефону
9.	Індивідуальний підбір музичного центру
10.	Індивідуальний підбір одягу
11.	Індивідуальний підбір спеціальності для абітурієнтів
12.	Індивідуальний підбір телевізора
13.	Класифікація дерев
14.	Класифікація квітів
15.	Класифікація класичного машинного навчання
16.	Класифікація комах
17.	Класифікація мавп
18.	Класифікація машинного навчання
19.	Класифікація мов програмування

№з/п	Предметна область
20.	Класифікація овочів
21.	Класифікація операційних систем
22.	Класифікація птахів
23.	Класифікація риб
24.	Класифікація собак
25.	Класифікація фруктів
26.	Класифікація спортивного інвентаря
27.	Персональний підбір автомобіля
28.	Персональний підбір ділового одягу
29.	Персональний підбір меблів
30.	Персональний підбір спортивного одягу

Зміст звіту

1. Титульна сторінка.
2. Тема та мета практичного заняття.
3. Наведені варіанти завдань.
4. Опис виконаних завдань з наведеними блок-схемами, рисунками, тощо.
5. Тексти ***SWI-Prolog*** програм
6. Висновки щодо виконаного практичного завдання.

Теоретичні відомості

Програмний онлайн-сервіс SWI-Prolog

SWI-Prolog є вільно розповсюджуваною (Free Software) реалізацією (діалектом) мови програмування Prolog, яка практично повністю відповідає стандарту ISO/ EC13211.

SWI-Prolog дозволяє розробляти програми будь-якої спрямованості, включаючи Web-додатки і паралельні обчислення, але основним напрямом використання є розробка експертних систем, програм обробки природної мови, навчальних програм, інтелектуальних ігор тощо.

Програмування в SWI- Prolog можливо в різних варіантах:

- за допомогою стандартного offline-середовища у вільному доступі, можна завантажити з (***http://www.swi-prolog.org*** опція **Download SWI-Prolog)**
- за допомогою online-середовища на сайті ***http://www.swi-prolog.org*** кнопка **Try SWI-Prolog online**

Для переходу в режим редагування і виконання програм необхідно натиснути на кнопку «***Program***» (рис. 40). Після цього можна створювати і редагувати Prolog-програму, створювати запити і отримувати відповіді (рис. 41).

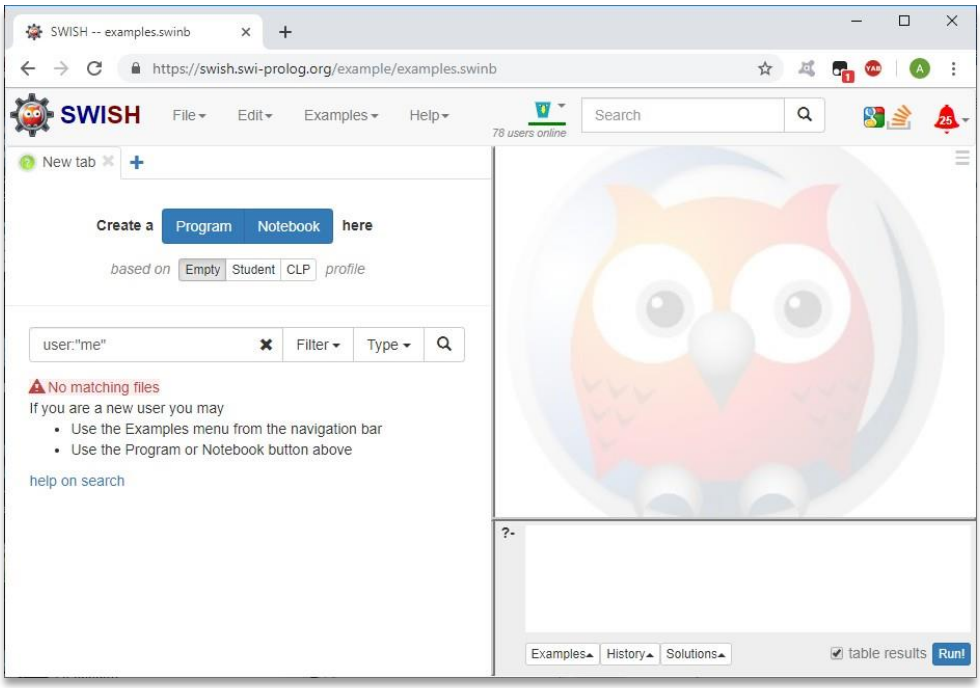


Рисунок 40. – Стандартне online-середовище програмування SWI-Prolog

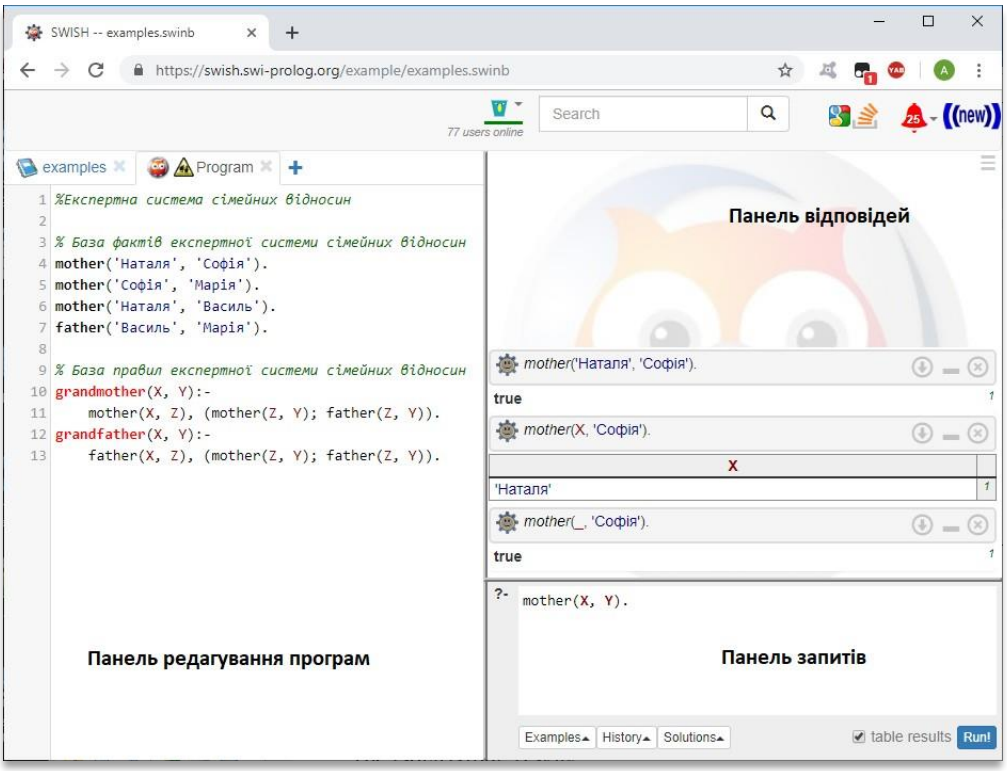


Рисунок 41. – Режим редагування і виконання програм SWI-Prolog

У лівій панелі здійснюється редагування програми, яка містить факти і правила. У правій нижній панелі виконується набір питань і запуск їх на виконанняза допомогою кнопки «**Run**!».

У правій верхній панелі інтерпретатор SWI-Prolog видає відповіді на запити. У разі якщо на питання може бути отримано більше однієї відповіді, за допомогою кнопок «Next», «10», «100» і «1,000» можна вивести на панель інші відповіді (рис. 42).

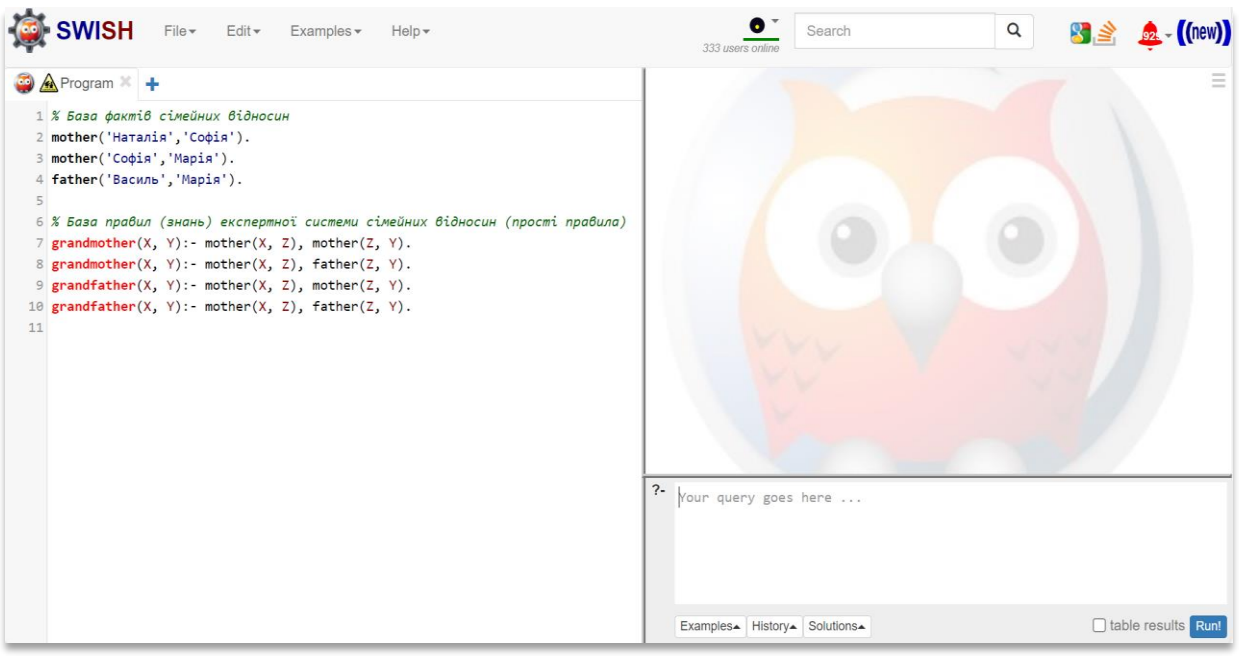


Рисунок 42. – Застосування кнопок панелі інструментів «Next», «10», «100» і «1,000»

Основні поняття мови Prolog

Програма на Prolog складається з речень (тверджень). Кожне речення закінчується крапкою. В загальному випадку речення складається із заголовкаі тіла (предиката). Речення бувають трьох видів: факти, правила, питання.

Факт – це речення, яке фіксує (визначає) певне відношення між об'єктами. Об'єкти називаються термами. Наприклад, факт, що Наталя є мамою Софії, може бути записаний таким чином:

mother ('Наталія', 'Софія').

Факт має тільки тіло, яке позначається ім'ям **mother** і два терми **'Наталія'** і **'Софія'**, зв'язані логічною операціє **I**, яка позначена символом « **,** ». Терми і зв'язок між ними подаються у круглих дужках. Факт може складатись і з одного терму. Факт є безумовно істинним твердженням. Термами факту можуть бути константа, змінна або складений об'єкт (список або функція). Як правило, об'єкт, що є причиною іншого об'єкта ставиться першим, але це не обов'язково. Важливо дотримуватись вибраної послідовності термів факту на протязі всієї програми. Факт є безумовно істинним твердженням.

Правило – речення, яке може бути істинним або не істинним, складаєтьсяз заголовка і тіла. Істинність залежить від істинності однієї або декількох формул, зазначених в тілі. Зазвичай правило містить кілька цілей, які повинні бути істинними для того, щоб саме правило було істинним. Правило має ім'я, після якого йдуть символи « **:-** » – ознака правила.

B:- A1, ..., An.

B називається заголовком або головою речення, а **A1, ..., An** – тілом. У цьому реченні ключовою логічною операцією є перевернута імплікація (**B:- A** еквівалентно **B ← A**, «B випливає з A»). Символ "**:-**" означає "випливає з», символ «,» – кон'юнкція (логічне І, ∧).

При необхідності застосування диз'юнкції (логічне АБО, ∨) використовується символ «;», що діє до наступної диз'юнкції, закінчення правила або закриваючої круглої дужки, наприклад**, D:- A, (B; C)**. В логікі предикатів це виглядає так: **A ∧ (B ∨ C) → D.**

Наприклад, відомо, що бабуся особи – це мама його мами або мама його тата. Відповідні правила мають наступний вигляд:

grandmother (X, Y):- mother (X, Z), mother (Z, Y).

grandmother (X, Y): - mother (X, Z), father (Z, Y).

або

grandmother (X, Y):- mother (X, Z), (mother (Z, Y); father (Z, Y)).

Питання (запит, мета) – речення, що складається тільки з тіла. Запити позначаються символами **«?-»**.

Запити використовують для з'ясування здійсненності деяких відносин між описаними в програмі об'єктами. Автоматична система логічного виведення Prolog розглядає питання як мету, яку треба довести. Відповідь на питання може виявитися позитивною (true) або негативною (false), в залежності від того, чи може бути досягнута мета.

Програма може містити питання в тілі (внутрішня мета). Якщо внутрішньої мети в програмі немає, то після запуску програми система видає запрошення **«?-»** вводити питання в діалоговому режимі (зовнішня мета).Якщо мета досягнута, система відповідає «yes» («true»), в іншому випадку «no» («false»). Слід зазначити, що відповідь «no» на питання не завжди означає, що вона негативна. Система може дати таку відповідь і в тому випадку, коли у неї просто недостатньо інформації, щоб позитивно відповісти на питання. Тобто Prolog заснований на так званій «Моделі закритого світу», в якій все, що можна отримати на основі опису моделі є істиною, а решта – хибність.

Заголовок в Prolog повинен бути складатись з літер латинського алфавітуі починатися з малої літери. Ім'я терму повинне починатись з малої літери латинського алфавіту, або поміщатись в одинарні лапки, наприклад, **'Наталія'**, тоді алфавіт може будь який.

Змінні. Ім'я змінної в Prolog може складатися з літер латинського алфавіту, цифр, знаків підкреслення і повинно починатися з великої літери або символу підкреслення. Наприклад, у реченні:

grandmother (X, Y):- mother (X,Z), mother (Z, Y)

Змінні мають імена з однієї літери – **X, Y, Z**. При цьому змінних тілі правила еквівалентні об'єктам предметної області. Змінні можуть бути вільними або зв'язаними.

Вільна змінна – змінна, яка ще не отримала значення. Вона не дорівнює ні нулю, ні пробілу; у неї взагалі немає ніякого значення. Такі змінні ще називають неконкретизованими.

Змінна, яка отримала якесь значення, називається зв'язаною. Такій змінній не може бути присвоєно нове значення. Областю дії змінної в Prolog є одне речення. У різних реченнях може використовуватися одне й теж ім'я змінної для позначення різних об'єктів. Винятком з правила визначення області дії є анонімна змінна, яка позначаєтьсясимволом підкреслення «_».

Анонімна змінна наказує інтерпретатору (компілятору) проігнорувати значення терма. Анонімні змінні можуть записуватися тільки в якості терма предиката. Використовувати їх у виразах (наприклад, арифметичних) не можна.

Таким чином, програма на Prolog складається з фактів і правил, що виражають деякі знання про предметну область. Питання – це предикат, істинність якого нас цікавить. Якщо питання не містить змінних, то обчислення його значення дає відповідь «true» при його істинності, або відповідь «false» при його хибності. Якщо в питанні є змінні, то відшукуються їхні значення, при яких цей предикат і всі предикати програми стають істинними. В цьому і полягає суть логічного виведення програми на Prolog.

Приклади бази фактів і бази знань в Prolog

Розглянемо приклад бази фактів і бази знань сімейних відносин. Зауважимо, що символ % означає, що після нього йде коментар, який не впливає на програму.

Спосіб 1.

```
% База фактів сімейних відносин
mother('Наталія','Софія').
mother('Софія','Марія').
father('Василь','Марія').
% База правил (знань) експертної системи сімейних відносин (прості правила)
grandmother(X, Y):- mother(X, Z), mother(Z, Y).
grandmother(X, Y):- mother(X, Z), father(Z, Y).
grandfather(X, Y):- father(X, Z), father(Z, Y).
grandfather(X, Y):- father(X, Z), mother(Z, Y).
```

Спосіб 2.

```
% База фактів сімейних відносин
```

```
mother('Наталія','Софія').
mother('Софія','Марія').
father('Василь','Марія').
% База правил (знань) експертної системи сімейних відносин (складні правила)
grandmother(X, Y):-
mother(X, Z),(mother(Z, Y);father(Z, Y)).
grandfather(X, Y):-
father(X, Z), (mother(Z, Y); father(Z, Y)).
```

Розглянемо, як питання природною мовою записуються в Prolog.

Питання 1. Чи є Наталія матір'ю Софії?

```
?-mother ('Наталія', 'Софія').
    % Відповідь системи. true.
```

Відповідь означає, що такий факт в базі фактів існує (є істинним), тобто Наталя є матір'ю Софії.

Питання 2. Хто є матір'ю Софії?

Для таких питань необхідно використовувати змінні, наприклад, змінну з ім'ям X. Змінна X отримує значення першого терму предикатів-фактів **mother**, в яких другим термом є терм **'Софія'**

```
?-mother(X, 'Софія').
    % Відповідь системи.
X = 'Наталія';
```

Перший рядок повідомлення означає, що відповідь знайдена і матір'ю Софії є Наталія. Другий – що в базі знань для решти речень не виявлені інші матері Софії.

Питання 3 – Чи є у Софії матір? Використовується анонімна змінна.

```
?-mother(_, 'Софія').
    % Відповідь системи. true.
```

Питання 4 – Знайти всіх матерів та їх дітей.

```
?-mother(X, Y).
    % Відповідь системи.
    X = 'Наталія',
    Y = 'Софія';
    X = 'Софія',
    Y = 'Марія';
```

В даному випадку після третьої відповіді не видається **false**, тому що в базі знань були перебрані всі речення і вони всі істинні. Результат виконання запиту даного питання в ***SWI-Prolog*** наведено на рис. 42.

Питання 5 – Для кого Наталя є бабусею?

?-grandmother('Наталія', X).

% Відповідь системи.

X = 'Марія';

X = 'Марія'.

В даному випадку видається дві однакові відповіді «Марія», тому що правило grandmother в одному випадку спрацювало по ланцюжку «Наталя – Софія – Марія», а в іншому – «Наталія – Василь – Марія». Очевидно, що в наведеному прикладі бази знань бабусі **'Наталія'** – це дві різні жінки з однаковими іменами і онучки теж мають однакові імена **'Марія'**.

Процедура виведення в Prolog

При пошуку розв'язку (доказу мети) в Prolog використовується метод перебору з поверненнями (з пошуком в глибину). Prolog при доказі твердження по-чергово намагається встановити істинність предикатів (тверджень), що входять в нього.

Якщо перший предикат істинний, то Prolog переходить до другого. Якщо і він істинний, то переходить до третього. Якщо другий предикат хибний, то Prolog намагається встановити його істинність при іншихзначеннях змінних, що входять в нього. Якщо цього не вдається зробити, то вінповертається до першого предикату і намагається встановити його істинність для нових значень змінних, а потім знову повертається до доказу другого предиката.

Така процедура повторюється до тих пір, поки не буде досягнута істинність останнього предиката. Після доведення істинності останнього предиката мети Prolog завершує роботу. Процес повернення в Prolog називається **backtracking**.

Наприклад: визначимо питання існуючих дітей та виведемо їх імена, використовуючи наведений складний запит:

?- (mother(Y,Z); father(Y, Z)),

write(Z).

% Відповідь системи

Софія

Y = 'Наталія',

Z = 'Софія'

Марія

Y = 'Софія',

Z = 'Марія'

Марія

Y = 'Василь',
Z = 'Марія'

База даних на Пролозі - це сукупність фактів. В процесі роботи в базу даних можна додавати нові факти, видаляти або змінювати існуючі факти. За наведеною схемою (рис. 43). розглянемо приклад дерева родинних відношень, використовуючи предикати: **parent** з двома параметрами ім'я одного з батьків та ім'я дитини (рис. 43); **brother** з параметрами імен брата/сестри.

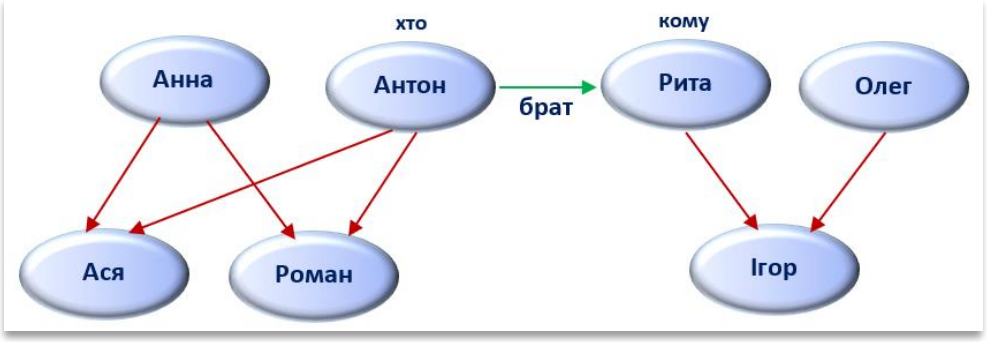


Рисунок 43. – Схема родинних зв’язків

За наведеною схемою складемо базу знань мовою Пролог:

```
% Приклад програми Пролог.
male(anton).
male(roma).
male(oleg).
male(igor).

female(anna).
female(asya).
female(rita).

parent(anna,asya).
parent(anton,asya).
parent(rita,igor).
parent(anton,roma).
parent(anna,roma).
parent(oleg,igor).

brother(anton,rita).

brothers(X,Y):- brother(X,Y), female(X).
brothers(X,Y):- brother(X,Y), male(X).

parents(X,Y):- parent(X,Y), female(X).
parents(X,Y):- parent(X,Y), male(X).
```

Написати на мові Prolog наступні запити та запустити їх на виконання:

- *які батьки мають дітей?;*
- *хто із дітей має батьків?;*
- *у кого із дітей є брат?;*
- *як звати сестру та вивести її ім'я.*

Подання знань в продукційних системах

Продукція визначається як четвірка:

$$< (i)\ Q;\ P;\ A \rightarrow B;\ N >,$$

де: ***(i)*** – ім'я продукції;

Q – предметна область (сфера застосування);

P – умова застосування продукції;

$A \vee B$ – ядро продукції (правило) інтерпретується як «**Якщо маєш А, зроби В**» або «**Якщо А то В**»;

N – після умова продукції (процедури, які необхідно виконати після реалізації ядра.

Найбільше застосування ця модель знайшла в продукційних (логічних) експертних системах (рис. 44). Для спрощення реалізації таких систем створена мова програмування Prolog. Її особливістю є те, що алгоритм логічного виведення вбудований в неї, що не вимагає писати програму логічного виведення, чим прискорює розробку і тестування систем.

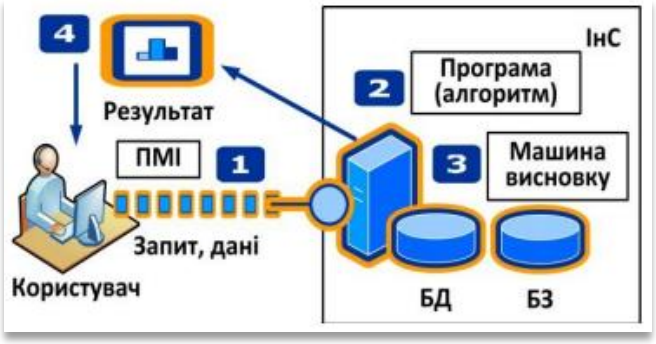


Рисунок 44. – Структура продукційної експертної системи

В математичних термінах Prolog-програма інтерпретується наступним чином:

- факти і правила – множина аксіом;
- питання користувача – теорема;
- механізм виведення намагається з аксіом логічно довести дану теорему (в Prolog дана теорема іменується метою).

Факти і правила використовуються для доведення мети. Prolog шукає факти і заголовки правил, зіставні з метою. Факт із бази фактів і факт з умови правила відповідають один одному, якщо виконуються наступні три умови: імена фактів однакові; факти мають рівну кількість термів; терми розташовані в однакових позиціях.

Етапи створення експертної системи.

Ідентифікація. Необхідно визначити мету і задачі експертної системи; визначити експертів і тип користувачів.

Концептуалізація. Проводиться змістовний аналіз предметної області; виділяються основні поняття і їх взаємозв'язки; визначаються методи рішення задач.

Формалізація. Вибираються програмні засоби розробки ЕС,визначаються способи представлення усіх видів знань, формалізуються основні поняття.

Виконання. Здійснюється наповнення бази знань. Знання"витягаються" з експерта, представляються у логічному або інших видах для реалізації програмно.

Тестування. Експерт і інженер по знанням з використанням діалоговихі пояснювальних засобів перевіряють компетентність ЕС. Процес тестування продовжується до визнання експертом необхідного рівня компетентності системи.

Дослідна експлуатація. Перевіряється придатність ЕС для кінцевих користувачів**.**

Режими роботи експертних систем

Експертні системи працюють у двох режимах: придбання знань; режим рішення задачі (режим консультації).

В режимі ***придбання знань*** інженер по знанням по результатамспілкування з експертом наповнює ЕС новими знаннями і фактами.

В режимі ***консультації*** користувач взаємодіє з ЕС для рішення своєї задачі шляхом діалогу. Якщо в системі не вистачає фактів для доведення мети, наприклад встановлення діагнозу, то вона задає питання користувачеві ітой ці факти надає. Кількість запитань залежить від внутрішньої структури діалогу конкретної ЕС. Після закінчення діалогу машина виведення доводить або спростовує мету.

Модель експертної системи для вибору туристичної путівки

Етап ідентифікації експертної системи.

Постановка завдання: у процесі вибору туру важко зупинитися на якійсь моделі, тому що велика кількість фірм надає подібні послуги з різними характеристиками, і в цьому достатньо важко орієнтуватися. Пропонується розробити прототип експертної системи (ЕС) прийняття рішень по вибору туру за перевагами, які є важливими для клієнта (користувача).

Призначення ЕС: консультування клієнта під час визначення оптимального туру за властивостями, яким він надає перевагу.

Сфера застосування прототипу ЕС: туристичні фірми, що займаються продажом турів.

Ціль: вибір оптимального варіанта туру для клієнта згідно з його потребі вимог до послуги.

Вхідні дані: країна відвідування, вид відпочинку, тривалість відпочинку.

Очікувані результати: конкретний тур.

Об'єкти (фактори) Предметної Області (ПрО). Тур – це послуга, на вибір якої впливають різні фактори. Виходячи з соціологічного опитування, найбільш важливими факторами вибору є наступні:

- країна відвідування;
- вид відпочинку;
- тривалість відпочинку.

Етап концептуалізації експертної системи.

Кількість об'єктів ПрО – 6. Дерево логічних можливостей вибору туру подано на рис. 45.

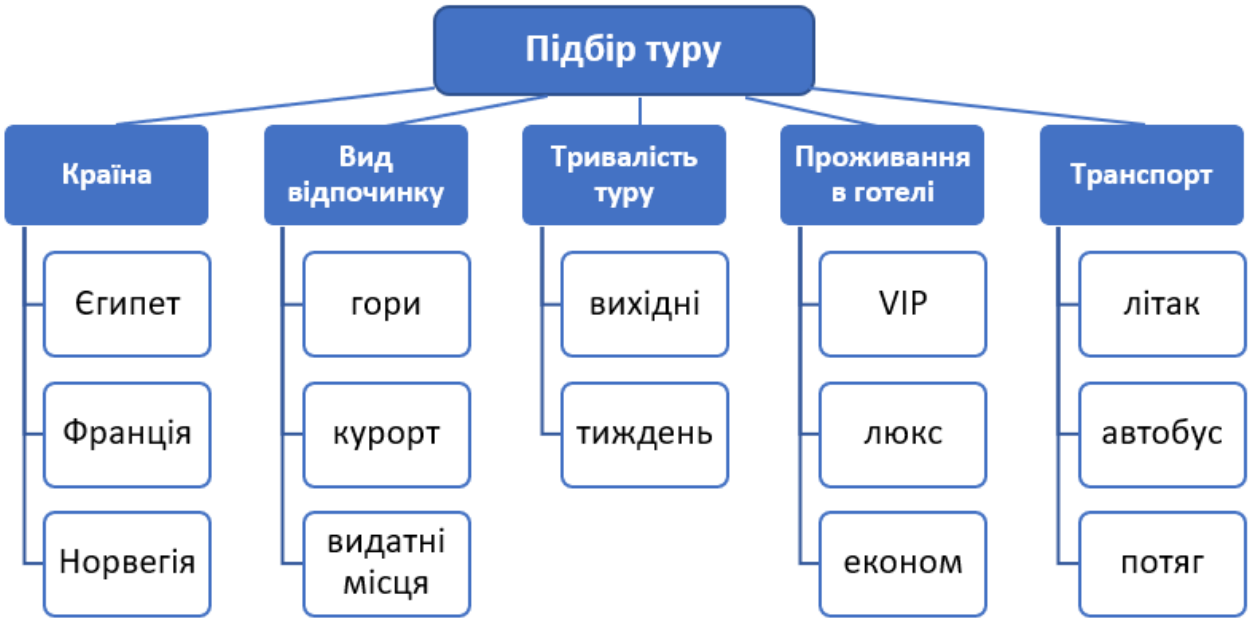


Рисунок 45. – Дерево логічних можливостей для вибору туру

Загальна кількість можливих варіантів турів дорівнює добутку всіх кількостей значень атрибутів (***3 x 3 x 2 x 3 x 3 = 162***). Але серед цих варіантів експерт відібрав декілька, які відповідають існуючим на даний час пропозиціям (табл. 9).

Наведемо приклад правила:

ЯКЩО

A Країна = egypt

B Вид Відпочинку = mountain

C Тривалість відпочинку = weekend

D Проживання в готелі = ekonom

E Транспорт = plane

ТОДІ

Тур = 'Квадро-тур на Синай'.

Таблиця 9. – Атрибути Бази знань

Атрибут (термін)	*Питання (опис)*	*Відповідь*
Країна	Яку країну Ви бажаєте відвідати?	egypt
		france
		norway
Тип відпочинку	Якому виду відпочинку Ви надаєте перевагу?	mountain
		resort
		sightseeing
Тривалістьвідпочинку	Як довго Ви хочете відпочивати?	weekend
		week
Проживання в готелі	В якому готелі ви бажаєте проживати?	vip
		luxury
		budget
Транспорт	Як плануєте доїхати до місця відпочинку?	plane
		train
		bus

Операції та вбудовані предикати SWI-Prolog

В табл. 10 наведені деякі операції і предикати SWI-Prolog, які вподальшому будуть використовуватись для реалізації прикладів.

Таблиця 10. – Деякі операції та предикати SWI-Prolog

Операція / Предикат	*Призначення*
true	істина
fail, false	хибність
=	Для змінної, що стоїть зліва від операції: ➢ вільної – присвоювання без обчислення виразу праворуч відоперації; ➢ зв’язаної – порівняння без обчислення виразу праворуч відоперації.
<, =<, >=, >	Арифметичні (тільки для чисел) операції порівняння
=:=	арифметична рівність
=\=	арифметична нерівність

Операція / Предикат	*Призначення*
is	Для змінної, що стоїть зліва від операції: ➢ вільної – присвоювання з обчисленням виразу праворуч; ➢ зв'язаної – порівняння з обчисленням виразу праворуч від is.
@<, @=<, @>=, @>	Операції порівняння для констант і змінних будь-якого типу(чисел, рядків, списків і т.д.)
==	Рівність констант і змінних будь-якого типу
\==	Нерівність констант і змінних будь-якого типу
not(A)	Заперечення логічного виразу A
read(A)	Читання значення з клавіатури і присвоєння його змінній A
write(A)	Друк A на екрані з установкою курсору після останнього надрукованого символу
writeln(A)	Друк A на екрані з установкою курсору в початок наступногорядка
nl	Установка курсора в початок наступного рядка
!	Предикат забороняє повернення до тієї точки, в якій була зупинка
assert(A), assertz(A)	Динамічне додавання факту (правила) в початок списку подібних фактів (правил) бази фактів і знань
asserta(A)	Динамічне додавання факту (правила) в кінець списку подібнихфактів (правил) бази фактів і знань
retract(A)	Видалення першого факту (правила) бази фактів і знань
retractall(A)	Видалення всіх фактів (правил) із бази фактів та знань з іменем A
dynamic(A/n)	Вказівка, що предикат A може змінюватись в процесі виконання програми за допомогою предикатів **assert і retract**, де n – кількість термів предикату A

Програмна реалізація експертної системи вибору туру

На початку діалогу в базі фактів факти відсутні. Під час діалогу користувач вводить факти і після цього робиться логічне виведення. Дляорганізації діалогу застосовуються вбудовані предикати **write(A)** – вивід на екран, **nl –** перехід на новий рядок на екрані, **read(A)** – читання з клавіатури, **assert(A)** – динамічне

додавання факту в базу фактів (дозвіл на зміну бази фактів дає предикат **:- dynamic**. Система задає три питання: **ask1, ask2, ask3**.

Приклад програмного коду:

```
main:-
    greeting,
    ask1, ask2, ask3, ask4, ask5,
    find_tour(Product),
    describe(Product),
    nl.

greeting:- write('Експертна система вибору туристичного маршруту'), nl, nl.
:- dynamic(country/1).
:- dynamic(type/1).
:- dynamic(duration/1).
:- dynamic(hotel/1).
:- dynamic(transport/1).

ask1:- write('Якому виду відпочинку ви надаєте перевагу?'), nl,
    write('mountain, resort, sightseeing'), nl,
    read(N), assertz(type(N)).
ask2:- write('В якій країні ви б хотіли відпочивати?'), nl,
    write('egypt, thailand, france'), nl,
    read(N), assertz(country(N)).
ask3:- write('Як довго ви плануєте відпочивати?'), nl,
    write('weekend, week'), nl,
    read(N), assertz(duration(N)).
ask4:- write('В якому готелі ви бажаєте проживати?'), nl,
    write('vip, luxury, budget'), nl,
    read(N), assertz(hotel(N)).
ask5:- write('Як плануєте доїхати до місця відпочинку?'), nl,
    write('plane, train, bus'), nl,
    read(N), assertz(transport(N)).

find_tour(Tour):-
    tour(Tour), !.

tour('Круіз Норвегії'):-
    country(norway),
    type(resort),
    duration(week),
    hotel(luxury),
```

```
    transport(plane).

tour('Екскурс по Парижу'):-
    country(france),
    type(sightseeing),
    duration(week),
    hotel(luxury),
    transport(plane).

tour('Горнолижний курорт Парк & Сьютс Престиж Межев - Ле Лож Бланш'):-
    country(france),
    type(mountain),
    duration(weekend),
    hotel(vip),
    transport(bus).
-------------------------
    describe(Tour):-
        write('Тур, який ви обрали '), nl,
        write(Tour), nl, nl.
```

Для запуску програми необхідно на панелі запитів SWI-Prolog ввести команду **main** і натиснути **Run!** Результати роботи експертної системи наведено на рис. 46.

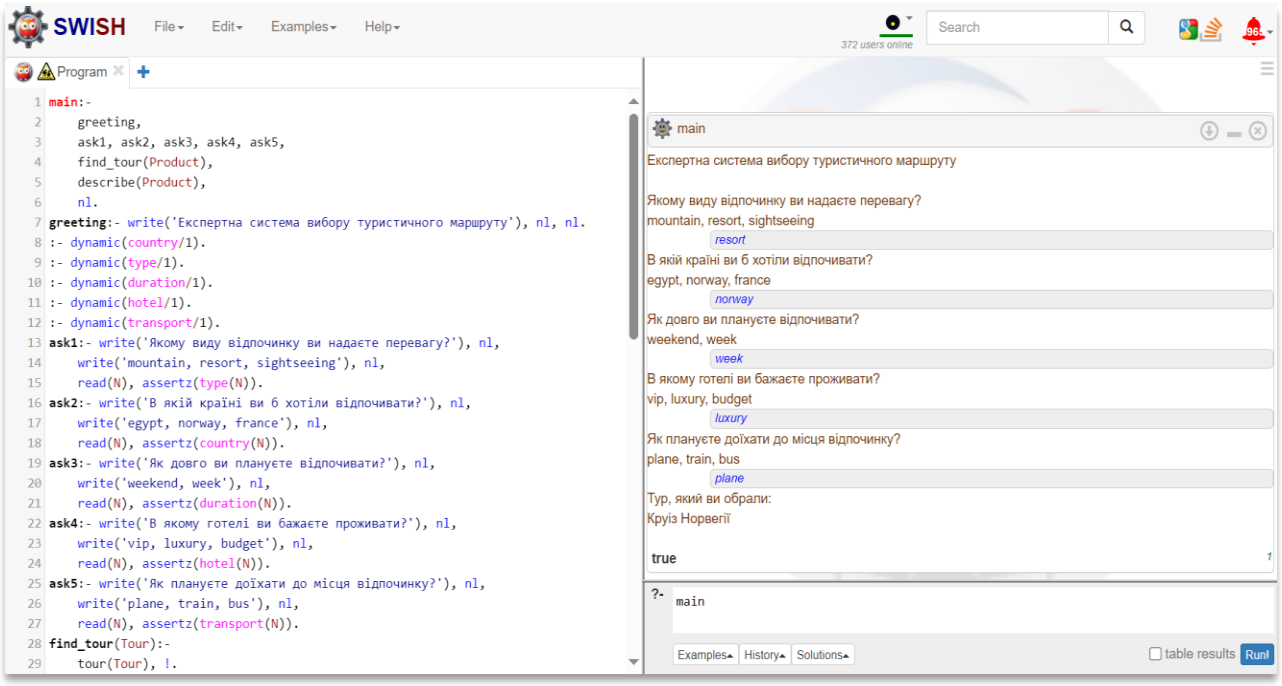


Рисунок 46. – Тестування створеної експертної системи Пролог.

Контрольні питання

1. Механізм логічного виводу в Prolog.
2. Типи предикатів в Prolog.
3. Типи запитань в Prolog.
4. Переваги і недоліки продукційних систем.
5. Формальна модель продукційної системи подання знань.
6. Структура статичної експертної системи.
7. Етапи створення експертної системи.
8. Режими роботи експертних систем.
9. Механізм логічного виводу в Prolog.

Практичне завдання 5
Дослідження персептрону та ондошарових нейронних мережі для моделювання логічних операцій.

Мета. вивчення структурних схем моделі нейрона та архітектури персептронної одношарової нейронної мережі; побудова та дослідження моделей простих нейронних мереж програмними засобами Python.

Завдання 1. Створити модель простої нейронної мережі, яка здатна розпізнавати логічну операцію кон'юнкції (логічне І). Перевірити працездатність створеної моделі нейронної мережі на тестовій вибірці.

Таблиця 11. – Істинність операції І (∧)

X_1	X_2	Y
0	0	0
0	1	0
1	0	0
1	1	1

1. Створити модель штучного нейрона, підібравши вагові коефіцієнти.
2. Створити модель штучної мережі для визначення таблиці істинності **логічної операції І**
3. Перевірити функціональність створеної модель нейронної мережі на тестовій вибірці

Завдання 2. Створити модель простої нейронної мережі, яка здатна розпізнавати логічну операцію диз'юнкції (логічне АБО). Перевірити працездатність створеної моделі нейронної мережі на тестовій вибірці.

Таблиця 12. – Істинність операції АБО (∨)

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	1

1. Створити модель штучного нейрона, підібравши вагові коефіцієнти.
2. Створити модель штучної мережі для визначення таблиці істинності **логічної операції АБО**
3. Перевірити функціональність створеної модель нейронної мережі на тестовій вибірці

Завдання 3. Мовою програмування ***Python***, створити модель простої нейронної мережі, типу персептрон, яка складається з двох шарів: вхідного та вихідного. Відповідно до заданого варіанту (табл. 13) необхідно:

1. Скласти таблицю істинності для заданого логічного виразу (X_1, X_2, X_3).
2. Ввести в якості вхідних значень всі можливі комбінації X_1, X_2, X_3, а в якості вихідних значень – виходи логічної функції за складеною таблицею істинності.
3. Створити нейронну мережу та провести її навчання нейронної мережі. Вивести значення сформованих вагових коефіцієнтів.
4. Протестувати отриману нейронну мережу, задаючи в якості вхідних значень всі можливі комбінації значень X_1, X_2, X_3.
5. Проаналізувати роботу нейронної мережі.

Таблиця 13. – Варіанти логічних функцій до виконання завдання 3.

№ варіанту	*Логічна функція*
1	$(X_1 \vee X_2) \wedge X_3$
2	$X_1 \wedge X_2 \wedge X_3$
3	$X_1 \wedge (X_2 \vee X_3)$
4	$X_1 \vee (X_2 \vee X_3)$
5	$(X_1 \wedge X_2) \vee X_3$
6	$(X_1 \vee X_2) \wedge X_3$
7	$X_1 \vee (X_2 \vee X_3)$
8	$X_1 \wedge X_2 \vee X_3$
9	$X_1 \vee X_2 \wedge X_3$

Завдання 4. На мові програмування ***Python***, побудувати функцію активації відповідно заданого у варіанті проміжку (табл. 14). Враховуючи задані параметри, побудувати структуру персептрона. Розробити алгоритм навчання прямого поширення похибки в нейронній мережі. Побудувати графік, на якому відобразити розподіл двох класів відповідно заданої лінійної функції. Протестувати отриману нейронну мережу, задаючи в якості входів значення $X_1 – X_n$.

Зміст звіту

1. Титульна сторінка.
2. Тема та мета практичного заняття.
3. Опис виконання завдань по пунктам з наданням рисунків і скріншотів.
4. Текст створених програмних файлів.
5. Висновки, що відображують результати та їх критичний аналіз.

Таблиця 14. – Варіанти логічних функцій до виконання завдання 4.

№ варіанту	*Кількість входів*	*Діапазони значень входів*	*Розділова пряма для кожного нейрона*
1	3	[–10... +10]	y=2x+5
2	4	[– 6... + 6]	y=3x-4
3	3	[–8... + 8]	y=2,2x+3
4	2	[– 5... + 5]	y=2x-1
5	2	[– 7... + 7]	y=3x-2
6	3	[– 2... + 3]	y=x+2
7	2	[–1... +1]	y=1,2x-2,5
8	4	[–3... + 3]	y=3x-1,5
9	3	[–8... + 8]	y=0,5x+1
10	2	[– 7... + 7]	y=1,5x+5

Теоретичні відомості

Штучна нейронна мережа (ШНМ) – набір елементарних нейроподібних перетворювачів інформації (нейронів), з'єднаних між собою.

На вхід формального нейрона надходить набір вхідних сигналів $x_1, x_2, ..., x_n$ або вхідний вектор x. Кожен вхідний сигнал помножується на відповідну вагу $w_1, w_2,..., w_n$. Вага зв'язку є скалярною величиною, додатною для збудливих і від'ємною для гальмуючих зв'язків. Зважені вагами зв'язків вхідні сигнали надходять на блок сумації, де здійснюється їх алгебраїчне підсумовування та визначається рівень збудження нейроподібного елемента y:

$$y = f\left(\sum_{i=0}^{n} w_i x_i + \theta\right)$$

де,

f – нелінійна функція активації,

θ – деякий постійний зсув.

Для більшості моделей ШНМ такий нейрон є основним конструкційним елементом і часто представляється графічно рис. 47.

Одношаровий персептрон є одним з найпростіших варіантів ШНМ і містить лише один нейрон. В якості функції f за звичай використовуються найпростіші нелінійні функції, такі як:

- бінарна (порогова) функція:

$$f(x) = \begin{cases} 1, \text{при } x > 0 \\ 0, \text{при} < 0 \end{cases}$$

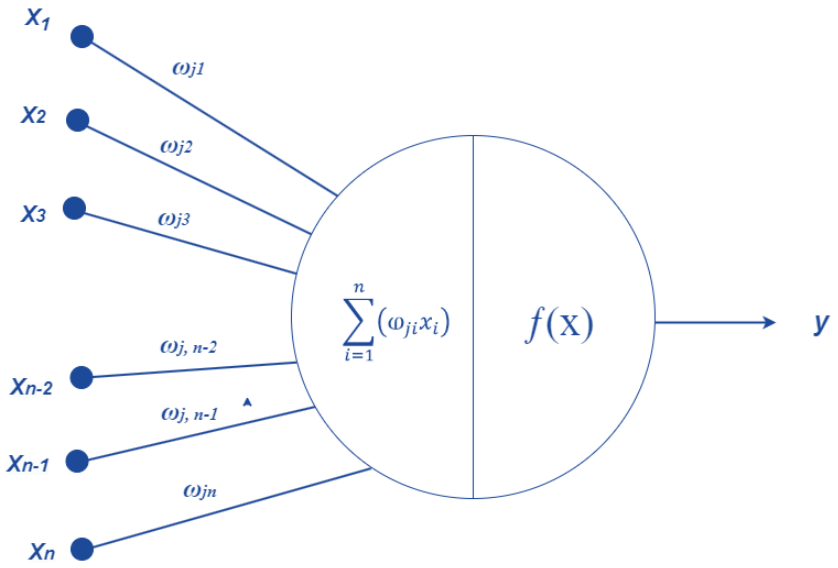


Рисунок 47. – Модель штучного нейрона.

- сигмоїдна функція:

$$f(x) = \frac{1}{1 + e^{-ax}}$$

Одним з типів нейронних мереж є мережі прямого поширення або багатошаровий персептрон рис. 48.

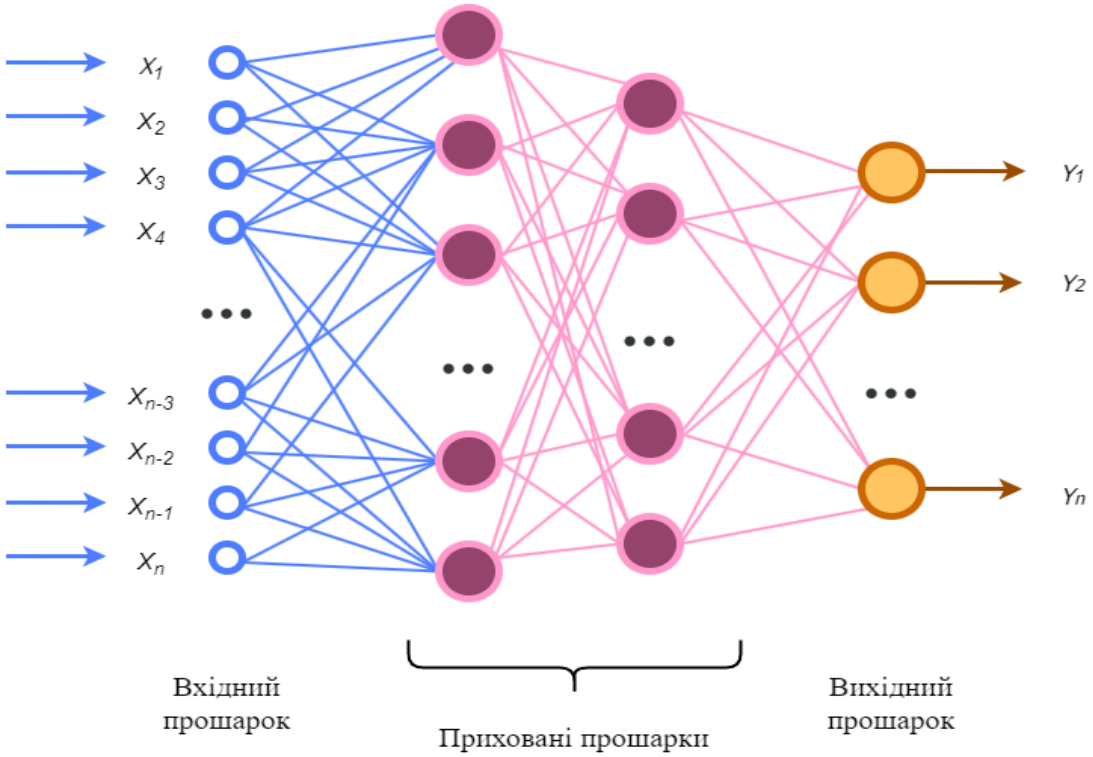


Рисунок 48. – Структура мережі прямого розповсюдження.

Штучна нейронна мережа прямого поширення складається з вхідного шару, декількох схованих шарів і вихідного шару і описується формулою:

$$y_i = f\left(\sum_{i=0}^{n} W_{ji} x_i + \theta_j\right)$$

Навчання нейронної мережі.

ШНМ прямого поширення навчаються за методом навчання з учителем, коли для відомих даних на вході мережі ми знаємо, який результат повинен бути на виході. В такому випадку навчити ШНМ означає знайти такі вагові коефіцієнти w_{ji} і зсуви θ_j для кожного нейрону мережі з формули (4), щоб різниця між бажаними і обчисленими результатами на виході не перевищувалазаданий наперед рівень помилки. В алгоритмах навчання ця помилка описується функцією втрат (loss function). Обчислення w_{ji} і θ_j здійснюється покрокам за формулами:

$$w_{ij}(n+1) = w_{ij}(n) - \beta \Delta w_{ij}$$

$$\theta_j(n+1) = \theta_j(n) - \beta \Delta \theta_j$$

На першому кроці ці значення обираються випадковим чином, тому повторне навчання на тих же даних буде давати різні результати. Спосіб обчислення значень Δw_{ji} і $\Delta\theta_j$ визначається алгоритмом навчання. Суттєве значення в цих формулах має коефіцієнт β, що зветься коефіцієнтом швидкості навчання. Його встановлюють у межах $0 < \beta < 1$ за звичай $\beta = 0.1$. При великих значеннях β є ймовірність проскочити оптимальне значення ваг, при малих значеннях β навчання буде відбуватись довго. Досвід роботи з ШНМ показав, що не для любих даних мережу можна навчити, тому, щоб уникнути нескінченого циклу вводять обмеження на кількість епох. Епоха – це коли в алгоритмі навчання на вхід ШНМ поступають усі дані для навчання. Крім цього існує явище перенавчання, коли кажуть, що мережа не навчилась, а просто запам'ятала вхідні дані. Для виявлення цього явища навчальну вибірку розбивають на основну або тренувальну (*Train*), перевірочну (*Validation*) для виявлення перенавчання і тестову (*Test*) для оцінки якості навченої мережі. Перенавчання фіксується на кроці, коли функція втрат продовжує зменшуватись для тренувальної вибірки і починає збільшуватись для перевірочної. Крім цього для припинення навчання можна задавати максимальний час навчання і специфічні параметри алгоритмів навчання, наприклад, мінімальний рівень градієнту функції втрат, що використовується при обчисленні Δw_{ji} і $\Delta\theta_j$.

Створення нейронної мережі

Вибір структури нейронної мережі являє собою окреме завдання йполягає у виборі топології мережі (кількості шарів і кількості нейронів вкожному шарі)

й функцій активації кожного нейрона. За звичай функція активації однакова для всіх нейронів шару.

Створення навчальної вибірки. Навчальна вибірка повинна мати достатній розмір, щоб забезпечити ефективність навчання, враховуючи, що в алгоритмах навчання вона випадковим чином розбивається на тренувальну, перевірочну і тестову. Зазвичай співвідношення між Train, Validation та Test вибірками: 70: 20: 10.

Функція втрат (*loss function*) показує якість навчання нейронної мережі або ступінь відповідності множини виходів ШНМ бажаній множині виходів. ***Епоха*** – крок навчання ШНМ, за який на вхід послідовно представляютьсяусі вектори навчальної вибірки.

Модель штучного нейрона

Штучний нейрон (формальний нейрон, нейроподібний елемент) – це примітивний обчислювальний пристрій (або його модель), що має кілька входів і один вихід, і є основним обчислювальним елементом НМ. Схему штучного нейрона зображено на рис. 47. На вхід одношарового нейрона надходить вхідний вектор – набір вхідних сигналів $x=\{x_j\}$, $j = 1, 2, ..., N$, де N – кількість входів.

Кожний вхідний сигнал x_j зважується (масштабується певним чином) відносно зіставленої йому ваги зв'язку (вагового коефіцієнта) w_j, яка моделює перетворення сигналу у синапсі (міжнейронному контакті).

Дискримінантна (вагова, постсинаптична) функція нейрона φ поєднує зважені вхідні сигнали, отримуючи постсинаптичний потенціал та подає його значення до функції активації (передатної функції) ψ, яка видає скалярне значення, що видається на виході нейрона. Таким чином, формальний нейрон реалізує скалярну функцію векторного аргументу, моделюючи перетворення вхідних сигналів у синапсах та тілі біологічного нейрона.

Отже, математична модель функціонування штучного нейрона описується співвідношенням: $y = \psi(\varphi(w, x))$, де x – вектор вхідних аргументів (сигналів); y – значення на виході нейрона; ψ – функція активації; φ – дискримінантна функція; $w=\{w_j\}$ – вектор, що містить значення вагових коефіцієнтів w_j і значення зсуву (порогове значення) w_0.

Набір вагових коефіцієнтів нейрона w моделює його пам'ять. Тому нейрони можна розглядати як запам'ятовуючі пристрої. У той же час нейрони можуть розглядатися як примітивні процесори, що здійснюють обчислення значення функції активації на основі значення дискримінантної функції вхідних сигналів і ваг.

Розглянемо приклад реалізації штучного нейрона для визначення логічної функції АБО з двома вхідними сигналами (табл. 12). Мовою програмування Python побудуємо модель штучного нейрона.

Програмна реалізація штучного нейрона

```
# імпорт бібліотеки по роботі з лінійною алгеброю
from numpy import*

#реалізуємо функцію штучного нейрона
def find_answer(arr):
    #Ініціалізація масиву вхідних даних у вигляді numpy-матриці. Кожен рядок – тренувальний приклад. Стовпці – це вхідні вузли. Для даної задачі 2 вхідні вузли та 4 тренувальні приклади.
  training_set_inputs = array([[0,0], [0,1], [1,0], [1,1]])

    #Ініціалізація вихідних даних. ".T" – функція транспонування матриці-вектора, а в нашому випадку– вихідний вузол. Значить нейрон містить 2 входи та 1 вихід.
  training_set_outputs = array([[0,1,1,1]]).T

    # Випадковий розподіл щоразу буде однаковим, що дозволить відстежити роботу нейрона.
  random.seed(1)
    #  Ініціалізація матриці вагових коефіцієнтів (синапсів) розмірністю 2x1, так як у нейрона 2 входи та 1 вихід. Оскільки пов'язуються всі вхідні вузли з усіма вихідними вузлами, то необхідно сформувати матрицю випадкових величин розмірністю 2x1.
weights = 2*random.random((2,1))-1

    # алгоритм прямого розповсюдження
  for iteration in range(100):
    # визначення сиглоїдної функції
    output = 1/(1+exp(-(dot(training_set_inputs, weights))))
    # оновлення вагових коефіцієнтів
    weights += dot(training_set_inputs.T, (training_set_outputs - output) *
        output *(1 - output))
  # визначення вихідного сигнала нейрона
print(1/(1+exp(-(dot(arr, weights)))))
```

За заданою таблицею істинності логічної функції АБО, перевіримо роботу штучного нейрона на комбінації вхідних сигналів 1,1:

```
arr = array([1,1])
find_answer(arr)
```

Отримаємо результат роботи із значенням `[0.99291741],` що свідчить про достовірність роботи штучного нейрона за таблицею істинності логічної функції **АБО**. Аналогічно можна перевірити інші комбінації таблиці істинності.

Розглянемо приклад реалізації штучного нейрона для визначення логічної функції **АБО** з трьома вхідними сигналами (табл. 15). Мовою програмування ***Python*** побудуємо модель штучного нейрона.

Таблиця 15. – Істинність операції АБО (V) трьох вхідних сигналів

Десяткове число	X_1	X_2	X_2	*Y*
0	**0**	**0**	**0**	**0**
1	**0**	**0**	**1**	**1**
2	**0**	**1**	**0**	**1**
3	**0**	**1**	**1**	**1**
4	**1**	**0**	**0**	**1**
5	**1**	**0**	**1**	**1**
6	**1**	**1**	**0**	**1**
7	**1**	**1**	**1**	**1**

Створимо модель штучного нейрона для вирішення даної задачі. За таблицею істинності логічної функції АБО виберемо комбінації двійкового коду для чисел 1, 7, 5, 3. Виділені такі зразки прийнято називати ***тренувальною вибіркою***. Розглянемо тренувальний приклад, у якому будуть оновлюватись вагові коефіцієнти (рис. 49) та оновимо крайній лівий ваговий коефіцієнт (9.5). Інші значення таблиці істинності залишаються для ***тестової вибірки.***

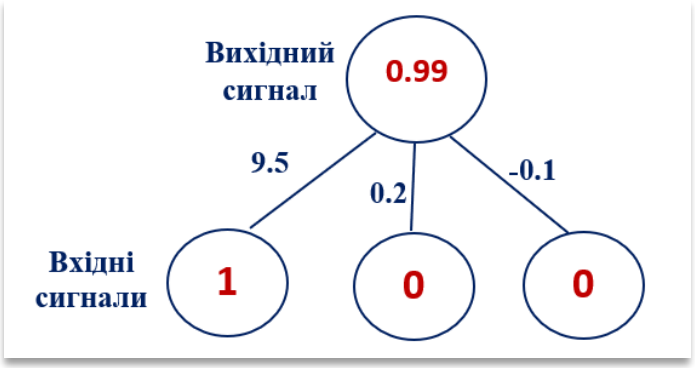


Рисунок 49. – Навчання штучного нейрона.

Навчання нейрона на Python. При навчанні штучного нейрона додається до кожного входу ваговий коефіцієнт сформований випадковим чином, значення яких може бути позитивним або негативним. Процес навчання проходить за наступні етапи (рис. 50):

1. Вибираються вхідні дані з навчальної вибірки, коригуються вагові коефіцієнти та передається розрахований за спеціальною формулою результат на вихід нейрона.
2. Розраховується похибка між прогнозованим результатом та значенням із навчальної вибірки.
3. В залежності від напрямку розповсюдження похибки, коригуються вагові коефіцієнти.
4. Процес навчання виконується за відповідну кількість разів (10000 епох).

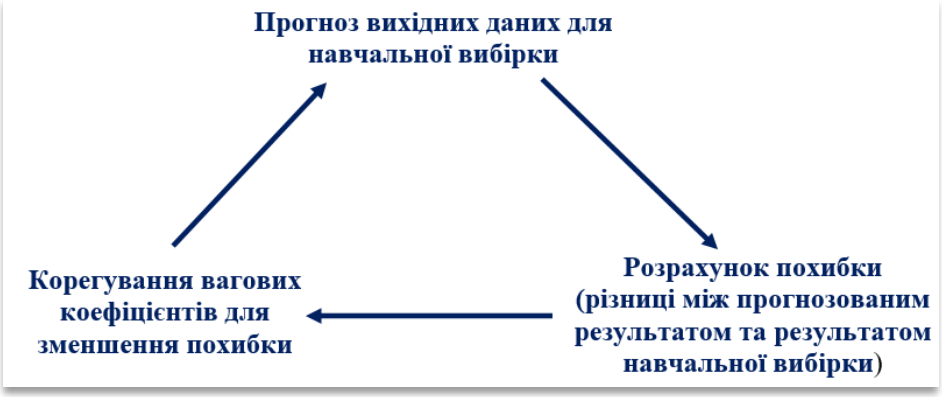


Рисунок 50. – Процес навчання штучного нейрона.

Зрештою, ваговий коефіцієнт досягне оптимального значення на тренувальній вибірці та визначить достовірне прогнозоване значення.

Формула для розрахунку виходу штучного нейрона. Спочатку визначається зважена сума входів нейрона за формулою:

$$\sum weight_i \cdot input_i = weight_1 \cdot input_1 + weight_2 \cdot input_2 + weight_3 \cdot input_3$$

Після чого проводиться нормалізація, так як результат повинен бути в інтервалі від 0 до 1. Для цього використовується математична функція, яка називається ***сигмоїдою*** (***Sigmoid***):

$$y = \frac{1}{1 + e^{-x}}$$

Графік функції сигмоїди наведено на рис. 51 та має S-подібну криву.

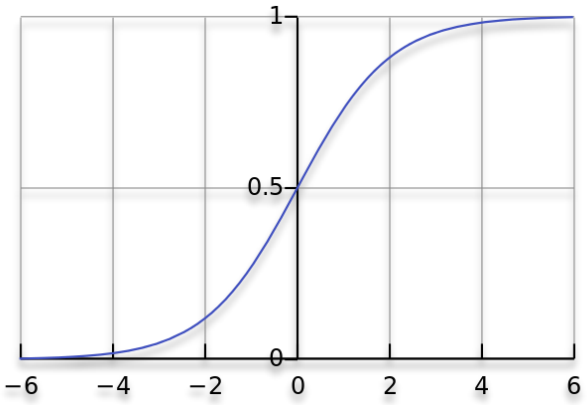


Рисунок 51. – Процес навчання штучного нейрона.

Підставивши перше рівняння в друге, отримаємо остаточну формулу для розрахунку виходу штучного нейрона:

$$Output\ of\ neuron = \frac{1}{1 + e^{-(\sum weight_i \cdot input_i)}}$$

Розрахунок вагових коефіцієнтів. Під час тренувального циклу (рис. 49) проводиться налаштування вагових коефіцієнтів, а їх розрахунок виконується за формулою:

$$Adjust\ weights\ by = error\ \cdot input \cdot SigmoidCurveGradient(output)$$

Оскільки виконується коригування вагових коефіцієнтів виконується пропорційно до величини похибки, то їх перемножують на вхідні значення (0 або 1). Якщо вхідне значення дорівнює 0, вагові коефіцієнти не коригуються. Надалі їх перемножують на градієнт кривої сигмоїди (рис. 51). При цьому важливо враховувати:

1. сигмоїда використовується для розрахунку виходу нейрона;
2. якщо вихідний сигнал є великим позитивним або негативним числом значенням, то це означає, що нейрон достовірно спрогнозував прогноз;
3. як видно з рис. 51, при великих значеннях сигмоїда має незначний градієнт;
4. якщо нейрон визначає правильний ваговий коефіцієнт, то не проходить, даному етапі, процес коригування вагових коефіцієнтів, а застосовується похідна від градієнту сигмоїди:

$$SigmoidCurveGradient(output) = output \cdot (1 - output)$$

Віднявши друге рівняння з першого отримаємо підсумкову формулу розрахунку виходу нейрона:

$$Adjust\ weights\ by = error\ \cdot input \cdot SigmoidCurveGradient(1 - output)$$

Існують інші формули, які дозволяють нейрону навчатися з підвищеною швидкістю. Дана функція має перевагу простоти реалізації.

Програмна реалізація штучного нейрона на Python. Для даної задачі не будуть використовуватись бібліотеки по роботі з нейронними мережами, але застосуємо 4 методи з математичної бібліотеки ***numpy*** такі як:

- exp – визначення експоненти;
- array – робота та створення матриць;
- dot – визначення добутку матриць;
- random - генерація випадкових чисел.

```
from numpy import exp, array, random, dot
training_set_inputs = array([[0, 0, 1], [1, 1, 1], [1, 0, 1], [0, 1, 1]])
training_set_outputs = array([[0, 1, 1, 0]]).T
random.seed(1)
synaptic_weights = 2 * random.random((3, 1)) - 1
```

```
for iteration in range(10000):
    output = 1 / (1 + exp(-(dot(training_set_inputs, synaptic_weights))))
    synaptic_weights += dot(training_set_inputs.T, (training_set_outputs -
    output) * output * (1 - output))
print("Логічна функція АБО числа 6 в двійковому форматі [1, 1, 0]) -> ?: ")
print(1 / (1 + exp(-(dot(array([1, 1, 0]), synaptic_weights)))))
```

Протестувати роботу нейрона можна задавши двійкове число яке не було включене до навчальної вибірки (число 6 у двійковому форматі [1, 1, 0]) та отримаємо наступний результат:

```
Логічна функція АБО числа 6 в двійковому форматі [1, 1, 0]) ->
?: [0.9999225]
```

Отриманий результат свідчить про достовірність роботи штучного нейрона за таблицею істинності логічної функції **АБО**. Аналогічно можна перевірити інші комбінації за таблицею істинності.

Одношаровий персептрон.

Одношаровий персептрон є одним з найпростіших варіантів штучних нейронних мереж і містить лише один нейрон (рис. 52). Будучи самостійною моделлю, одношаровий персептрон (формальний нейрон) є основним конструкційним елементом для більшості моделей нейронних мереж. Одношаровий персептрон використовує як дискримінантну функцію зважену суму, так і активаційну – порогову або сигмоїдну, рідше – лінійну функцію. В залежності від типу функції активації розрізняють дискретні персептрони, що використовують порогову функцію активації, і дійсні персептрони, що використовують дійсні функції активації (сигмоїдну та інші функції).

Розглянемо можливості архітектури нейронної мережі на прикладі простого персептрона для задачі класифікації двох класів ознак, представлених характеристиками *x1*, *x2*.

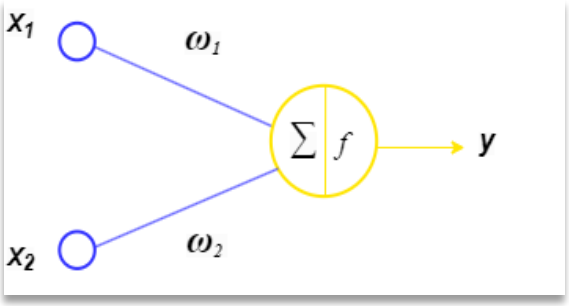


Рисунок 52. – Математична модель персептрона.

Для даної мережі використовується функція активації виду:

$$f(x) = \begin{cases} 1, & x \geq 0 \rightarrow C_1 \\ -1, & x < 0 \rightarrow C_2 \end{cases}$$

Якщо значення суми більше або дорівнює нулю, то вектор належить до класу А:

$$[x_1, x_2]^{-r} \in C_1$$

Інакше, до класу В:

$$[x_1, x_2]^{r} \in C_2$$

Це може бути розділення на будь-які дві множини: 0 або 1, істина / хибність, сигнал / відсутність сигналу. Використовуючи функцію активації, можна побачити, як межа розділює два класи на рівні осей [0,0]:

$$f(x) = \begin{cases} w_1 x_1 + w_2 x_2 \geq 0 \rightarrow C_1 \\ w_1 x_1 + w_2 x_2 < 0 \rightarrow C_2 \end{cases}$$

Надалі визначається сума за формулою:

$$w_1 x_1 + w_2 x_2 = 0$$

Межа розподілу між класами визначається як:

$$x_2 = -\frac{w_1}{w_2} \cdot x_1$$

Тоді функція прямої з кутовим коефіцієнтом записується як:

$$k = -\frac{w_1}{w_2}$$

Розрахована функція, що проходить через початок системи координат наведена на рис. 53.

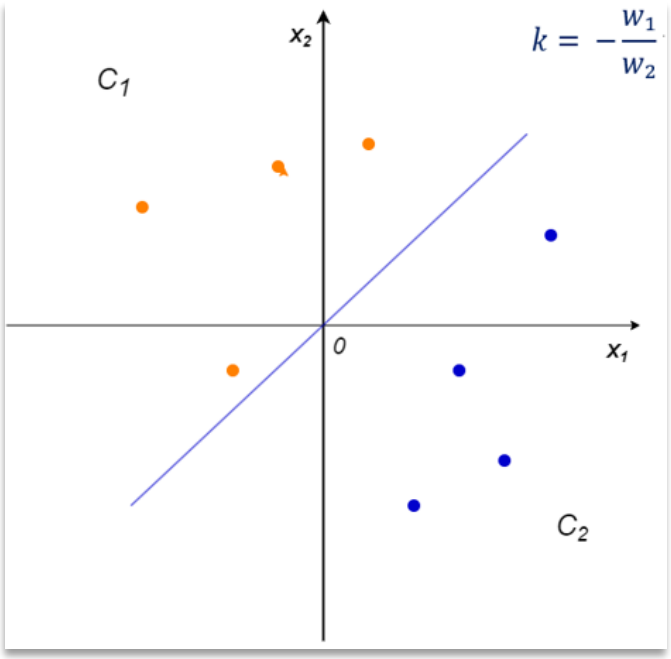


Рисунок 53. - Графік лінійної функції розподілу між двома класами

Усі точки з одного боку графіку від прямої будуть належати до класу C_1, з іншого - до класу C_2. Дану пряму прийнято називати ***класифікатором***. Даний двовимірний графік демонструє можливу класифікацію простим перцептроном

лише для лінійно-роздільних ознак. Реалізуємо дані приклади двох лінійно-розподілених класів для прямої:

$$x_2 = 1 \cdot x_1$$

Дана пряма нахилена під кутом 45° до осей координат. Для правильної класифікації необхідно вибрати вагові коефіцієнти нейронної мережі рівні за абсолютною величиною, але протилежні за знаком:

$$k = 1 = -\frac{w_1}{w_2} \Rightarrow w_1 = -0.5;\ w_2 = 0.5$$

Якщо взяти коефіцієнт 0.5 з протилежними знаками, або іншу величину в інтервалі від -1 до 1, то нейронна мережа успішно класифікує дані ознаки класів. Можна застосувати не рівні вагові коефіцієнти такі як:

$$w_1 = -0.2;\ w_2 = 0.1$$

тоді перший клас буде розпізнано невірно. В даному випадку нейронна мережа неправильно налаштована на класифікацію розпізнавання образів. Отже, правильними є вагові коефіцієнти:

$$w_1 = -0.5;\ w_2 = 0.5$$

Реалізуємо даний прикладу на мові програмування ***Python*** та відобразимо на графіку розподіл значень за двома класами.

Програмна реалізація персептрона

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

N = 5
x1 = np.random.random(N)
x2 = x1 + [np.random.randint(10)/10 for i in range(N)]
C1 = [x1, x2]

x1 = np.random.random(N)
x2 = x1 - [np.random.randint(10)/10 for i in range(N)] - 0.1
C2 = [x1, x2]

f = [0, 1]
w = np.array([-0.5, 0.5])
for i in range(N):
  x = np.array([C2[0][i], C2[1][i]])
  y = np.dot(w, x)
  if y >= 0:
    print("Клас C1")
  else:
    print("Клас C2")

plt.scatter(C1[0][:], C2[1][:], s=10, c='red')
```

```
plt.scatter(C2[0][:], C2[1][:], s=10, c='blue')
plt.plot(f)
plt.grid(True)
plt.show
```

В результаті отримаємо побудований графік розподілу точок за двома класами відносно лінійного класифікатора (рис. 54)

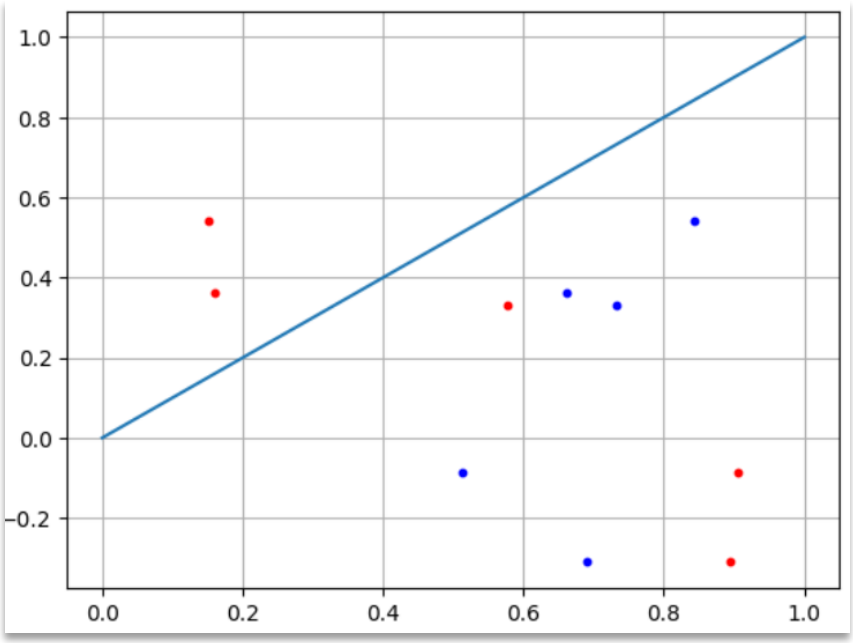


Рисунок 54. – Розподіл двох класів нейронною мережею типу персептрон.

Розглянемо інший випадок, коли ознаки зміщуються вверх відповідно осі x_2 (рис. 55).

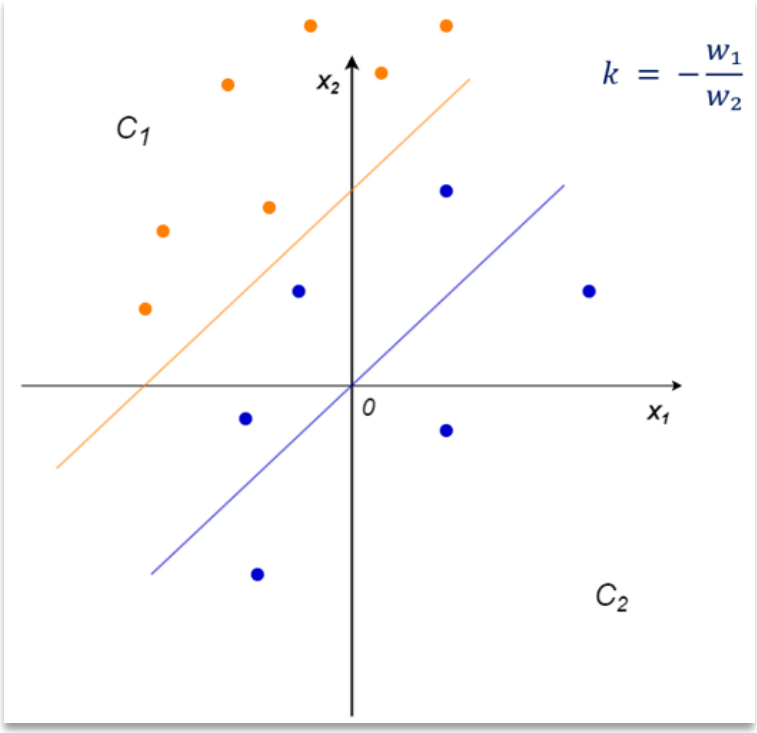


Рисунок 55. - Графік зміщення лінійної функції.

В даному прикладі роздільна пряма не може вірно класифікувати образи, оскільки вона проходить через початок координат. Саме тому потрібно її змістити. У нейронній мережі для цього додають ще один вхід - зсув (англ. bias), для роздільної гіперплощини (рис. 56).

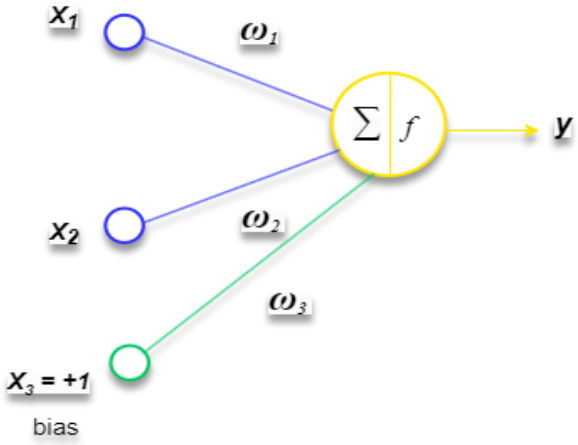


Рисунок 56. - Перцептрон з додатковим входом ***bias***.

Враховуючи додатковий вхід, рівняння прямої має наступний вигляд:

$$x_2 = -\frac{w_1}{w_2}x_1 - \frac{w_3}{w_2}\cdot 1$$

Якщо всі значення зміщені вверх відносно осі x_2 на b, то третій ваговий коефіцієнт нейронної мережі визначається з рівняння:

$$-\frac{w_3}{w_2} = b \Rightarrow w_3 = -b \cdot w_2$$

До вхідного вектора додаємо значення +1 і нейронна мережа достовірно класифікує зміщені ознаки. Даний зсув (bias), зазвичай, використовується в усіх сучасних нейронних мережах.

Програмна реалізація персептрона із зсувом

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

N = 5
b = 3
x1 = np.random.random(N)
x2 = x1 + [np.random.randint(10)/10 for i in range(N)] + b
C1 = [x1, x2]

x1 = np.random.random(N)
x2 = x1 - [np.random.randint(10)/10 for i in range(N)] - 0.1 + b
C2 = [x1, x2]

f = [0+b, 1+b]
w2 = 0.5
```

```
w3 = -b*w2
w = np.array([-w2, w2, w3])

for i in range(N):
  x = np.array([C1[0][i], C1[1][i], 1])
  y = np.dot(w, x)
  if y >= 0:
    print("Клас C1")
  else:
    print("Клас C2")

plt.scatter(C1[0][:], C2[1][:], s=10, c='red')
plt.scatter(C2[0][:], C2[1][:], s=10, c='blue')
plt.plot(f)
plt.grid(True)
plt.show
```

В результаті отримаємо побудований графік розподілу точок за двома класами відносно лінійного класифікатора зміщеного відносно осі ординат на 3 одиниці (рис. 57)

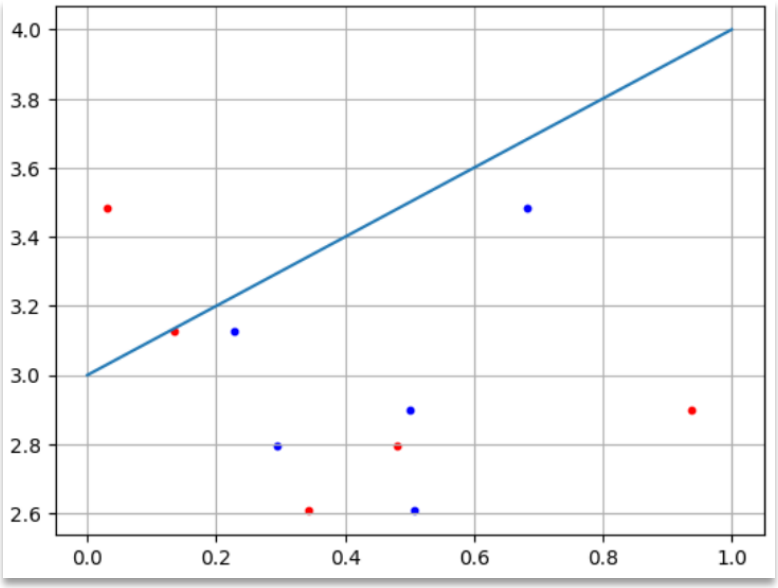


Рисунок 57. - Перцептрон з додатковим входом.

Контрольні запитання

1. Модель штучного нейрона.
2. Спосіб обчислення коефіцієнтів w_{ij} в алгоритмах навчання.
3. Поняття: дані, інформація, інтелект.
4. Визначення штучної нейронної мережі.
5. Біологічний нейрон. Поняття: дендрит, аксон, синапс.
6. Персептрон. Архітектура персептрона.

Практичне завдання 6
Дослідження багатошарових нейронних мереж для вирішення задач класифікації

Мета. Вивчення архітектури багатошарової нейронної мережі та побудова та дослідження моделей персептронних нейронних мереж програмними засобами. Отримання практичних навичок побудови багатошарових нейронних мереж прямого поширення похибки.

Завдання 1. Мовою програмування ***Python***, створити модель багатошарової нейронної мережі типу багатошаровий персептрон, який складається з трьох прошарків: вхідним прошарком із 4-х нейронів, прихованим прошарком із 2-х нейронів та вихідним прошарком з одного нейрона (4-2-1). Відповідно до наведеного варіанту (табл. 16):

1. Скласти таблицю істинності заданого логічного виразу (X_1, X_2, X_3, X_4, приймають значення 0 або 1).
2. Ввести в якості вхідних значень всі можливі комбінації X_1, X_2, X_3,X_4, в якості вихідних значень – значення за таблицею істинності.
3. Провести навчання нейронної мережі і вивести значення вагових коефіцієнтів.
4. Протестувати отриману нейронну мережу, задаючи в якості входу тестову вибірку вхідних значень X_1, X_2, X_3, X_4.
5. Проаналізувати роботу нейронної мережі та визначити якість її навчання.

Таблиця 16 – Варіанти до виконання завдання 1

№ варіанту	*Логічна функція*
1	$(X_1 \vee X_2) \wedge X_3 \wedge X_4$
2	$X_1 \wedge X_2 \wedge (X_3 \vee X_4)$
3	$X_1 \wedge (X_2 \vee X_3) \vee X_4$
4	$X_1 \vee (X_2 \vee X_3) \wedge X_4$
5	$(X_1 \wedge X_2) \vee (X_3 \wedge X_4)$
6	$(X_1 \vee X_2) \wedge X_3 \vee X_4$
7	$X_1 \vee (X_2 \vee X_3) \wedge X_4$
8	$X_1 \wedge X_2 \vee X_3 \wedge X_4$
9	$X_1 \vee X_2 \wedge X_3 \vee X_4$

Завдання 2. Програмними засобами, створити модель багатошарової нейронної мережі, яка виконує підбір товарі клієнта відповідно до наведених в табл. 18 варіантів за 3-4 параметрами. Перевірити працездатність створеної

моделі нейронної мережі на тестовій вибірці. Проаналізувати роботу нейронної мережі.

Таблиця 17. – Перелік рекомендованих предметних областей для створення нейронної мережі

№з/п	Предметна область
1.	Індивідуальний підбір автомобіля
2.	Індивідуальний підбір взуття
3.	Індивідуальний підбір відеокамери
4.	Індивідуальний підбір житла
5.	Індивідуальний підбір комп’ютера
6.	Індивідуальний підбір косметики
7.	Індивідуальний підбір мобільного телефону
8.	Індивідуальний підбір музичного центру
9.	Індивідуальний підбір одягу
10.	Індивідуальний підбір спеціальності для абітурієнтів
11.	Індивідуальний підбір телевізора
12.	Індивідуальний підбір системи відеоспостереження
13.	Персональний підбір спортивного інвентаря
14.	Персональний підбір ділового одягу
15.	Персональний підбір квадрокоптера
16.	Персональний підбір спортивного одягу
17.	Індивідуальний підбір ноутбука
18.	Індивідуальний підбір квадроцикла
19.	Персональний підбір меблів
20.	Індивідуальний підбір ігрової приставки PlayStation

Теоретичні відомості

Нейронні мережі прямого поширення – це нейронні мережі, де нейрони розділено на групи – шари, а зв’язки між нейронами організовано таким чином, що перший (вхідний) шар сприймає зовнішній сигнал, обробляє його і передає значення з виходів своїх нейронів наступному шару, при цьому всередині кожного шару зв’язки між нейронами відсутні, а на входи нейронів схованих шарів (усіх інших шарів, крім першого), поступають значення з відповідних виходів нейронів попереднього шару. Останній шар мережі називають вихідним. Значення з виходів його нейронів утворюють загальний вихідний сигнал НМ.

Задача XOR. Нейронна мережа з одним нейроном може класифікувати лише лінійно-роздільні зразки. Однак на практиці зустрічаються складніші задачі. Уявімо, що класи зразків розподілені наступним чином:

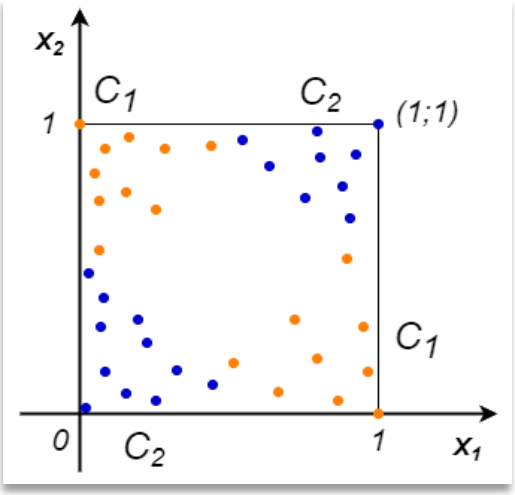


Рисунок 58. - Графік розміщення зразків класів.

У даному випадку неможливо провести одну лінію для їх класифікації. Наприклад лінійні функції будуть розташовані так як показано на рис. 59.

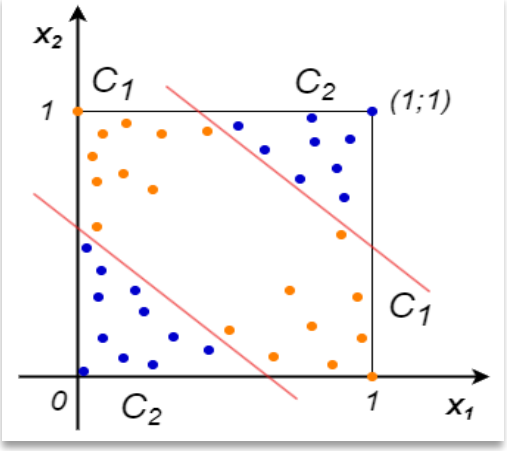


Рисунок 59 – Розподіл класів задачі XOR.

Все, що знаходиться поміж ліній – відноситься до першого класу, поза лініями – другий клас. Кожна роздільна лінія може бути окремим нейроном, а результат їх класифікації об'єднується результуючим нейроном вихідного прошарку (рис. 60).

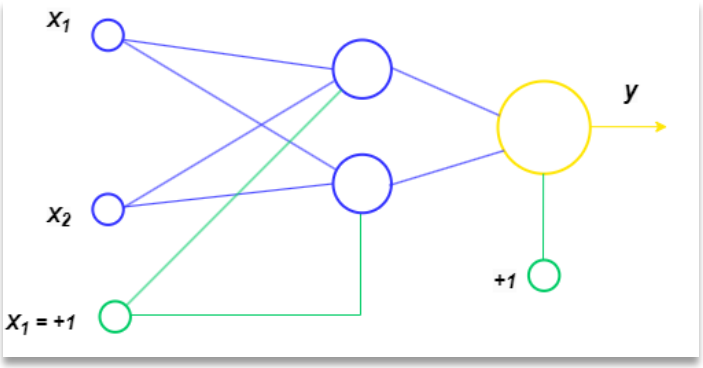


Рисунок 60. - Результуючий нейрон вихідного прошарку.

Припустимо, що на вхід подаються значення 0 або 1: $x_1, x_2 \in [0,1]$. Тоді всі значення будуть розподілені відповідно до табл. 17.

Таблиця 17. – Таблиця істинності операції XOR

X_1	X_2	Y	Клас
0	**0**	**0**	C_2
0	**1**	**1**	C_1
1	**0**	**1**	C_1
1	**1**	**0**	C_2

Якщо $C_2 = 0$ та $C_1 = 1$ отримуємо таблицю істинності операції XOR (виключне АБО). В такому випадку функція активації кожного нейрону буде мати вигляд:

$$f(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

Для вирішення задачі класифікації визначимо вагові коефіцієнти між нейронами. Нехай перший нейрон прихованого прошарку буде формувати межу:

$$x_2 = -1 \cdot x_1 + 1.5$$

Враховуючи формулу:

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{w_3}{w_2} \cdot 1$$

Вагові коефіцієнти першого нейрону для x_1, x_2 дорівнюють:

$$w_1 = w_2 = 1$$

Вагові коефіцієнти третього нейрона w_3 розрахуємо як:

$$w_3 = -b \cdot w_2 = -1.5$$

В результаті отримаємо пряму, яка формує розподіл площини за класами (рис. 61):

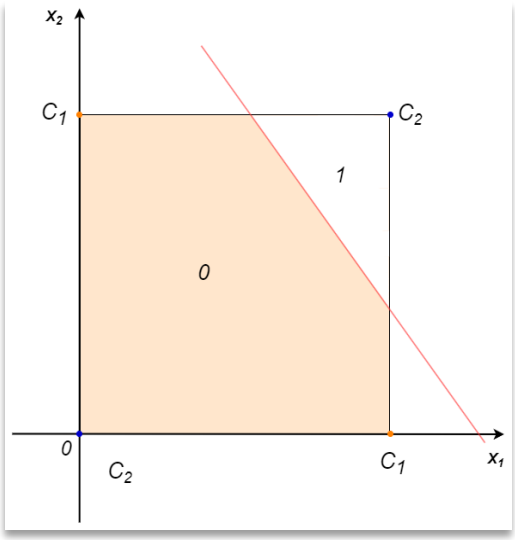


Рисунок 61. - Розподіл класів на площині.

Другий нейрон прихованого прошарку буде формувати розподіл прямої:

$$x_2 = -1 \cdot x_1 + 0.5$$

Вагові коефіцієнти нейронів прихованого прошарку можна прирівняти одне до одного, тоді отримаємо:

$$w_1 = w_2 = 1 \text{ та } w_3 = -0.5$$

Якщо, після проведених розрахунків відобразити на площині лінію розподілу, то отримаємо такий результат, який наведено на рис. 62.

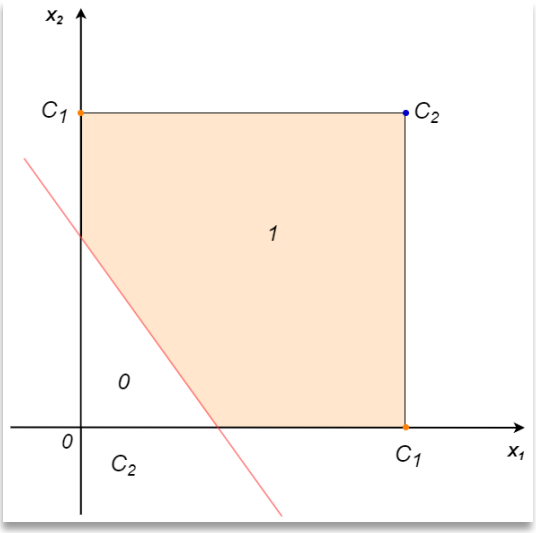


Рисунок 62. - Розподіл класів на площині за результатами налаштування нейронів прихованого прошарку.

Якщо об’єднати результати попередніх розрахунків, то отримаємо наступну розподілену площину, яку наведено на рис. 63.

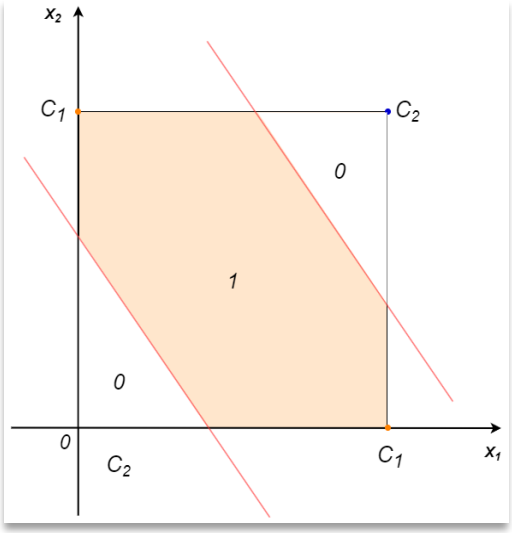


Рисунок 63. - Об’єднана по розподілу площина.

Якщо отримані результати додати, то буде розподіл такого виду, як показано на рис. 64, і тоді на виході активаційної функції отримаємо область із значеннями, які наближені до «1» та іншу область з «0» значеннями.

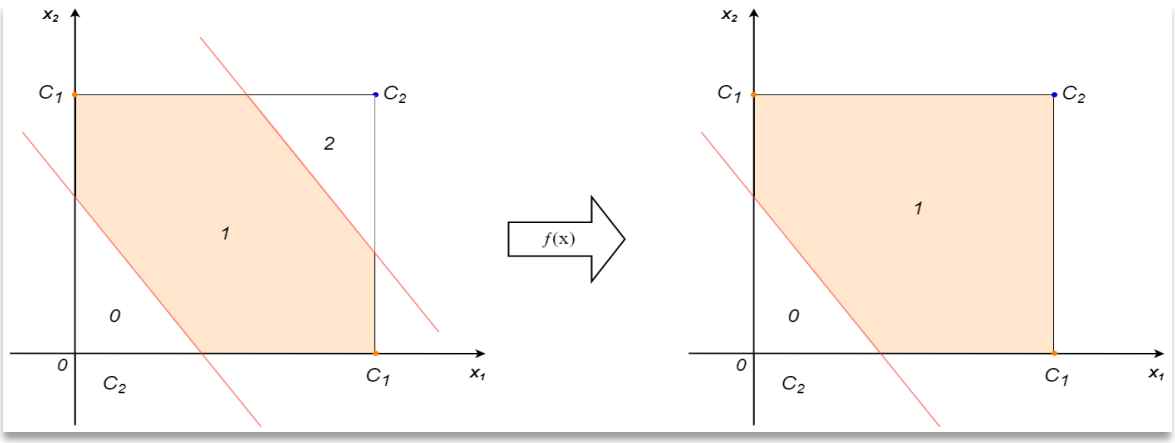


Рисунок 64. – Розподіл даних за двома класами.

Але наданий варіант розрахунку нас не задовольняє, тоді доречним буде визначення різниці між двома функціями (рис. 65). Тоді на виході вихідного нейрона отримаємо відповідне значення розподілу на класи.

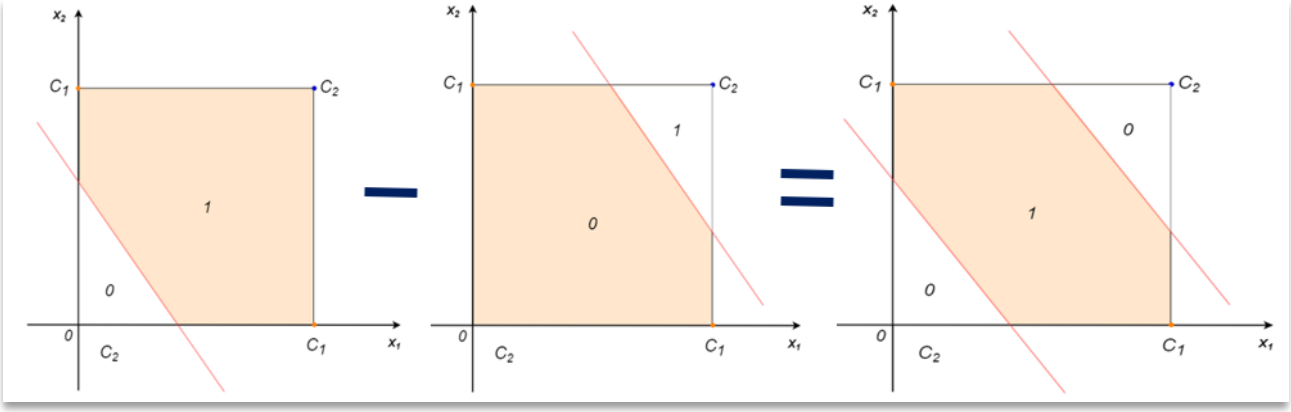


Рисунок 65. – Результат розрахунку різниці між двома функціями

Якщо змістити розділові лінії на значення -0.5, то отримаємо кінцевий результат роботи функції XOR, яку наведено на рис. 66.

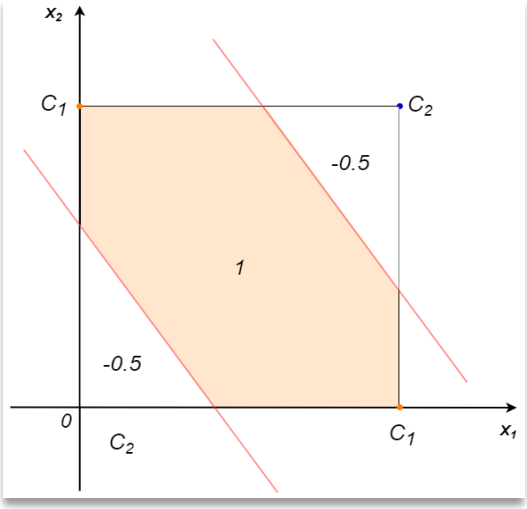


Рисунок 66. - Результат розподілу двох класів для функції XOR.

Тепер на схемі (рис. 67) можна відобразити вагові коефіцієнти нейронів на всіх прошарках нейронної мережі: вхідному, прихованому та вихідному.

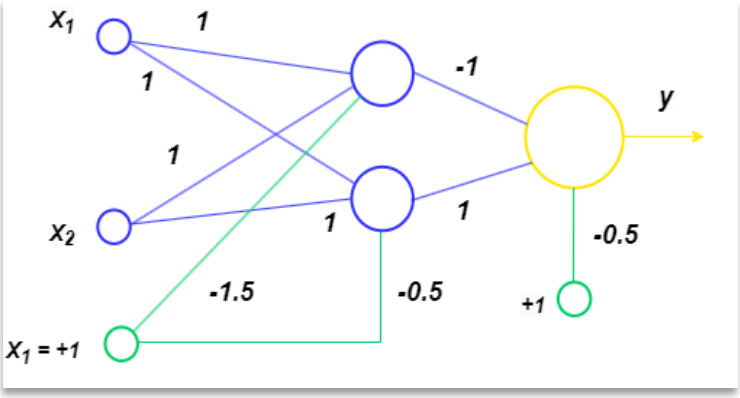


Рисунок 67. – Функціональна схема роботи нейронної мережі для визначення логічної функції XOR.

На мові програмування Python дану штучну нейронну мережу можна реалізувати наступним чином:

Програмна реалізація логічної функції XOR

```
import numpy as np
def act(x):
  return 0 if x <= 0 else 1

def go(C):
  x = np.array([C[0], C[1], 1])
  w1 = [1, 1, -1.5]
  w2 = [1, 1, -0.5]
  w_hidden = np.array([w1, w2])
  w_out = np.array([-1, 1, -0.5])

  sum = np.dot(w_hidden, x)
  out = [act(x) for x in sum]
  out.append(1)
  out = np.array(out)

  sum = np.dot(w_out, out)
  y = act(sum)
  return y

C1 = [(1,0), (0,1)]
C2 = [(0,0), (1,1)]

print(go(C1[0]), go(C1[1]))
print(go(C2[0]), go(C2[1]))
```

В результаті роботи нейронної мережі отримаємо розподіл на класи:

```
1 1
0 0
```

В результаті роботи нейронної мережі бачимо, що класифікація в задачі ***XOR*** успішно виконана завдяки реалізованому прихованому прошарку. Даний приклад демонструє, що приховані прошарки в нейронних мережах забезпечують моделювання більш складних форм класифікаторів на площині та/або в просторі.

Багатошаровий персептрон

Багатошарова нейронна мережа (БНМ) прямого поширення (***багатошаровий персептрон***) складається з формальних нейронів і характеризується наступними параметрами та властивостями:

- M – кількість прошарків нейронної мережі;
- Nμ – кількість нейронів у μ-му прошарку, зв'язки між нейронами у шарі відсутні.

Виходи нейронів μ-го прошарку, μ=1,2,..., M–1, надходять на входи нейронів тільки наступного μ+1-го прошарку. Зовнішній векторний сигнал x надходить на входи нейронів тільки першого прошарку, виходи нейронів останнього M-го прошарку утворюють вектор виходів мережі (M) . Схема мережі показана на рис. 48.

Для кращого розуміння роботи нейронних мереж, розглянемо приклад. При покупці ноутбука, клієнт здійснює власний вибір за такими параметрами, як:

- об'єм оперативної пам'яті;
- розмір дисплею;
- ємність акумулятора.

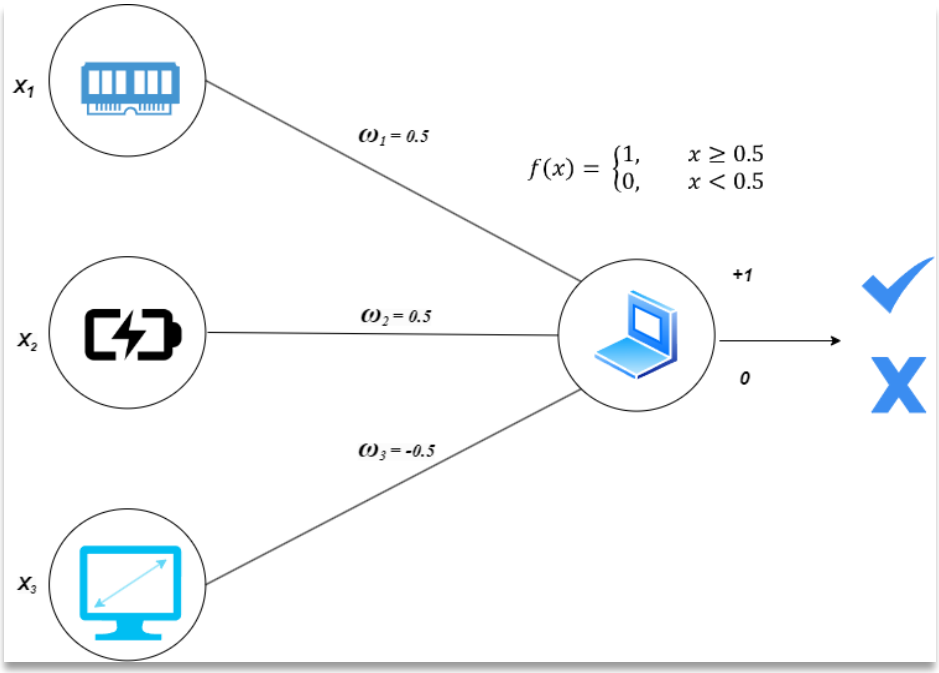


Рисунок 68. - Схематичне зображення організації нейронної мережі.

Необхідно врахувати, що при вирішенні задачі обрані показники параметрів позначимо «1», інші показники «0». Розглянемо такий варіант, коли клієнт обирає ноутбук:

- з розміром пам’яті 16 Гб;
- розміром дисплею 15.6', а також можливі варіанти дисплею в діапазоні від 14.6' до 15.6';
- ємність акумулятора 40 Вт/год, або акумулятори в діапазоні від 31 – 41 Вт/год.

Припустимо, що клієнт знайшов модель, яка підходить за об’ємом пам’яті та ємністю акумулятора, але розмір дисплею перевищує бажаний розмір, то модель нейронної мережі буде мати вигляд, наведений на рис. 68.

В момент вибору ноутбука, на вхід у нейронну мережу подаються три вхідних значення x_1, x_2, x_3 у вигляді значень 0 та 1 відповідно, які відповідають за вибір параметрів пам’яті, акумулятора та дисплея відповідно. Як видно з рис. 68, вагові коефіцієнти для вхідних значень x_1, x_2 є додатними, а для дисплею ваговий коефіцієнт - від’ємний. Надалі кожне вхідне значення перемножується з ваговим коефіцієнтом вхідних значень, які поступають на входи штучного нейрон. Формується сумарний сигнал у вигляді:

$$x = w_1x_1 \cdot w_2x_2 \cdot w_3x_3$$

Надалі дане значення проходить через функцію активації:

$$f(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

В результаті на виході нейрона формується сигнал “так” або “ні”. Якщо клієнтом вибрана модель з двома позитивними параметрами і одним негативним, то на входи нейронної мережі подаються два додатних значення та одне від’ємне та отримаємо граничне значення порогової функції $f(x)$.

$$x = 0.5 \cdot 1 + 0.5 \cdot 1 - 0.5 \cdot 1 = 0.5$$

Розглянемо випадок, коли обрана модель задовольняє клієнта за дисплеєм та акумулятором, але не задовольняє за об’ємом пам’яті. Тоді маємо наступні розрахунки:

$$x = 0.5 \cdot 0 - 0.5 \cdot 0 + 0.5 \cdot 1 = 0.5$$

Якщо модель ноутбука не задовольняє клієнта за розміром дисплеєм та об’ємом пам’яті, але задовольняє за ємністю акумулятора, то отримаємо:

$$x = 0.5 \cdot 0 - 0.5 \cdot 1 + 0.5 \cdot 1 = 0$$

У випадку, коли два параметри є негативними, а один позитивний, то ймовірно що клієнт не вибере модель, бо сума рівна:

$$x = 0.5 \cdot 1 + 0.5 \cdot 1 - 0.5 \cdot 1 = 0$$

На даному прикладі було розглянуто загальний принцип роботи простої нейронної мережі. Даний приклад показує спосіб прийняття рішень нейронними

мережами. Якщо до структури нейронної мережі додати прихований прошарок з двома нейронами, то перший нейрон буде активним, коли клієнт вибере модель за двома параметри, тоді як другий нейрон стане пасивним. В даному випадку нейрон вихідного прошарку набуде значення «1». За даним принципом функціонують нейронні мережі прямого розповсюдження похибки. Реалізуємо дану модель нейронної мережі мовою програмування ***Python***.

Програмна реалізація задачі класифікації

```
import numpy as np

def act(x):
  return 0 if x < 0.5 else 1 # функція активації

def go(memory, display, energy): # вхідний сигнал з 3-ма параметрами
  x = np.array([memory, display, energy]) # вектор з 3-ма вхідними
сигналами
  w11 = [0.3, 0.3, 0] # ваги для 1-го нейрону прихованого прошарку
  w12 = [0.4, -0.5, 1] # ваги для 2-го нейрону прихованого прошарку
  weight1 = np.array([w11, w12]) # матриця 2x3
  weight2 = np.array([-1, 1]) # матриця 1x2
  sum_hidden = np.dot(weight1, x) # визначення суми на вході нейронів
прихованого прошарку
  print("Значення суми на нейронах прихованого прошарку: " +
str(sum_hidden))

  out_hidden = np.array([act(x) for x in sum_hidden]) # пропускаємо суму
через функцію активації
  print("Значення на виходах нейронів прихованого прошарку: " +
str(out_hidden))
  sum_end = np.dot(weight2, out_hidden) # сума на вихідному нейроні
останнього прошарку
  y = act(sum_end)
  print("Вихідне значення нейронної мережі: " + str(y))
  return y

memory = 1
display = 0
energy = 1

result = go(memory, display, energy)
if result == 1:
  print("Ми обрали дану модель")
else:
  print("Подивимось інші моделі")
```

В результаті роботи проектованої нейронної мережі отримаємо персональний вибір технічного обладнання за бажаними параметрами:

```
Значення суми на нейронах прихованого прошарку:[0.3 1.4]
Значення на виходах нейронів прихованого прошарку:[0 1]
Вихідне значення нейронної мережі: 1
Ми обрали дану модель
```

На практиці нейронні мережі реалізовані у вигляді ***нейрокомп'ютера*** – обчислювальної системи, архітектура якої спеціалізована на виконанні операцій, адекватних структур та їх архітектури.

Контрольні запитання

1. Поняття функції активації та їх різновиди.
2. Багатошаровий персептрон: модель і принципи побудови архітектури.
3. Дайте порівняльну характеристику відомих методів навчання багатошарових нейромереж.
4. Математична модель штучної нейронної мережі.
5. Алгоритм прямого поширення помилки.
6. Навчання одношарового персептрона.
7. Недоліки методу прямого поширення помилки.
8. Опишіть функціонування багатошарового персептрона.
9. Опишіть функціонування одношарового персептрона.
10. Порівняння моделей та методів навчання нейромереж прямого поширення.

Практичне завдання 7
Застосування нейронних мереж в машинному навчанні Data Science.

Мета. Придбати практичні навички дослідження можливостей застосування нейронних мереж для вирішення прикладних задач класифікації в машинному навчанні ***Data Science*** програмними засобами ***Python***. Отримати практичні навички створення і навчання мережі прямого поширення.

Завдання 1

1. Програмними засобами побудувати модель нейронної мережі для визначення та класифікації статі особи за двома заданими параметрами: зріст та вага.
2. Сформувати модель штучного нейрона з двома вхідними значеннями, наведеного на рис. 69 з врахуванням сигмоїдної функції активації.

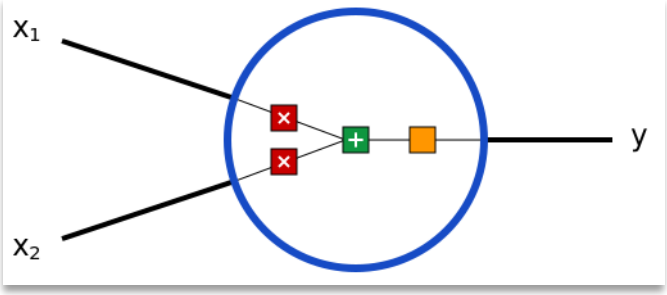


Рисунок 69. – Модель штучного нейрона для задачі класифікації

3. Використовуючи бібліотеку NumPy для Python, розрахувати вхідні значення та отримати на виході значення штучного нейрона, використовуючи алгоритм прямого поширення помилки (feed forward).
4. Створити архітектуру нейронної мережі з вхідним, вихідним та прихованим прошарком, наведеної на рис. 70.

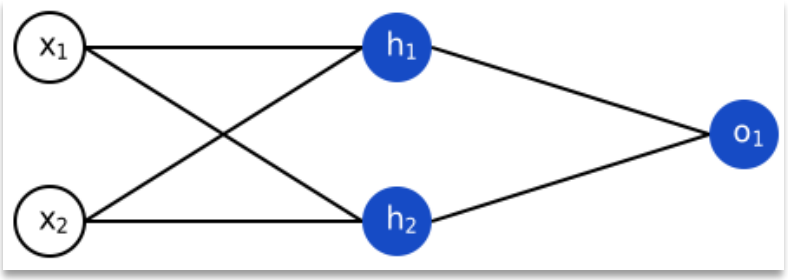


Рисунок 70. – Архітектура нейронної мережі з вхідним, прихованим та вихідним прошарком задачі класифікації

5. Провести навчання нейронної мережі за заданими параметрами, наведених в табл. 19. та вивести значення вагових коефіцієнтів.
6. Протестувати отриману нейронну мережу, задаючи в якості входу різні значення параметрів X1 та X2.

Таблиця 19. Параметри навчальної вибірки в нейронній мережі.

Ім'я	Вага (в фунтах)	Зріст (в дюймах)	Стать
Валерія	133 (54.4 кг)	65 (165,1 см)	Ж
Євген	160 (65,44 кг)	72 (183 см)	Ч
Антон	152 (62.2 кг)	70 (178 см)	Ч
Діана	120 (49 кг)	60 (152 см)	Ж

7. Проаналізувати роботу нейронної мережі та розрахувати середньоквадратичну функцію втрат відповідно до декількох заданих значень змінних (рис. 71) за наступною формулою:

$$L(w_1, w_2, w_3, w_4, w_5, w_6, b_1, b_2, b_3)$$

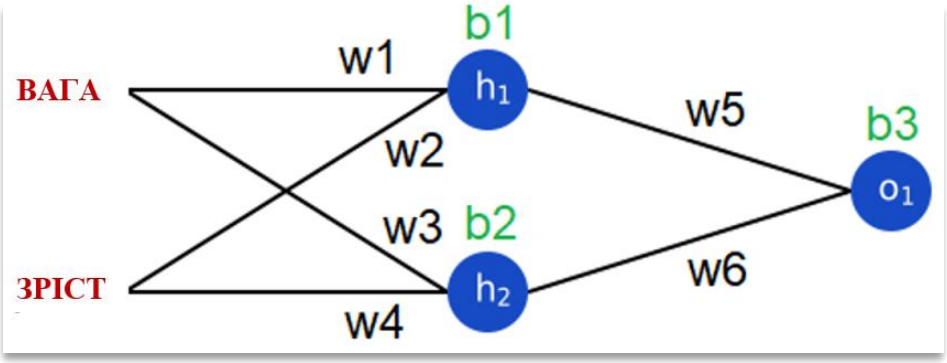


Рисунок 71. – Аналіз роботи нейронної мережі за тестовою вибіркою.

8. Зберегти файл з архітектурою та результатами тестування навченої нейронної мережі.

Завдання 2

1. Програмними засобами побудувати модель нейронної мережі відповідно до варіанту (табл. 20) з врахуванням 2-3 параметрів.
2. Використовуючи алгоритм прямого поширення помилки (feed forward), розрахувати вхідні значення та отримати на виході значення штучного нейрона у відповідності до заданого класу.
3. Провести навчання нейронної мережі за довільними параметрами та вивести значення вагових коефіцієнтів.
4. Протестувати отриману нейронну мережу, задаючи в якості входу різні значення параметрів X_1, X_2, X_3.
5. Проаналізувати роботу нейронної мережі та розрахувати середньоквадратичну функцію втрат.
6. Зберегти файл з архітектурою та результатами тестування навченої нейронної мережі.

Таблиця 20. Варіанти завдань моделювання нейронних мереж при машинному навчанні.

№ варіанту	Предметна область
1.	Сегментація марок автомобілів
2.	Сегментація клієнтів мережі магазинів «Єва»
3.	Сегментація птахів
4.	Сегментація верхнього одягу
5.	Сегментація та дегустація вин
6.	Сегментація спортивного взуття
7.	Сегментація харчової продукції
8.	Сегментація квітів
9.	Сегментація тварин
10.	Сегментація плодових дерев
11.	Продаж нерухомості в житловому секторі
12.	Сегментація новин пошукових систем
13.	Сегментація пасажирів національних авіаліній
14.	Сегментація жіночих сумок
15.	Сегментація кінофільмів за відгуками
16.	Сегментація комах
17.	Сегментація чоловічого/жіночого одягу
18.	Сегментація декоративних риб
19.	Сегментація та продаж букетів
20.	Сегментація спортивного одягу
21.	Сегментація пасажирів таксі Uber
22.	Сегментація чоловічих аксесуарів
23.	Сегментація комп'ютерної техніки
24.	Сегментація університетів країни
25.	Сегментація жіночих аксесуарів
26.	Сегментація побутових приладів
27.	Продаж та реалізація канцелярських товарів
28.	Сегментація валютного ринку
29.	Сегментація чоловічого взуття
30.	Сегментація дитячого взуття

Зміст звіту

1. Титульна сторінка.
2. Тема та мета практичного заняття.
3. Опис виконання завдань по пунктам з наданням рисунків і скріншотів
4. Висновки.

Теоретичні відомості

Нейронні мережі не так вже й складно реалізувати, якщо знати, як моделюється кожен ***нейрон*** мережі, що таке ***функція втрат*** та ***алгоритм зворотного поширення помилки*** (***backpropagation***).

Термін ***нейронні мережі*** асоціюється у багатьох користувачів як, складні математичні моделі, але насправді нейронні мережі легко реалізовувати. Розглянемо на прикладі ***задачу класифікації*** та побудуємо нейронну мережу на мові програмування ***Python***, яка буде виявляти стать особи за відповідними параметрами: вагою та ростом.

Складові елементи штучного нейрона

Насамперед розглянемо структуру нейронів, які стануть базовими елементами проектованої нейронної мережі. Нейрон приймає декілька входів (рис. 72), виконує над ними відповідні математичні операції та на вихід один результат.

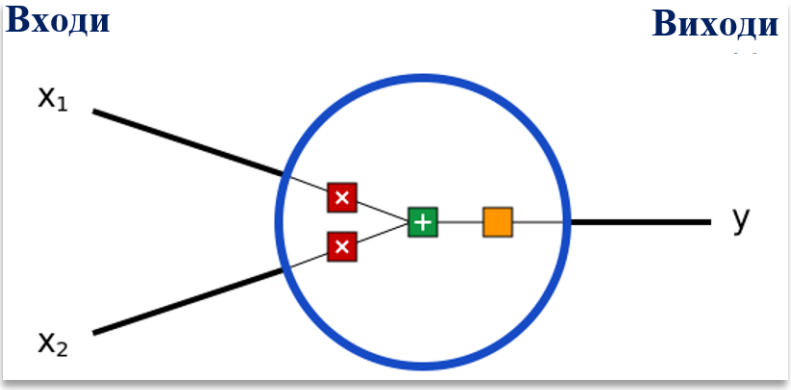


Рисунок 72. – Модель штучного нейрона для задачі класифікації з вхідними та вихідними даними.

Всередині нейрона відбуваються три основні операції:

1. Визначаються значення вхідних даних, які перемножаться з ваговими коефіцієнтами:

$$x_1 \rightarrow x_1 * w_1,\ x_2 \rightarrow x_2 * w_2$$

2. До зважених входів додається значення порогової величини ***b***:

$$h_1 = x_1 * w_1 + x_2 * w_2 + b$$

3. Визначена сума проходить через функцію активації:

$$y = f(x_1 * w_1 + x_2 * w_2 + b)$$

Функція активації перетворює всі вхідні значення у вихідне значення нейрона з використанням функції ***сигмоїди***, графік якої наведено на рис. 73.

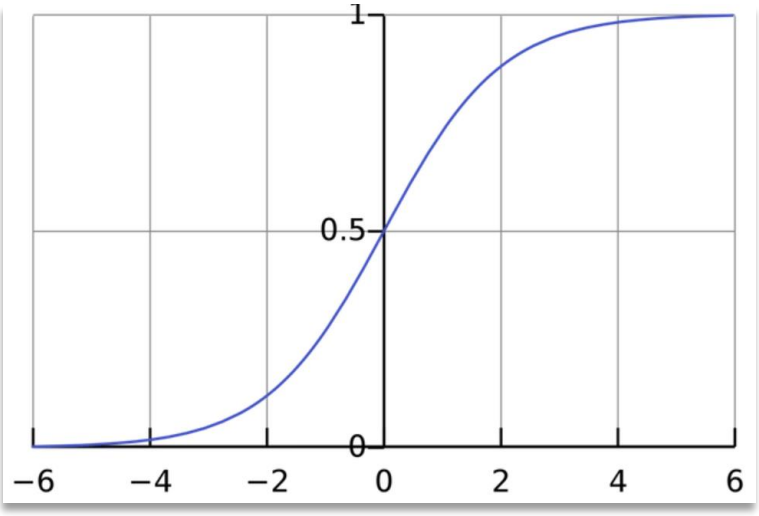


Рисунок 73. – Зовнішній вигляд функції сигмоїди.

Сігмоїда видає результати в інтервалі (***0, 1***). Можна уявити, що вона опрацьовує інтервал значень від мінус нескінченності до плюс нескінченності (0, 1): великі негативні числа перетворюються на числа, близькі до 0, а великі позитивні – до 1. Наприклад, 2-х входовий нейрон використовує сигмоїдну функцію активації та має наступні параметри:

$$w=[0,1]\ b=4$$

де, ***w=[0, 1]*** – запис вагових коефіцієнтів $w_1=0$, $w_2=1$ у векторному вигляді. Якщо підставити нейрону вхідні дані: x=[2, 3], то використовується скалярний добуток векторів, щоб записати формулу в стислому вигляді:

$$(w \cdot x)+b=((w_1 * x_1)+(w_2 * x_2))+b=0 * 2+1 * 3+4=7$$

$$y=f(w \cdot x+b)=f(7)=0.999$$

В результаті нейрон набуває вихідного значення 0.999, якщо на входи подано ***x=[2, 3].*** Такий процес передачі даних називається ***прямим зв'язком*** (***feed forward***).

Програмний код штучного нейрона на мові Python

Для створення моделі штучного нейрона скористаємось влаштованою бібліотекою ***NumPy*** в мові програмування ***Python***, яка допоможе з обчисленням моделі штучного нейрона:

```
# підключення бібліотеки NumPy
import numpy as np

def sigmoid(x):
  # функція активації розраховується за формулою f(x) = 1 / (1 + e^(-x))
  return 1 / (1 + np.exp(-x))

class Neuron:
```

```
    def __init__(self, weights, bias):
      self.weights = weights
      self.bias = bias

    def feedforward(self, inputs):
  # Перемножимо вхідні дані на вагові коефіцієнти, додаємо порогову величину та визначаємо функцію активації
      total = np.dot(self.weights, inputs) + self.bias
      return sigmoid(total)

  weights = np.array([0, 1]) # w1 = 0, w2 = 1
  bias = 4 # b = 4
  n = Neuron (weights, bias)

  x = np.array([2, 3]) # x1 = 2, x2 = 3
  print(n.feedforward(x)) # 0.9990889488055994
```

В результаті виконання даного програмного коду, отримаємо результат розрахованого штучного нейрона – 0.999.

Модель штучної нейронної мережі

Для даної задачі класифікації ***нейронна мережа*** містить декілька нейронів, з'єднаних між собою. Архітектуру нейронної мережі з прихованим прошарком наведено на рис. 74.

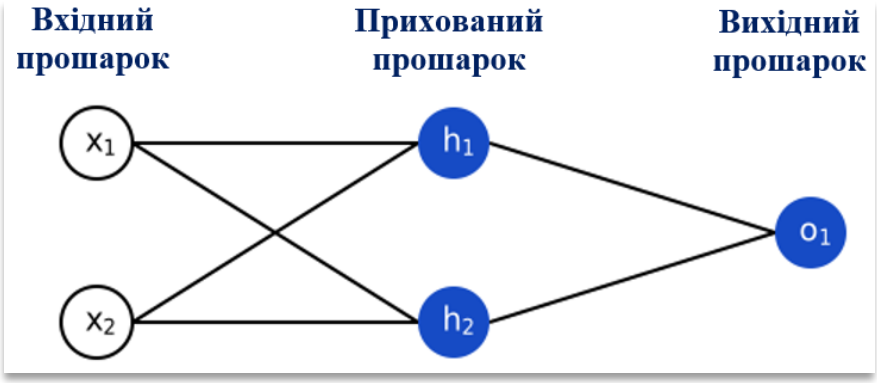


Рисунок 74. – Модель штучної нейронної мережі з прихованим прошарком.

Дана нейронна мережа містить два входи, прихований прошарок з двома нейронами (h_1 і h_2) і вихідний прошарок з одним нейроном (o_1). Варто звернути увагу, що входи для o_1 – це виходи з нейронів h_1 та h_2 прихованого прошарку мережі.

Прихований прошарок – це будь-який прошарок між вхідним (першим) прошарком мережі та вихідним (останнім) прошарком. Прихованих прошарків може бути декілька в штучній нейронній мережі.

На прикладі розглянемо штучну нейронну мережу із прямий зв'язком, архітектуру якої наведено на рис. 74. Припустимо, що нейрони мережі мають однакові вагові коефіцієнти ***w=[0, 1]***, однакові порогові значення ***b=0***, і однакову функцію активації – ***сигмоїду***. Нехай h_1, h_2 та o_1 значення вихідних нейронів мережі. Якщо подано на входи мережі значення x = [2, 3], то отримаємо наступні розрахунки:

$$h_1=h_2=f(w \cdot x+b)=f((0*2)+(1*3)+0)=f(3)=0.9526$$

$$o_1=f(w \cdot [h_1,h_2]+b)=f((0*h_1)+(1*h_2)+0)=f(0.9526)=0.7216$$

Як результат, при подачі вхідних значень ***x=[2, 3]*** нейронної мережі, на виході отримаємо значення нейрона ***0.7216***.

Нейронна мережа може мати будь-яку кількість прошарків і в них може бути будь-яка кількість нейронів. Основна ідея моделювання нейронних мереж: передача вхідних даних по нейронній мережі, до отримання вихідних значень нейронів мережі.

Програмний код нейронної мережі мовою Python

В даній нейронній мережі реалізуємо архітектуру нейронної мережі (2-2-1) з прямим зв'язком, як наведено на рис. 75.

```
import numpy as np
# ... вставте код із попереднього розділу

class OurNeuralNetwork:
  ''' Нейронна мережа з:
    - 2 входами
    - прихованим прошарком з 2 нейронами (h1, h2)
    - вихідним прошарком з 1 нейроном (o1)
  Усі нейрони мають однакові вагові коефіцієнти та порогові величини:  - w = [0, 1] - b = 0  '''
  def __init__(self):
    weights = np.array([0, 1])
    bias = 0

    # Використовуємо клас Neuron з попереднього розділу
    self.h1 = Neuron(weights, bias)
    self.h2 = Neuron(weights, bias)
    self.o1 = Neuron
(weights, bias)

  def feedforward(self, x):
    out_h1 = self.h1.feedforward(x)
    out_h2 = self.h2.feedforward(x)
```

```
        # Входи для o1 - це виходи h1 та h2
        out_o1 = self.o1.feedforward(np.array([out_h1, out_h2]))
        return out_o1

network = OurNeuralNetwork()
x = np.array([2, 3])
print(network.feedforward(x)) # 0.7216325609518421
```

Після виконання даного програмного коду отримаємо вихідний результат штучного нейрона ***0.7216*** в наслідок чого можемо зробити висновок, що нейронна мережа працює вірно.

Навчання нейронної мережі

Навчання нейронної мережі будемо проводити у відповідності до попередньо заданого дата-сету, який наведено в табл. 21.

Таблиця 21. – Дані початково заданого дата-сету.

Ім'я	Вага (в фунтах)	Зріст (в дюймах)	Стать
Валерія	133 (54.4 кг)	65 (165,1 см)	Ж
Євген	160 (65,44 кг)	72 (183 см)	Ч
Антон	152 (62.2 кг)	70 (178 см)	Ч
Діана	120 (49 кг)	60 (152 см)	Ж

Проведемо процес навчання створеної нейронної мережі для прогнозування статі особи за заданими параметрами: вагою та зростом. Дані заданого дата-сету будемо подавати на входи нейронної мережі (рис. 75.)

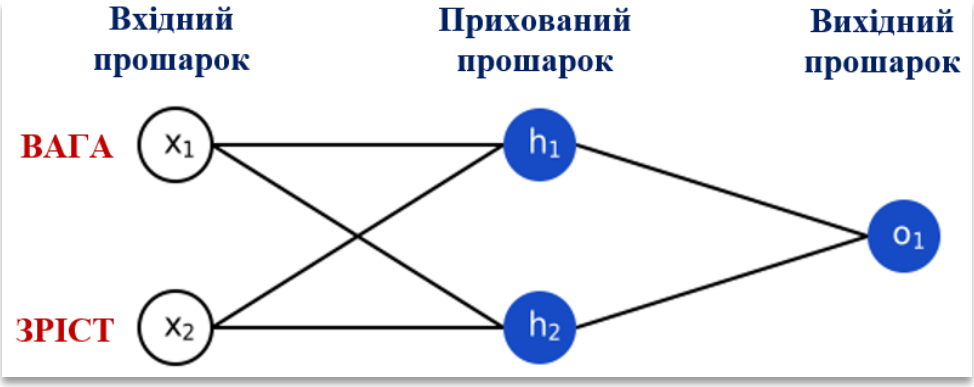


Рисунок 75. – Модель штучної нейронної мережі для класифікації статі особи за вагою та зростом.

В результаті роботи, нейронна мережа буде визначати чоловічу стать як **0**, жіночу – як **1**, а також виконаємо зсув даних на величини (135 та 66), в залежності від розрахованих середніх значень (табл. 22).

Таблиця 22. – Зсув початкових значень заданого дата-сету відповідно до середніх величин.

Ім'я	Вага (в фунтах)	Зріст (в дюймах)	Стать
Валерія	-2	-1	**1**
Євген	25	6	**0**
Антон	17	4	**0**
Діана	-15	-6	**1**

Похибки (функція втрат). Перш ніж навчати нашу нейронну мережу, необхідно визначити наскільки вона точно працює. Тому необхідно в мережі передбачити визначення точності функцією втрати (***loss***). Для її розрахунку використаємо середню квадратичну помилку (***mean squared error, MSE***):

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_{true} - y_{pred})^2$$

де,

- ***n*** – це кількість записів дата-сету, в даному випадку дані кожної особи (Євген, Валерія, Антон, Діана);
- ***y*** – прогнозоване значення статі особи;
- y_{true} – дійсне значення змінної в разі правильної відповіді, тобто 1 – для жіночої статі, 0 – для чоловічої статі;
- y_{pred} – прогнозоване значення змінної, тобто вихідне результативне значення, яке визначає нейронна мережа;
- $(y_{true}-y_{pred})^2$ – розрахунок квадратичної помилки.

В даному випадку функція втрати визначає середнє значення всіх квадратичних помилок, тому її і називають ***середньою квадратичною помилкою***. Важливо пам'ятати, що чим кращими будуть прогнози мережі, тим меншими повинні бути втрати:

- ***Кращі прогнози = менші втрати.***
- ***Навчання нейронної мережі = мінімізація її втрат.***

Розрахуємо на прикладі функцію втрат. Припустимо, що нейронна мережа завжди повертає значення ***0***, тобто визначає, в результат своєї роботи, тільки чоловічу стать. В цьому випадку (табл. 23) розрахуємо функцію втрат за формулою:

$$\text{MSE} = \frac{1}{4}(1 + 0 + 0 + 1) = 0{,}5$$

Таблиця 23. – Розрахунок середньо квадратичної похибки.

Ім'я	y_{true}	y_{pred}	$(y_{true}-y_{pred})^2$
Валерія	0	0	**1**
Євген	1	0	**0**
Антон	0	0	**0**
Діана	1	0	**1**

Функція середньої квадратичної похибки. Запрограмуємо дану розраховану функцію втрат, код якої наведено нижче:

```
import numpy as np

def mse_loss(y_true, y_pred):
  # y_true та y_pred - масиви питру однакової довжини.
  return ((y_true - y_pred) ** 2).mean()

y_true = np.array([1, 0, 0, 1])
y_pred = np.array([0, 0, 0, 0])

print(mse_loss(y_true, y_pred)) # 0.5
```

Навчання нейронної мережі: мінімізація втрат

На даному етапі необхідно мінімізувати втрати в нейронній мережі. Знаючи, що в нейронній мережі можна змінювати вагові коефіцієнти та порогові величини при проведенні змін в прогнозах. Припустимо, що в дата-сеті існують дані тільки для одного запису (табл. 24)

Таблиця 24. – Спрощений дата-сет для відповідного прикладу

Ім'я	Вага (-135)	Зріст (-66)	Стать
Валерія	-2	-1	**1**

В даному випадку середня квадратична помилка буде квадратичною похибкою тільки для даної особи – Євгена:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_{true} - y_{pred})^2 = (y_{true} - y_{pred})^2 = (1 - y_{pred})^2$$

Можна розглянути іншим методом функцію втрат як: функцію в залежності від вагових коефіцієнтів та порогових значень в нейронній мережі (рис. 76). Давайте відзначимо всі ваги та пороги нашої нейронної мережі:

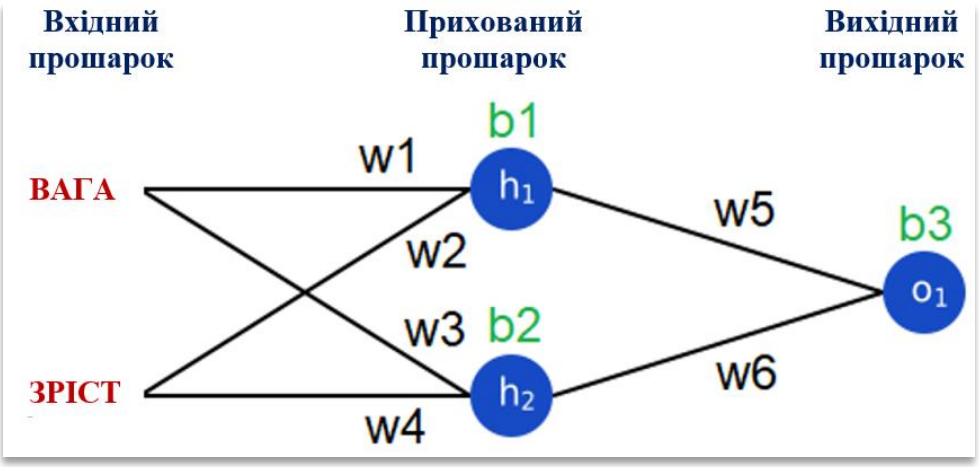


Рисунок 76. – Розрахунок середньої квадратичної похибки в моделі штучної нейронної мережі.

Тепер можна записати функцію втрат як функцію з розрахунком декількох змінних:

$$L(w_1, w_2, w_3, w_4, w_5, w_6, b_1, b_2, b_3)$$

Наприклад, необхідно налаштувати ваговий коефіцієнт w_1, тоді як може змінитися значення функції втрати L. У цьому випадку необхідно визначити часткову похідну за формулою: dL/dw_1.

Визначимо часткову похідну через dy_{pred}/dw_1, використовуючи наступне правило:

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_{pred}} \cdot \frac{\partial}{\partial w_1}$$

Звідси можна визначити dL/dy_{pred}, оскільки вже відомо що $L=(1-ypred)^2$, тоді можна записати:

$$\frac{\partial L}{\partial y_{pred}} = \frac{\partial (1 - y_{pred})^2}{\partial y_{pred}} = -2(1 - y_{pred})^2$$

Якщо перетворити dy_{pred}/dw_1 та позначити виходи нейронів, як h_1, h_2 та o_1, то отримаємо наступний вираз:

$$y_{pred} = o_1 = f(w_5 h_1 + w_6 h_2 + b_3)$$

При врахуванні функція активації сигмоїди $f(x)$ та впливу w_1 тільки на h_1 (але не на h_2), можна використати вище вказане правило та записати:

$$\frac{\partial y_{pred}}{\partial w_1} = \frac{\partial y_{pred}}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_1}$$

$$\frac{\partial y_{pred}}{\partial w_1} = w_5 \cdot f'(w_5 h_1 + w_6 h_2 + b_3)$$

Якщо виразити через dh_1/dw_1 та застосовуючи відповідне правило, то отримаємо:

$$h_1 = f(w_1 x_1 + w_2 x_2 + b_1)$$

$$\frac{\partial h_1}{\partial w_1} = x_1 \cdot f'(w_5 h_1 + w_6 h_2 + b_3)$$

де,

x_1 – це ваговий коефіцієнт;

x_2 – зростання.

Тоді можна обчислити похідну сигмоїдної функції f'(x) за наступною формулою:

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

$$f'(x) = \frac{e^{-x}}{(1 + e^{-x})^2} = f(x) \cdot (1 - f(x))$$

Якщо розкласти dL/dw_1 на декілька частин, можна розрахувати:

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_{pred}} \cdot \frac{\partial y_{pred}}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_1}$$

Даний метод розрахунку часткових похідних називається ***методом зворотного розповсюдження помилки*** (**backpropagation**).

На прикладі розрахуємо часткову похідну для одного запису із запропонованого дата-сету табл. 25

Таблиця 25. – Спрощений дата-сет для розрахунку часткової похідної

Ім'я	Вага (-135)	Зріст (-66)	Стать
Валерія	-2	-1	**1**

Якщо ініціалізувати всі вагові коефіцієнти як 1, а всі порогові величини як 0, то отримаємо прямий прохід в нейронній мережі та отримаємо наступні розрахункові значення:

$$h_1 = f(w_1 x_1 + w_2 x_2 + b_1) = f(-2 + (-1) + 0) = 0.0474$$

$$h_2 = f(w_3 x_1 + w_4 x_2 + b_2) = f(-2 + (-1) + 0) = 0.0474$$

$$o_1 = f(w_5 h_1 + w_6 h_2 + b_3) = f(0.0474 + 0.0474 + 0) = 0.524$$

В результаті нейронна мережа видає значення $y_{pred}=0.524$, що міститься значення, яке можна віднести до одного та іншого класу, як до чоловічого так і до жіночого. Якщо розрахувати dL/dw_1, то отримаємо:

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_{pred}} \cdot \frac{\partial y_{pred}}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_1}$$

$$\frac{\partial L}{\partial y_{pred}} = -2\big(1 - y_{pred}\big) = -2(1 - 0.0524) = -0.952$$

$$\frac{\partial y_{pred}}{\partial h_1} = w_5 \cdot f'(w_5 h_1 + w_6 h_2 + b_3) = 1 \cdot f'(0.0474 + 0.0474 + 0) =$$

$$= f(0.948) \cdot \big(1 - f(0.948)\big) = 0.249$$

$$\frac{\partial h_1}{\partial w_1} = x_1 \cdot f'(w_1 x_1 + w_2 x_2 + b_1) = -2 \cdot f'(-2 + (-1) + 0) =$$

$$= -2 \cdot f(-3) \cdot \big(1 - f(-3)\big) = -0.0904$$

$$\frac{\partial L}{\partial w_1} = -0.952 \cdot 0.249 \cdot (-0.0904) = 0.0214$$

Після визначення похідної сигмоїди за формулою ***f'(x)=f(x)(1-f(x))***, можна судити про збільшенням w_1 та підвищення точності функції втрати.

Навчання нейронної мережі: стохастичний градієнтний спуск

На даному етапі навчання нейронної мережі використовуємо ***алгоритм оптимізації*** під назвою ***стохастичний градієнтний спуск*** (***stochastic gradient descent***), який дозволить визначити зміну вагових коефіцієнтів та порогових значень для мінімізації фактичних втрат. Стохастичний градієнтний спуск визначається за наступною формулою оновлення:

$$w_1 \leftarrow w_1 - \eta \frac{\partial L}{\partial w_1}$$

де, η(eta) – константа, яку називають ***швидкістю навчання*** (***learning rate***). Швидкість навчання визначає процес, який проходить в нейронній мережі під час машинного навчання на відповідній вибірці даних. Якщо відняти значення $eta*dL/dw_1$ із w_1, то отримаємо наступні результати:

- якщо dL/dw_1 є позитивним, w_1 зменшиться, що зменшить **L**;
- Якщо dL/dw_1 є негативною, w_1 збільшиться, що також зменшить **L**.

Якщо реалізувати даний процес в мережі із усіма ваговими коефіцієнтами та пороговими величинами, то в мережі, втрати будуть поступово зменшуватися, і мережа видавати значно точніші результати навчання.

Процес навчання мережі проходить наступні етапи навчання:

1. Вибираємо одне спостереження із дата-сету. При виборі одного спостереження зробить градієнтний спуск стохастичним.
2. Приймаємо всі часткові похідні функції втрат за всіма вагами та порогами як dL/dw_1, dL/dw_2 і т. д.
3. Використовуємо формулу оновлення, щоб відкоригувати значення кожного вагового коефіцієнту та порогових значень.
4. Повторно переходимо до ***<u>кроку 1</u>***.

Програмний код нейронної мережі для задачі класифікації

Реалізуємо програмно нейронну мережу на даному дата-сеті (табл. 22) з врахуванням всіх етапів проектування (рис. 76).

Програмний код на мові ***Python*** для реалізації задачі класифікації

```
import numpy as np

def sigmoid(x):
    # Сигмоїдна функція активації: f(x) = 1 / (1 + e^(-x))
```

```
    return 1 / (1 + np.exp(-x))

def deriv_sigmoid(x):
    # Похідна сигмоїди: f'(x) = f(x) * (1 - f(x))
    fx = sigmoid(x)
    return fx * (1 - fx)

def mse_loss(y_true, y_pred):
    # y_true та y_pred – визначення масивів питру однакової довжини.
    return ((y_true - y_pred) ** 2).mean()

class OurNeuralNetwork:
    ''' Нейронна мережа з:
    - 2 входами
    - прихованим шаром з 2 нейронами (h1, h2)
    - вихідний шар із 1 нейроном (o1) '''
    def __init__(self):
        #Визначення вагових коефіцієнтів
        self.w1 = np.random.normal()
        self.w2 = np.random.normal()
        self.w3 = np.random.normal()
        self.w4 = np.random.normal()
        self.w5 = np.random.normal()
        self.w6 = np.random.normal()

        # Визначення порогових величин
        self.b1 = np.random.normal()
        self.b2 = np.random.normal()
        self.b3 = np.random.normal()

    def feedforward(self, x):
        # x is a numpy array with 2 elements.
        h1 = sigmoid(self.w1 * x[0] + self.w2 * x[1] + self.b1)
        h2 = sigmoid(self.w3 * x[0] + self.w4 * x[1] + self.b2)
        o1 = sigmoid(self.w5 * h1 + self.w6 * h2 + self.b3)
        return o1

    def train(self, data, all_y_trues):
''' - data - масив питру (n x 2) питру, n = кількості спостережень в дата-
    семі. - all_y_trues – масив питру з n елементами. Елементи all_y_trues
    відповідають спостереженням data. '''
        learn_rate = 0.1
        epochs = 1000 # кількість епох в наборі даних
```

```
for epoch in range(epochs):
 for x, y_true в zip(data, all_y_trues):
  # Прямий прохід при навчанні в мережі
  sum_h1 = self.w1 * x [0] + self.w2 * x [1] + self.b1
  h1 = sigmoid(sum_h1)

  sum_h2 = self.w3 * x [0] + self.w4 * x [1] + self.b2
  h2 = sigmoid(sum_h2)

  sum_o1 = self.w5 * h1 + self.w6 * h2 + self.b3
  o1 = sigmoid(sum_o1)
  y_pred = o1

  # Визначаємо часткові похідні.
  # d_L_d_w1 = частковій похідній L відповідно w1
  d_L_d_ypred = -2 * (y_true - y_pred)

  # Нейрон o1
  d_ypred_d_w5 = h1 * deriv_sigmoid(sum_o1)
  d_ypred_d_w6 = h2 * deriv_sigmoid(sum_o1)
  d_ypred_d_b3 = deriv_sigmoid(sum_o1)

  d_ypred_d_h1 = self.w5 * deriv_sigmoid(sum_o1)
  d_ypred_d_h2 = self.w6 * deriv_sigmoid(sum_o1)

  # Нейрон h1
  d_h1_d_w1 = x[0] * deriv_sigmoid(sum_h1)
  d_h1_d_w2 = x[1] * deriv_sigmoid(sum_h1)
  d_h1_d_b1 = deriv_sigmoid(sum_h1)

  # Нейрон h2
  d_h2_d_w3 = x[0] * deriv_sigmoid(sum_h2)
  d_h2_d_w4 = x[1] * deriv_sigmoid(sum_h2)
  d_h2_d_b2 = deriv_sigmoid(sum_h2)

  # Поновлюємо вагові коефіцієнти та порогові величини
  # Нейрон h1
  self.w1 -= learn_rate * d_L_d_ypred * d_ypred_d_h1 * d_h1_d_w1
  self.w2 -= learn_rate * d_L_d_ypred * d_ypred_d_h1 * d_h1_d_w2
  self.b1 -= learn_rate * d_L_d_ypred * d_ypred_d_h1 * d_h1_d_b1

  # Нейрон h2
  self.w3 -= learn_rate * d_L_d_ypred * d_ypred_d_h2 * d_h2_d_w3
  self.w4 -= learn_rate * d_L_d_ypred * d_ypred_d_h2 * d_h2_d_w4
```

```
            self.b2 -= learn_rate * d_L_d_ypred * d_ypred_d_h2 * d_h2_d_b2

            # Нейрон o1
            self.w5 -= learn_rate * d_L_d_ypred * d_ypred_d_w5
            self.w6 -= learn_rate * d_L_d_ypred * d_ypred_d_w6
            self.b3 -= learn_rate * d_L_d_ypred * d_ypred_d_b3

        # Визначаємо повні втрати наприкінці кожної епохи
        if epoch % 10 == 0:
            y_preds = np.apply_along_axis(self.feedforward, 1, data)
            loss = mse_loss(all_y_trues, y_preds)
            print("Epoch %d loss: %.3f" % (epoch, loss))

# Визначаємо навчальну вибірку в дата-сеті
data = np.array([
  [-2, -1], # Валерія
  [25, 6], # Євген
  [17, 4], # Антон
  [-15, -6], # Діана
])
all_y_trues = np.array([
  1, # Євген
  0, # Валерія
  0, # Антон
  1, # Діана
])

# Навчання нейронної мережі
network = OurNeuralNetwork()
network.train(data, all_y_trues)
```

Після запуску програмного коду можна спостерігати за зменшенням значень функції втрат, які проходять на кожному етапі навчання в мережі (рис. 77).

На даному етапі сформуємо тестову вибірку для перевірки адекватності функціонування навчальної мережі.

```
# Задаємо декілька передбачень
emily = np.array([-7, -3]) # 128 фунтів (52.35 кг), 63 дюйми (160 см)
frank = np.array([20, 2]) # 155 фунтів (63.4 кг), 68 дюйми (173 см)
print("Емілі: %.3f" % network.feedforward(emily)) # 0.951 - Ж
print("Френк: %.3f" % network.feedforward(frank)) # 0.039 - М
```

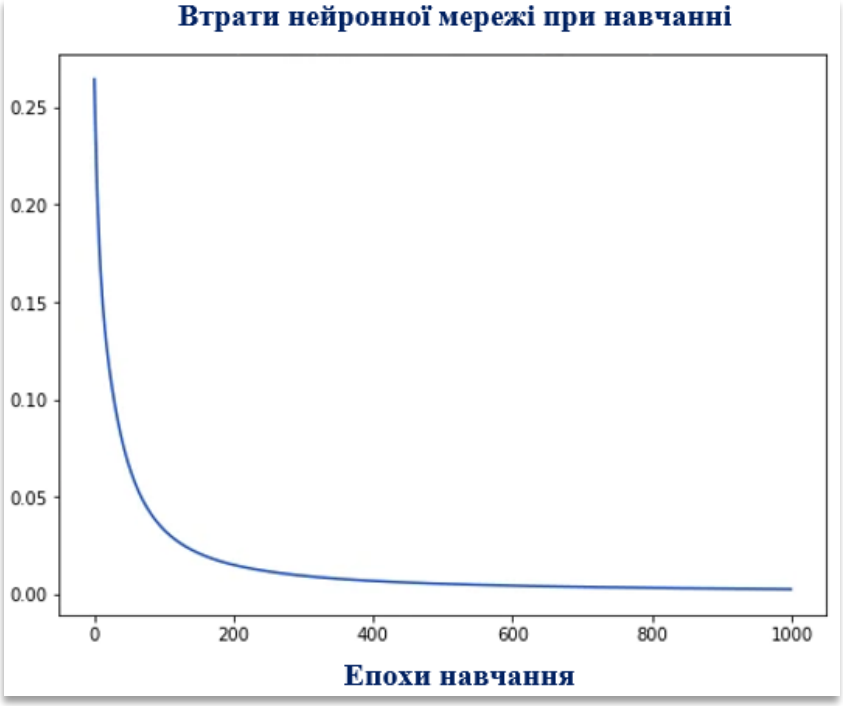


Рисунок 77. - Результативний графік функції втрат за кожну епоху навчання в нейронній мережі

При моделюванні нейронної мережі важливо дотримуватись наступних рекомендацій:

- Визначити нейрони та складові елементи нейронної мережі.
- Використати сигмоїдну функцію активації для розрахунку нейронів.
- Створити дата-сет, в якому необхідно провести нормалізацію за відповідними ознаками та класифікацію.
- Визначити функцію втрат і середню квадратичну похибку (MSE).
- Скористатись методом зворотного поширення похибки (***backpropagation***) для проведення мінімізації втрат та розрахунку часткових похідних.
- Визначити стохастичний градієнтний спуск (SGD) в нейронній мережі.

При моделюванні нейронних мереж можна експериментувати використовуючи відповідні бібліотеки мови ***Python*** таких як Tensorflow, Keras, PyTorch. При цьому можна застосовувати та досліджувати інші функції активації, крім сигмоїдів, (наприклад Softmax), а також інші оптимізатори, окрім стохастичного градієнтного спуску.

Контрольні питання

1. Чи доцільно застосовувати одношаровий персептрон для класифікації складно (нелінійно) роздільних образів? Обґрунтуйте і доведіть відповідь. Наведіть приклади.

2. Чи завжди збігається метод зворотного поширення помилки для багатошарового персептрона?
3. Чи завжди збігаються методи навчання одношарового персептрона?
4. Чи залежить якість навчання нейромереж від якості та обсягу навчальної вибірки?
5. Чи можливе використання неградієнтних методів багатовимірної безумовної оптимізації для настроювання ваг багатошарових нейромереж, і наскільки це доцільно виконувати?
6. Що таке крок навчання? Що таке шар нейронів?
7. Що таке нейронні мережі прямого поширення сигналу?
8. Як визначається цільова функція навчання у методі зворотного поширення помилки?
9. Які задачі можна вирішувати на основі багатошарових персептронів, а які не можна? Обґрунтуйте і доведіть відповідь. Приведіть приклади.
10. Які задачі можна вирішувати на основі одношарових персептронів, а які не можна?
11. Які методи навчання багатошарових нейромереж вам відомі?
12. Які функції активації нейронів найчастіше використовують у нейромережах прямого поширення?

Практичне завдання 8
Розпізнавання рукописних зображень за методами глибокого навчання Deep Learning.

Мета. Придбати практичні навички по розпізнаванню чорно-білих зображень рукописного тексту з використанням бібліотеки Keras; побудові нейронних мереж по розпізнавання образів за методами глибокого навчання ***Deep Learning***.

Завдання 1

1. Програмними засобами побудувати модель нейронної мережі для розпізнавання зображень рукописних цифр, які влаштовані в дата-сеті бібліотеки **Keras**.
2. Побудувати архітектуру нейронної мережі для розпізнавання чорно-білих рукописних цифр та вивести:
   - з бази даних зразків рукописних цифр **MNIST** (скорочення від "***Modified National Institute of Standards and Technology***") 25 зображень;
   - сформувати архітектуру нейронної мережі по розпізнаванню рукописних зразків з **трьома прошарками**: вхідним, прихованим та вихідним;
   - вхідний прошарок **Flatten** для зразків зображень 28x28 перетворити на вектор із 784 елементів;
   - для прихованого прошарку **Dense** задати 128 нейронів, для вихідного – 10 нейронів (рис. 78), враховуючи функцію активації **bias**. Вивести структуру побудованої нейронної мережі на консоль.

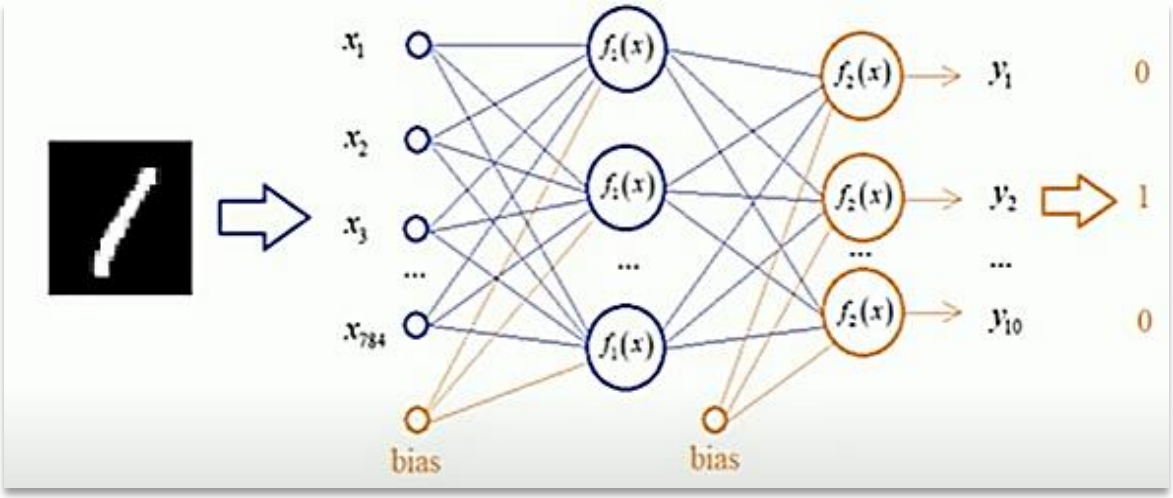


Рисунок 78. – Архітектура нейронної мережі по розпізнавання рукописних зразків.

   - провести стандартизацію (нормалізацію) вхідних даних у бінарному форматі за **10**-ю класами;

- провести навчання нейронної мережі з функцією втрат (**loss function**) та способом оптимізації градієнтного алгоритму по класифікації за категоріальною крос-ентропією (**categorical_crossentropy**) та активаційною функцією вихідних нейронів **softmax**, вказавши метод оптимізації **Adam** з врахуванням відповідної метрики;
- реалізувати процес навчання побудованої нейронної мережі за відповідними параметрами:
  - розмір батча (**batch_size**) 32 зразки;
  - розподіл навчальної та тестової вибірки (**validation_split**) забезпечити у співвідношенні 20% валідації.
- записати результати точності розпізнавання зразків відповідних епох в процесі навчання;
- виконати розпізнавання одного тестового зображення заданого дата-сету та вивести його на консоль;
- визначити кількість не достовірно розпізнаних зображень заданого дата-сету та вивести 5 таких зразків на консоль.

3. Побудувати архітектуру нейронної мережі для розпізнавання рукописних зразків відповідно до варіанту (табл. 26). Вивести на екран відповідні зразки. Записати результати точності навчання НМ.

Таблиця 26. Варіанти завдань для побудови архітектури нейронних мереж глибокого навчання

Параметри	**№ варіанта**								
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Вибір кількості зразків	30	35	40	45	55	60	65	70	75
Кількість нейронів прихованого прошарку	256	64	32	512	300	150	400	100	450
Кількість прихованих прошарків	2	1	2	1	2	1	2	1	2
Кількість епох	3	5	7	9	12	14	6	8	2
Розмір батча	30	25	42	15	20	5	35	28	12
Розподіл валідації	10%	12%	15%	17%	18%	22%	25%	27%	30%
Тестова перевірка цифр	5 25 7	8 12 35	14 20 39	30 20 40	25 17 50	22 11 59	33 44 55	11 22 44	70 50 45
Кількість помилково розпізнаних зразків	10	12	15	20	22	3	7	9	6

4. Зберегти файл з результатами розпізнавання зразків заданого дата-сету за алгоритмом глибокого навчання.

Завдання 2

1. Програмними засобами побудувати модель нейронної мережі для розпізнавання чорно-білих зразків дата-сету **Fashion MNIST**, влаштованого в бібліотеці ***Keras***.
2. Побудувати архітектуру нейронної мережі для розпізнавання рукописних зразків відповідно до варіанту (табл. 26). Вивести на екран відповідні зразки. Записати відповідні результати точності навчання НМ.
3. Зберегти файл з результатами розпізнавання зразків заданого дата-сету за алгоритмом глибокого навчання.

Зміст звіту

1. Титульна сторінка.
2. Тема та мета практичного заняття.
3. Опис виконання завдань по пунктам з наданням рисунків і скріншотів.
4. Висновки щодо виконаного дослідження.

Теоретичні відомості

Keras: бібліотека глибокого навчання на Python

Keras - бібліотека глибокого навчання, що представляє собою високорівневий ***API***, написаний на мові ***Python*** і здатний працювати з ***TensorFlow***, ***Theano***, ***CNTK***. Бібліотека ***Keras*** розроблена для реалізації швидкого навчання за методами ***Deep Learning***, яка:

- дозволяє легко та швидко створювати прототипи (завдяки зручності, модульності та масштабуванню);
- підтримує згорткові, рекурентні мережі, так і їх комбінації;
- працює як на процесорі (CPU), так і на графічному процесорі (GPU).

Keras: розпізнавання рукописних цифр

Перед нами поставлена задача – побудувати нейронну мережу, яка буде розпізнавати рукописні цифри. Скористаємось стандартним дата-сетом, який міститься БД зображень ***MNIST.***

MNIST - (***Modified National Institute of Standards and Technology***) – стандартна база даних зразків рукописного набору цифр. Бібліотека зразків ***MNIST*** постачається разом із ***Keras.*** Для доступу до дата-сету ***MNIST*** потрібно виконати наступний імпорт в ***Python***:

```
import tensorflow as tf
mnist=tf.keras.datasets.mnist
```

Після підключення бібліотек, завантажуємо стандартний дата-сет ***MNIST***:

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

В даному дата-сеті міститься 60 000 зображень у навчальній вибірці та 10 000 – у тестовій. В даному випадку будемо використовувати визначення:

- ***x_train*** – зображення цифр навчальної вибірки;
- ***y_train*** – вектор відповідних цифр (наприклад, якщо на і-му зображенні зображено 5, то y_train[i] = 5);
- ***x_test*** – зображення цифр тестової вибірки;
- ***y_test*** – вектор відповідних цифр для тестової вибірки.

Кожне зображення має розмір 28x28 пікселів. Для обробки зображень необхідно задати їх градацію у відтінках сірого, тобто кожен піксель має значення від 0 до 255, де 0 – чорний колір, 255 – білий (рис. 79):

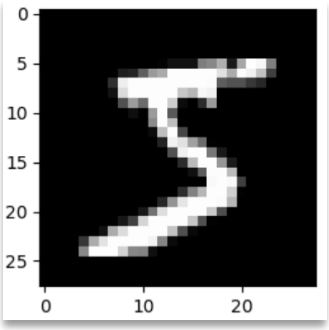


Рисунок 79. – Зразок зображення дата-сету ***MNIST***

Виведемо із дата-сету ***MNIST*** перші 25 зображень, використовуючи наступний програмний код:

```
# Відображення перших 25 зображень з навчальної вибірки MNIST
    plt.figure
    plt.figure(figsize=(10,5))
    for i in range(25):
      plt.subplot(5,5,i+1)
      plt.xticks([])
      plt.yticks([])
      plt.imshow(x_train[i], cmap=plt.cm.binary)

    plt.show()
```

Як бачимо різні зображення мають довільне написанням цифр. Перед нами постає задача - навчити нейронну мережу правильно розпізнавати задані рукописні цифри.

Реалізуємо структуру нейронної мережі. Чіткої методології по моделюванню нейронної мережі немає, тоді як деяка структура НМ вибирається розробником виходячи з його уявлень по реалізації поставленого завдання. В даному випадку скористаємось архітектурою звичайної повнозв'язаної

нейронної мережі, яка містить (рис. 80):

- ***28 x 28 = 784*** входів;
- ***128*** нейронів прихованого прошарку;
- ***10*** нейронів вихідного прошарку.

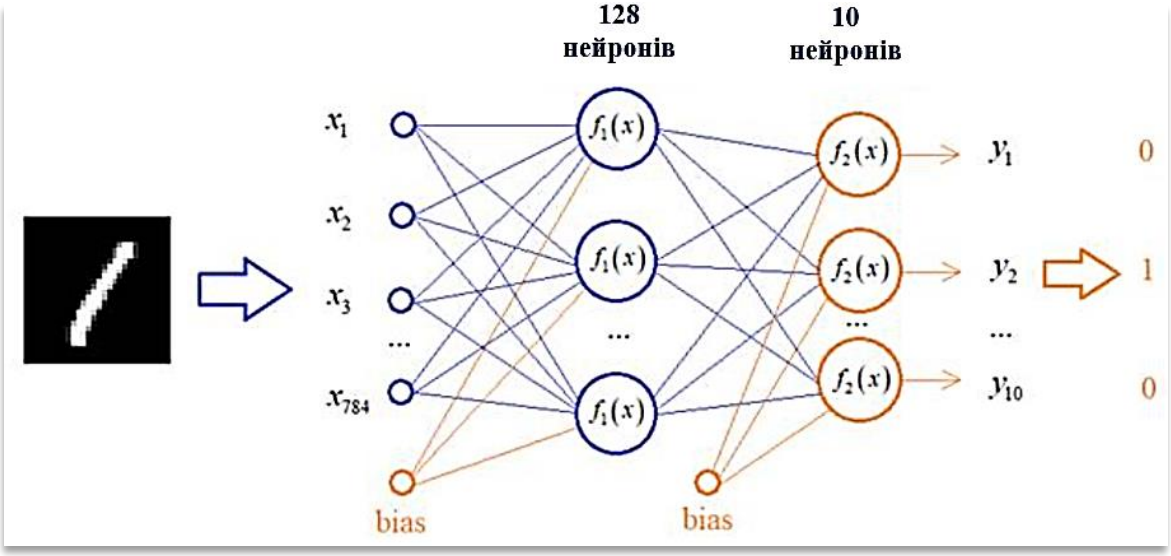


Рисунок 80. – Архітектура нейронної мережі по розпізнавання рукописних зразків.

Для прихованого шару нейронної мережі виберемо функції активації ***ReLu***, для вихідних нейронів - функції активації ***softmax***, що дозволить інтерпретувати вихідні значення в належності до відповідного класу десяткових цифр.

Реалізуємо структуру даної нейронної мережі. Перший (***вхідний***) прошарок повинен перетворювати зображення 28x28 пікселів у вектор із 784 елементів. Для такої операції в ***Keras*** можна створити прошарок спеціального виду – ***Flatten*** (рис. 81).

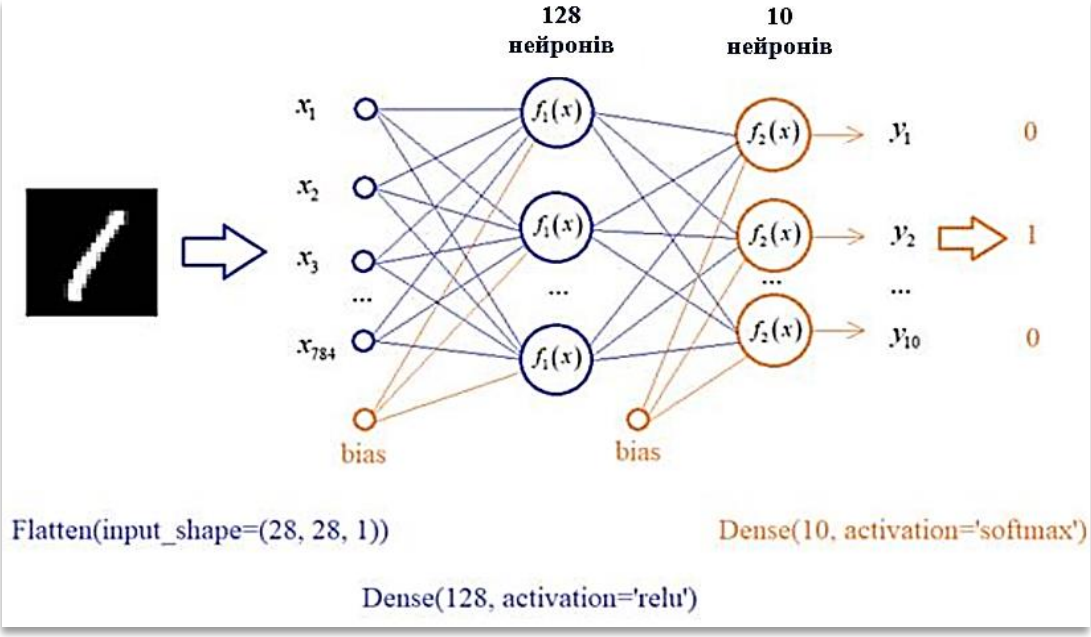


Рисунок 81. – Формування прошарків нейронної мережі

Наступний ***прихований*** прошарок реалізуємо за допомогою класу Dense, який повноцінно пов’яже 784 входи із 128 нейронами даного прошарку. Вихідний прошарок містить 10 нейронів, які пов'язані із 128 нейронами прихованого прошарку. Загальну модель нейронної мережі Keras можна записати як:

```
models=tf.keras.models
layers=tf.keras.layers
Flatten=tf.keras.layers.Flatten
Dense=tf.keras.layers.Dense

model = keras.Sequential([
    Flatten(input_shape=(28, 28, 1)),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])
# виведення структури Нейронної мережі на екран
print(model.summary())
```

На наступному етапі бажано вхідні значення вектора ***x*** стандартизувати таким чином, щоб всі значення знаходились у діапазоні від 0 до 1. Виконаємо дане перетворення наступним програмним кодом:

```
x_train = x_train / 255
x_test = x_test / 255
```

Після виконання, кожне значення тензору ***x_train*** та ***x_test*** буде розділене на максимальне значення 255, яке може прийматись. На виході отримаємо величини в діапазоні від 0 до 1.

Надалі потрібно підготувати відповідний формат вихідних значень. Для кожного зображення рукописної цифри вектор ***y_train*** буде відображати клас відповідного числа (рис. 82).

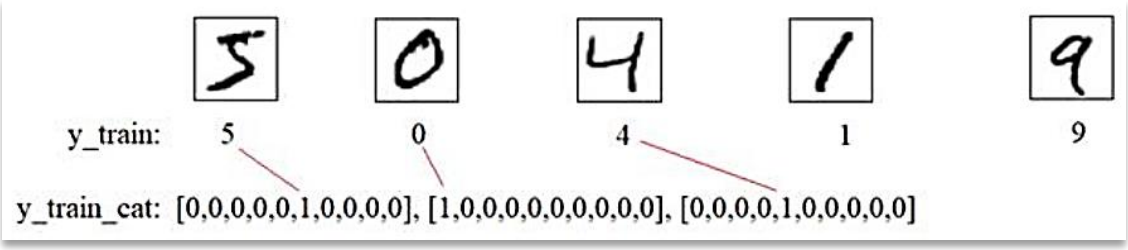


Рисунок 82. – Нормалізація вихідних значень в нейронній мережі

На вихідному прошарку потрібно побудувати вектор з 10-ма виходами, в якому кожен нейрон буде відповідати певній рукописній цифрі від 0 до 9. Для такого перетворення в Keras існує функція, яку застосуємо для навчальної та

тестової вибірки:

```
y_train_cat = keras.utils.to_categorical(y_train, 10)
y_test_cat = keras.utils.to_categorical(y_test, 10)
```

Для отримання наборів векторів ***y_train_cat*** та ***y_test_cat*** за заданим форматом, необхідно вказати другий параметр функції 10, що свідчить про розмірність кожного вектору.

Навчання нейронної мережі по розпізнаванню рукописних цифр

Виберемо функцію втрат (***loss function***) та спосіб оптимізації алгоритм ***градієнтного зсуву***. Важливо для задач класифікації, при моделюванні, першочергово застосовується категоріальна функція крос-ентропії ***categorical_crossentropy*** та активаційна функція вихідних нейронів ***softmax***.

Функцію активації можна записати, з врахуванням критерію якості, наступним чином:

```
model.compile(optimizer='adam',
loss='categorical_crossentropy',
metrics=['accuracy'])
```

В даному програмному коді вказана функція оптимізації ***Adam*** із врахуванням метрики. Метрика, при вирішенні задач класифікації, свідчить про відсоток правильно розпізнаних цифр у нейронній мережі. При побудові алгоритму навчання, необхідно врахувати ***metrics*** мінімізацію похибки у відсотках, що забезпечить зменшення втрат - категоріальної функції крос-ентропії.

На даному етапі нейронна мережа підготовлена до процесу навчання, достатньо запустити на виконання наступний програмний код:

```
model.fit(x_train, y_train_cat, batch_size=32,
epochs=10, validation_split=0.2)
```

де, вказані наступні параметри:

- ***batch_size = 32*** –розмір батча (32 картинки), після яких буде виконуватися коригування вагових коефіцієнтів;
- ***validation_split = 0,2*** – розбиття навчальної вибірки на навчальну та тестову. Значення 0,2 визначає, що на кожній епосі 20% випадкових зразків навчальної вибірки поміщаються у валідаційну вибірку (допустиме значення тестової вибірки від 10% до 30%).

Перевіримо роботу нейронної мережі на тестовій вибірці наступним програмним кодом:

```
#перевірка критерію якості навчання мережі на тестовій вибірці
model.evaluate(x_test, y_test_cat)
```

Метод ***evaluate*** надає можливість визначити тестову вибірку за один прямий прохід та обчислити величину критерію якості та відповідної метрики.

Перевірка розпізнаних рукописних цифр

Виконаємо розпізнавання будь-якого тестового зразка нейронною мережею, виконавши наступний програмний код:

```
n = 1
x = np.expand_dims(x_test[n], axis=0)
res = model.predict(x)
print(res)
```

Спочатку виділяємо із тензора ***n-й*** зразок і пропускаємо його по мережі, використовуючи метод ***predict***. На виході отримаємо 10 значень, за якими визначаємо клас вказаної рукописної цифри. В результаті отримаємо вектор вихідних значень:

[0 0 1 0 0 0 0 0 0]

Максимальне значення відповідає потрібному класу. В даному випадку – це число ***1***, яке є третім виходом, що свідчить про розпізнану цифру ***2***. Виведемо номер максимального числа із даного вектору, скориставшись влаштованою функцією ***argmax*** модуля ***numpy***:

```
print(np.argmax(res))
```

В результаті отримаємо індекс під номером ***2*** та відобразимо на екрані визначений тестовий зразок:

```
plt.imshow(x_test[n], cmap=plt.cm.binary)
plt.show()
```

Отримаємо зображення рукописної цифри ***2***, що, в результаті, свідчить про правильність функціонування побудованої нейронної мережі.

В будь-якому випадку нейронна мережа не може 100% розпізнати всі зразки, які містяться в дата сеті. Визначимо не вірно розпізнані зразки нейронною мережею. Пропустимо через НМ всю тестову вибірку та вектори вихідних значень перетворимо в числа від 0 до 9 наступним кодом:

```
pred = model.predict(x_test)
pred = np.argmax(pred, axis=1)

print(pred.shape)
```

```
print(pred[:20])
print(y_test[:20])
```

Сформуємо маску, яка буде містити ***True*** для вірних варіантів розпізнавання і ***False*** – для невірних. За допомогою даної маски виділимо з тестової вибірки всі неправильні результати:

```
mask = pred == y_test
print(mask[:10])
x_false = x_test[~mask]
y_false = x_test[~mask]

print(x_false.shape)
```

Виведемо перші 5 неправильно розпізнаних нейронною мережею зображень з них на екран:

```
for i in range( 5):
  print("Значеннння мережі: "+str(y_test[i]))
  plt.imshow(x_false[i], cmap=plt.cm.binary)
  plt.show()
```

В даній моделі нейронної мережі можна проекспериментувати з різною кількістю нейронів прихованого шару та проаналізувати результати класифікації. Структуру мережі можна змінювати – задавати в прихованому прошарку не 128 нейронів, як вказано у прикладі, а взяти 100 нейронів або 50; побудувати мережу з двома прихованими шарами і т. д. Важливо, при цьому, спостерігати за величиною функції втрат – чим менше її значення, тим точніше буде нейронна мережа розпізнавати рукописні цифри.

Контрольні питання

1. Що таке розпізнавання образів?
2. Що таке машинне навчання?
3. Що таке образ? Наведіть приклади образів.
4. Назвіть різновиди машинного навчання.
5. Сутність и процесу класифікації.
6. Що таке ознака, вектор ознак і простір ознак?
7. Назвіть різновиди класифікації.
8. Сформулюйте математичну постановку задачі класифікації з учителем.
9. Опишіть приклад класифікації, використовуючи терміни математичної постановки задачі: простір образів, простір ознак тощо.
10. Що таке чутливість, специфічність і точність бінарної класифікації?

Список використаних джерел

Основна

1. Alan Bundy, Rod Burstall. Artificial Intelligence: An Introductory Course. — Revised. — Edinburgh University Press, 1984. — 200 с. — ISBN 978-0852244104. (англ.)
2. Nils J. Nilsson. The Quest for Artificial Intelligence.— Cambridge University Press, 2009. — 578 с. — ISBN 978-0521116398. (англ.)
3. Stuart J. Russell, Peter Norvig. Artificial Intelligence: A Modern Approach. — 3. — Pearson, 2015. — ISBN 978-9332543515. (англ.)
4. Звенігородський О.С., Зінченко О.В., Чичкарьов Є.А., Кисіль Т.М., Штучний інтелект: навчальний посібник / Київ: ДУТ, 2022. – 150 с.
5. Звінігородський О.С., Катков Ю.І., Методичні вказівки до практичних занять з дисципліни "Штучний інтелект" для студентів спеціальності: 121 Інженерія програмного забезпечення, 123 Комп'ютерна інженерія, 124 Системний аналіз, 125 Кібербезпека, 126 Інформаційні системи та технології усіх форм навчання / Київ: ДУТ, 2019. – 79 с.
6. Кузьменко Б.В., Чайковська О.А. Системи штучного інтелекту: Навч.посібник.- К.:Альтерпрес, 2006.-140 с.
7. Лубко Д.В., Шаров С.В., Методи та системи штучного інтелекту: навч. посіб. / укл. – Мелітополь: ФОП Однорог Т.В., 2019. – 264 с.
8. Мороз О., Штучний інтелект // Філософський енциклопедичний словник / В. І. Шинкарук (голова редколегії) та ін. ; Л. В. Озадовська, Н. П. Поліщук (наукові редактори) ; І. О. Покаржевська (художнє оформлення). — Київ : Абрис, 2002. — С. 727. — 742 с. — 1000 екз. — ББК 87я2. — ISBN 966-531-128-Х.
9. Нікольський Ю. В. Системи штучного інтелекту : навч. посібник. – 2-ге вид., випр. та доп. / Нікольський Ю. В. – Львів : Магнолія-2006, 2013. – 279 с.
10. Системи штучного інтелекту: навч. посіб. / Ю. В. Нікольський, В. В. Пасічник, Ю. М. Щербина ; за наук. ред. В. В. Пасічника ; М-во освіти і науки, молоді та спорту України. — 2-ге вид., виправл. та доповн. — Львів: Магнолія-2006, 2013. — 279 с. : іл. — (Серія «Ком'пютинг»). — Бібліогр.: с. 275—278 (58 назв). — ISBN 978-617-57-40-11-4
11. Субботін С.О. Подання й обробка знань у системах штучного інтелекту та підтримки прийняття рішень : навчальний посібник / С.О. Субботін. – Запоріжжя: ЗНТУ, 2008. – 341 с.
12. Ткаченко Р. О., Кустра Н. О., Павлюк О. М., Поліщук У. В., Засоби штучного інтелекту: навч. посіб. / М-во освіти і науки України, Нац. ун-т «Львів. політехніка». — Львів: Вид-во Львів. політехніки, 2014. — 204 с. : іл. — Бібліогр.: с. 200 (11 назв). — ISBN 978-617-607-692-6

13. Чичкарьов Є.А., Зінченко О.В., Єльченко С.В., Прикладне програмування на Python: навчальний посібник / Київ: ДУТ, 2022. – 150 с.
14. Ямпольський Л.С. Нейротехнології та нейрокомп'ютерні системи: підручник / Л.С. Ямпольський, О.І. Лісовиченко, В.В. Олійник. – К.: «Дорадо- Друк», 2016. – 576 с.

Додаткова

1. Глибовець М.М., Олецький О.В. Штучний інтелект: навч. посіб. К.: Видав. дім "КМ Академія". – 2002- 366с.
2. Кавун С.В., Коротченко В.М. Системи штучного інтелекту: навч. посіб. Х: ХНЕУ. – 2007. – 320с.
3. Продеус А.М. Експертні системи в медицині: навч. Посібник / А.М. Продеус, Ю.С. Синєкоп, Є.Я Швець, Є.М. Кісельов, М.М. Баран. К.: ЛОГОС, 2014. – 173 с.
4. Руденко О.Г., Бодянський Е.В., Штучні нейронні мережі: Навчальний посібник. Харків: ТОВ «Компанія СМІТ», 2006. 404 с.

Інформаційні ресурси

1. Основи штучного інтелекту: [електронний ресурс]. Режим доступу: http://dn.dut.edu.ua/course/ОШІ (дата звернення 28.11.2021).
2. Логіка предикатів першого порядку. – Режим доступу: https://sites.google.com/site/anisimovkhv/learning/iis/lecture/tema9
3. Мова логічного програмування Пролог. – Режим доступу: https://sites.google.com/site/anisimovkhv/learning/iis/lecture/tema10
4. Короткий курс машинного навчання або як створити нейронну мережу – Режим доступу: https://habr.com/post/340792
5. Курст лекцій «Нейронні мережі» - Режим доступу: https://www.victoria.lviv.ua/library/students/nn/lecture.html
6. Engati: онлайн-платформа по створенню віртуальних асистентів - Режим доступу: https://help.engati.com/getting-started-with-engati документація
7. Dialogflow - Режим доступу: https://cloud.google.com/dialogflow/docs
8. Сервіс розсилок SendPulse - Режим доступу: https://sendpulse.ua/