Лабораторна робота №1

Прості нейронні архітектури. Вирішення задачі класифікації.

Войцеховський Вадим ПД-31

26.09.2025

2. ВАРІАНТ ІНДИВІДУАЛЬНОГО ЗАВДАННЯ

Завдання (основне). Варіант 15. Об’єктом класифікації буде автомобіль.

Ознаки:

* ознака 1 - час розгону до 100 км/год (с);
* ознака 2 - максимальна швидкість (км/год).

Класи:

* клас 1 (звичайний автомобіль) - розгін більше, ніж за 5 секунд

та швидкість менше за 250 км/год;

* клас 2 (спортивний автомобіль) - розгін менше, ніж за 5 секунд

та швидкість більше за 250 км/год.

3. ВМІСТ НАВЧАЛЬНОЇ ВИБІРКИ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Час розгону (с) | Макс. швидкість (км/год) | Клас | Тип автомобіля |
| 1 | 8.5 | 180 | 1 | Звичайний |
| 2 | 9.2 | 160 | 1 | Звичайний |
| 3 | 7.8 | 200 | 1 | Звичайний |
| 4 | 11.0 | 150 | 1 | Звичайний |
| 5 | 6.5 | 220 | 1 | Звичайний |
| 6 | 10.1 | 170 | 1 | Спортивний |
| 7 | 3.8 | 280 | 2 | Спортивний |
| 8 | 4.2 | 300 | 2 | Спортивний |
| 9 | 3.5 | 265 | 2 | Спортивний |
| 10 | 4.8 | 260 | 2 | Спортивний |
| 11 | 3.1 | 260 | 2 | Спортивний |
| 12 | 5.8 | 200 | 1 | Звичайний |
| 13 | 2.2 | 250 | 2 | Спортивний |
| 14 | 4.5 | 320 | 2 | Спортивний |
| 15 | 3.8 | 270 | 2 | Спортивний |

4. ВМІСТ ТЕСТОВОЇ ВИБІРКИ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Час розгону (с) | Макс. швидкість (км/год) | Клас | Тип автомобіля |
| 1 | 9.5 | 190 | 1 | Звичайний |
| 2 | 7.2 | 210 | 1 | Звичайний |
| 3 | 12.0 | 140 | 1 | Звичайний |
| 4 | 4.5 | 278 | 2 | Спортивний |
| 5 | 3.9 | 310 | 2 | Спортивний |
| 6 | 2.6 | 330 | 2 | Спортивний |
| 7 | 7.4 | 190 | 1 | Звичайний |

5. АРХІТЕКТУРА НЕЙРОМЕРЕЖІ ТА ОПИС ЇЇ ПАРАМЕТРІВ

5.1 Архітектура мережі.

Для вирішення задачі класифікації буде використовуватися простий перцептрон - однонейронна мережа з пороговою функцією активації.

* 1. Структура перцептрона:
* тип - одношаровий перцептрон (лінійний класифікатор);
* вхідний шар - 2 нейрони (для двох ознак - розгону і максимальної швидкості);
* вихідний шар - 1 нейрон;
* функція активації - порогова функція.
  1. Параметри нейромережі:
* алгоритм навчання:
* ініціалізація ваг випадковими малими значеннями;
* встановлення коефіцієнта навчання η = 0.1;
* обчислення вихідного сигналу Y(t);
* порівняння з очікуваним значенням d(t);
* оновлення ваг за формулою:
* wᵢ(t + 1) = wᵢ + η \* e \* xᵢ, де e = d(t) - Y(t).
* повторення до збіжності або досягнення максимальної

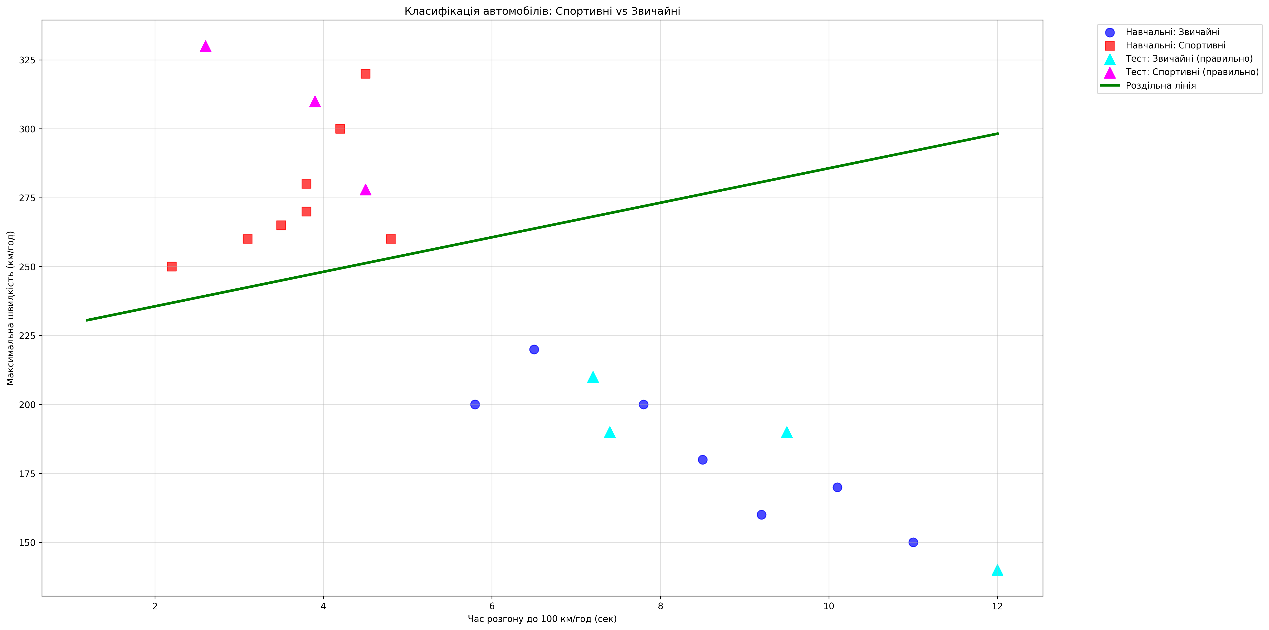
кількості ітерацій.

* швидкість навчання (η) - 0.1;
* максимальна кількість епох - 1000;
* критерій зупинки - досягнення нульової помилки або

максимальної кількості епох;

* функція виходу нейрона:
* Y = f(w₀ \* x₁ + w₁ \* x₂ + w₂), де:
* x₁ - час розгону до 100 км/год;
* x₂ - максимальна швидкість;
* w₀, w₁, w₂ - вагові коефіцієнти;
* f() - функція активації.
* правило оновлення ваг:
* wᵢ(t + 1) = wᵢ(t) + η \* e \* xᵢ, де:
* e = d(t) - Y(t) - різниця між очікуваним та отриманим результатом;
* η - швидкість навчання;
* t - номер ітерації.

6. ГРАФІК, ЯКИЙ ВІЗУАЛІЗУЄ КЛАСИ ДАНИХ ТА РОЗДІЛЬНУ ЛІНІЮ НА ОСНОВІ ЗНАЧЕНЬ ВАГОВИХ КОЕФІЦІЄНТІВ НЕЙРОМЕРЕЖІ



7. ПОКАЗНИКИ ТОЧНОСТІ КЛАСИФІКАЦІЇ НА НАВЧАЛЬНІЙ ТА ТЕСТОВІЙ ВИБІРКАХ

Точність на навчальній вибірці - 100%.

|  |  |  |  |
| --- | --- | --- | --- |
| Зразок | Очікуване | Передбачене | Правильно |
| 1 | Звичайний | Звичайний | Так |
| 2 | Звичайний | Звичайний | Так |
| 3 | Звичайний | Звичайний | Так |
| 4 | Звичайний | Звичайний | Так |
| 5 | Звичайний | Звичайний | Так |
| 6 | Звичайний | Звичайний | Так |
| 7 | Спортивний | Спортивний | Так |
| 8 | Спортивний | Спортивний | Так |
| 9 | Спортивний | Спортивний | Так |
| 10 | Спортивний | Спортивний | Так |
| 11 | Спортивний | Спортивний | Так |
| 12 | Звичайний | Звичайний | Так |
| 13 | Спортивний | Спортивний | Так |
| 14 | Спортивний | Спортивний | Так |
| 15 | Спортивний | Спортивний | Так |

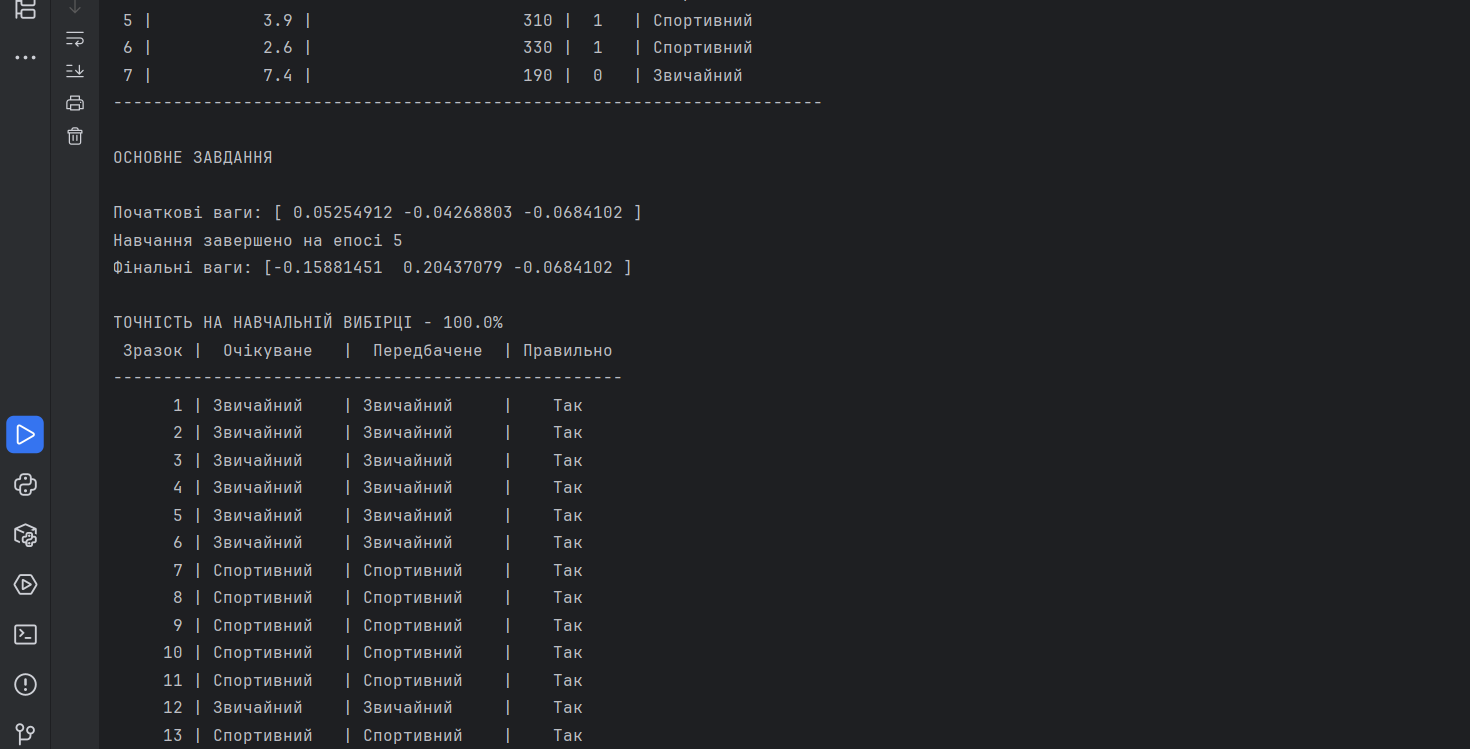
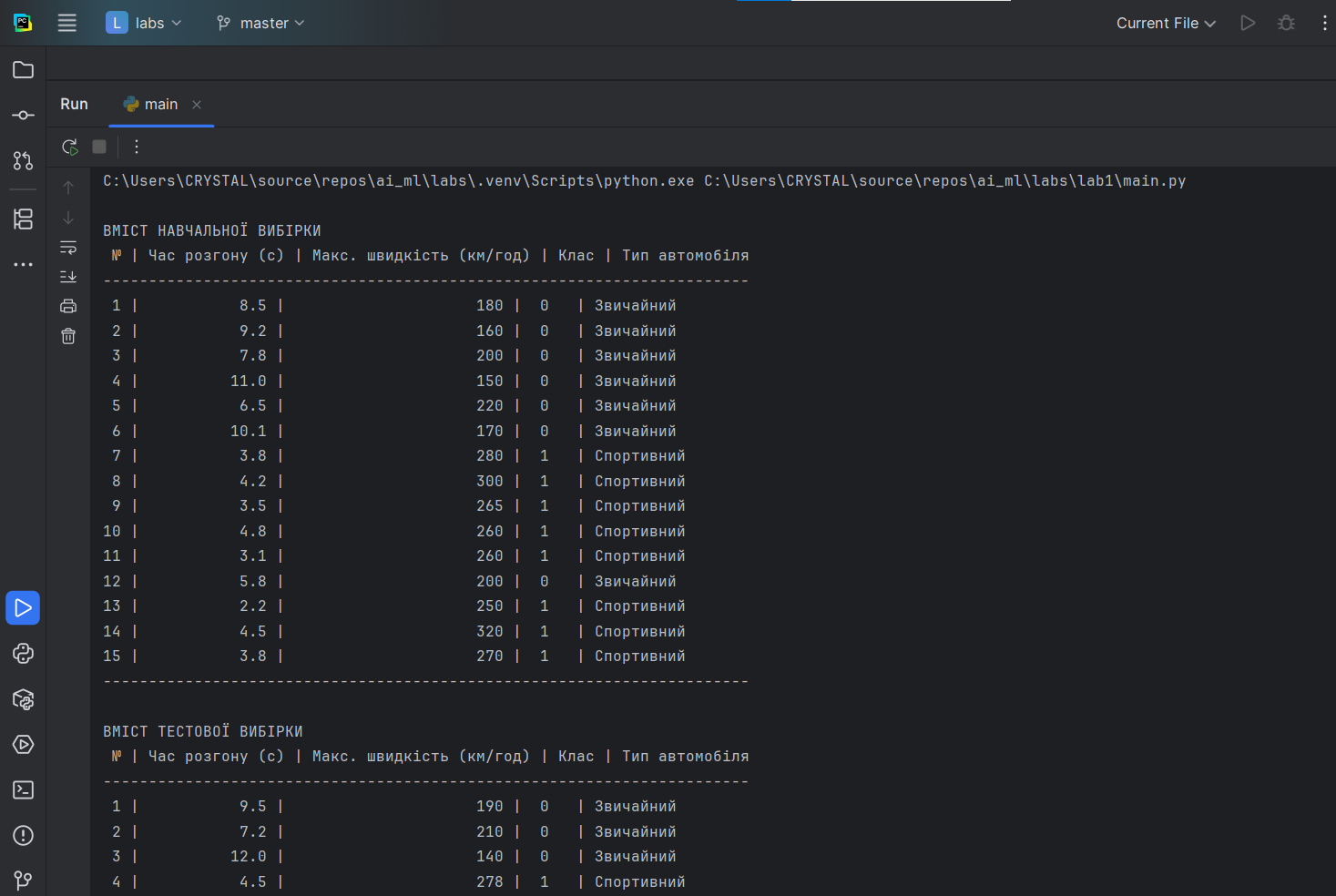
Точність на тестовій вибірці - 100%.

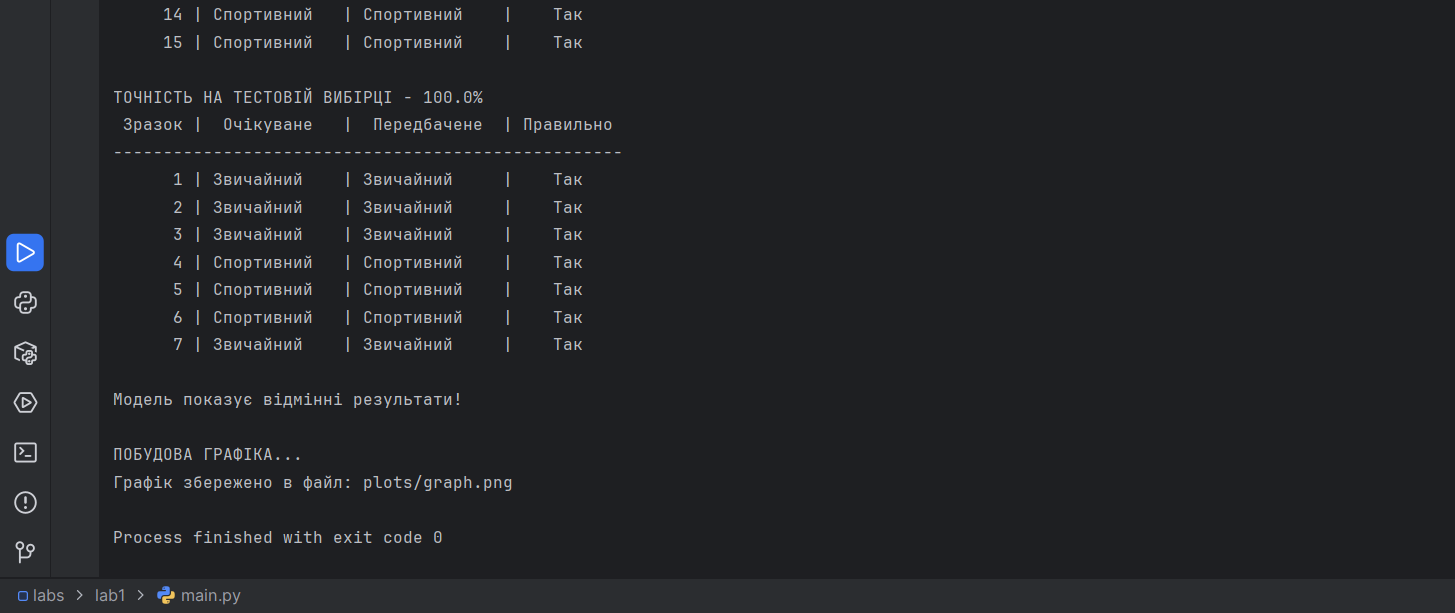
|  |  |  |  |
| --- | --- | --- | --- |
| Зразок | Очікуване | Передбачене | Правильно |
| 1 | Звичайний | Звичайний | Так |
| 2 | Звичайний | Звичайний | Так |
| 3 | Звичайний | Звичайний | Так |
| 4 | Спортивний | Спортивний | Так |
| 5 | Спортивний | Спортивний | Так |
| 6 | Спортивний | Спортивний | Так |
| 7 | Звичайний | Звичайний | Так |

АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

Модель показує відмінні результати. У навчальній вибірці модель показала 100% точність, що означає що усі приклади класифіковані правильно. Теж саме стосується і тестової вибірки. Це стало можливим, бо класи добре відокремлюються за заданими ознаками.

8. ЕКРАННІ ФОРМИ З РЕЗУЛЬТАТАМИ РОБОТИ ПРОГРАМИ





9. ЛІСТИНГ ПРОГРАМИ

perceptron.py

def train(self, training\_data, training\_labels, max\_epochs=1000):  
 epoch = 0  
  
 while epoch < max\_epochs:  
 total\_error = 0  
  
 for i, inputs in enumerate(training\_data):  
 prediction = self.predict(inputs)  
 expected = training\_labels[i]  
 error = expected - prediction  
 total\_error += abs(error)  
  
 if error != 0:  
 inputs\_with\_bias = np.append(inputs, 1)  
  
 for j in range(len(self.weights)):  
 self.weights[j] += self.learning\_rate \* error \* inputs\_with\_bias[j]  
  
 self.training\_errors.append(total\_error)  
  
 if total\_error == 0:  
 print(f"Навчання завершено на епосі {epoch + 1}!")  
 break  
  
 epoch += 1  
  
 if epoch == max\_epochs:  
 print(f"Навчання завершено після {max\_epochs} епох!")

Завдання (додаткове).

10. МОДИФІКАЦІЯ АЛГОРИТМУ НАВЧАННЯ

Метод 1. Поперемінна подача об’єктів різних класів.

При цьому підході дані подаються на вхід перцептрона поперемінно - спочатку зразок класу 0, потім класу 1, і так далі. Це забезпечує більш збалансоване навчання, особливо коли кількість зразків у класах відрізняється.

Результати:

* точність на навчальній вибірці: 100%;
* точність на тестовій вибірці: 100%.

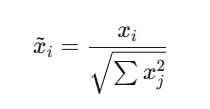
Метод 2. Випадкова вибірка з навчальної вибірки.

У цьому методі зразки вибираються випадково з навчальної вибірки. Кількість епох збільшена в 10 разів (до 10000) для компенсації випадковості подачі даних.

Результати:

* точність на навчальній вибірці: 100 %;
* точність на тестовій вибірці: 100 %.

Метод нормування. L2-нормалізація (відповідно до варіанту 15).



НОРМОВАНІ НАБОРИ ДАНИХ

Навчальна вибірка (L2-нормалізовані дані).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Час розгону (с) | Макс. швидкість (км/год) | Клас | Тип автомобіля |
| 1 | 8.5 | 180 | 1 | Звичайний |
| 2 | 9.2 | 160 | 1 | Звичайний |
| 3 | 7.8 | 200 | 1 | Звичайний |
| 4 | 11.0 | 150 | 1 | Звичайний |
| 5 | 6.5 | 220 | 1 | Звичайний |
| 6 | 10.1 | 170 | 1 | Спортивний |
| 7 | 3.8 | 280 | 2 | Спортивний |
| 8 | 4.2 | 300 | 2 | Спортивний |
| 9 | 3.5 | 265 | 2 | Спортивний |
| 10 | 4.8 | 260 | 2 | Спортивний |
| 11 | 3.1 | 260 | 2 | Спортивний |
| 12 | 5.8 | 200 | 1 | Звичайний |
| 13 | 2.2 | 250 | 2 | Спортивний |
| 14 | 4.5 | 320 | 2 | Спортивний |
| 15 | 3.8 | 270 | 2 | Спортивний |

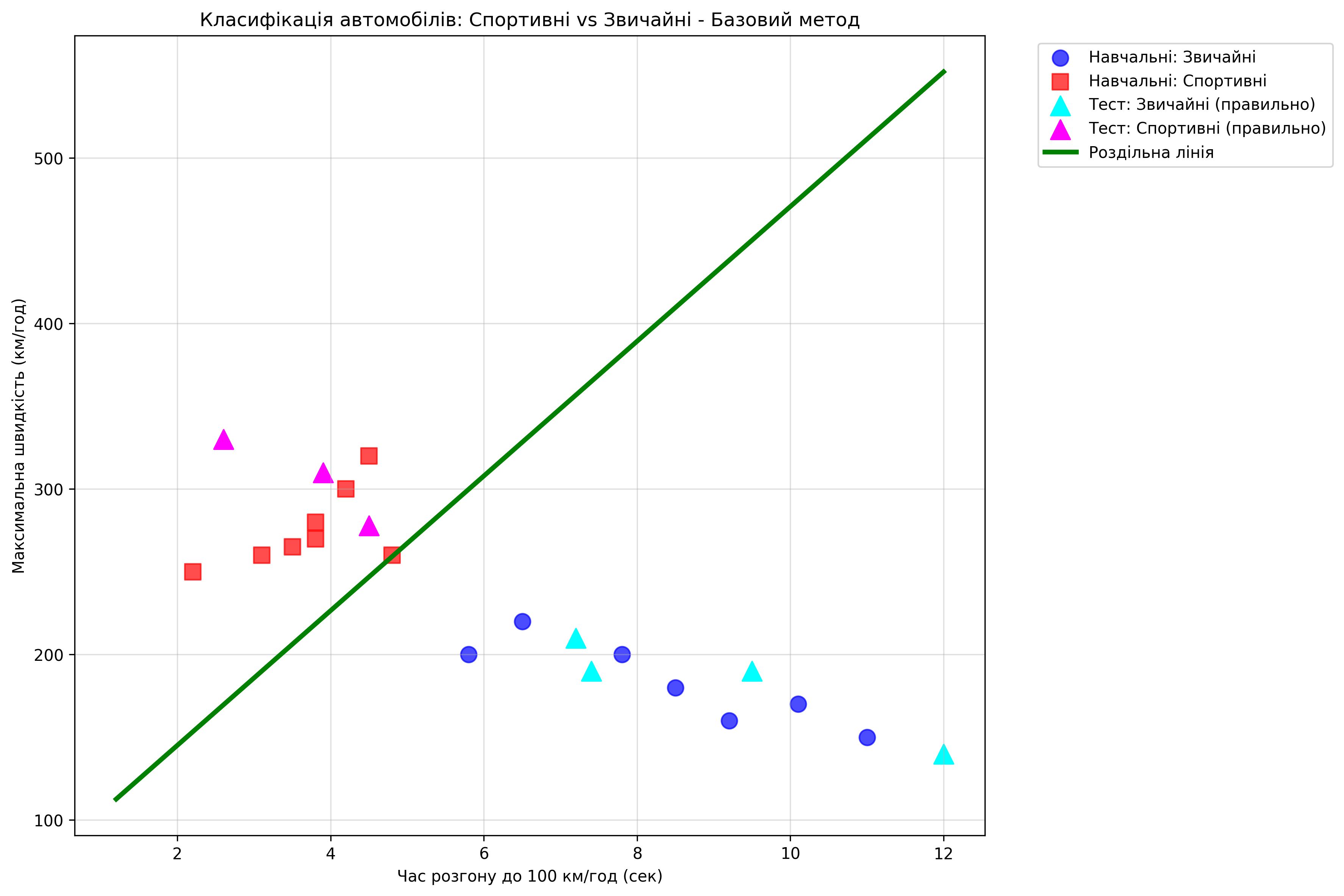
Тестова вибірка (L2-нормалізовані дані).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Час розгону (с) | Макс. швидкість (км/год) | Клас | Тип автомобіля |
| 1 | 9.5 | 190 | 1 | Звичайний |
| 2 | 7.2 | 210 | 1 | Звичайний |
| 3 | 12.0 | 140 | 1 | Звичайний |
| 4 | 4.5 | 278 | 2 | Спортивний |
| 5 | 3.9 | 310 | 2 | Спортивний |
| 6 | 2.6 | 330 | 2 | Спортивний |
| 7 | 7.4 | 190 | 1 | Звичайний |

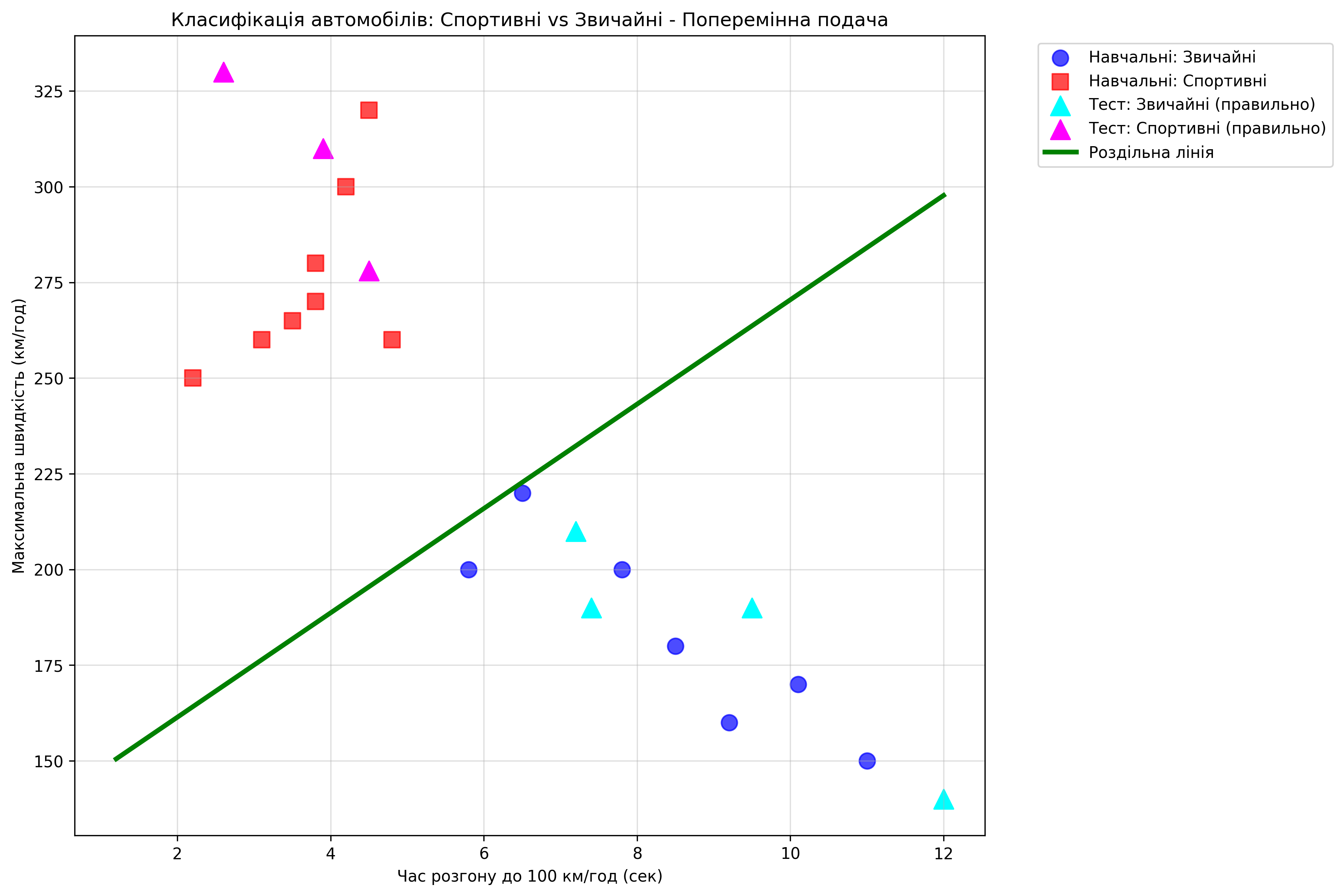
Результати з L2-нормалізацією:

* точність на навчальній вибірці: 100 %;
* точність на тестовій вибірці: 100 %.

Базовий метод



Поперемінна подача



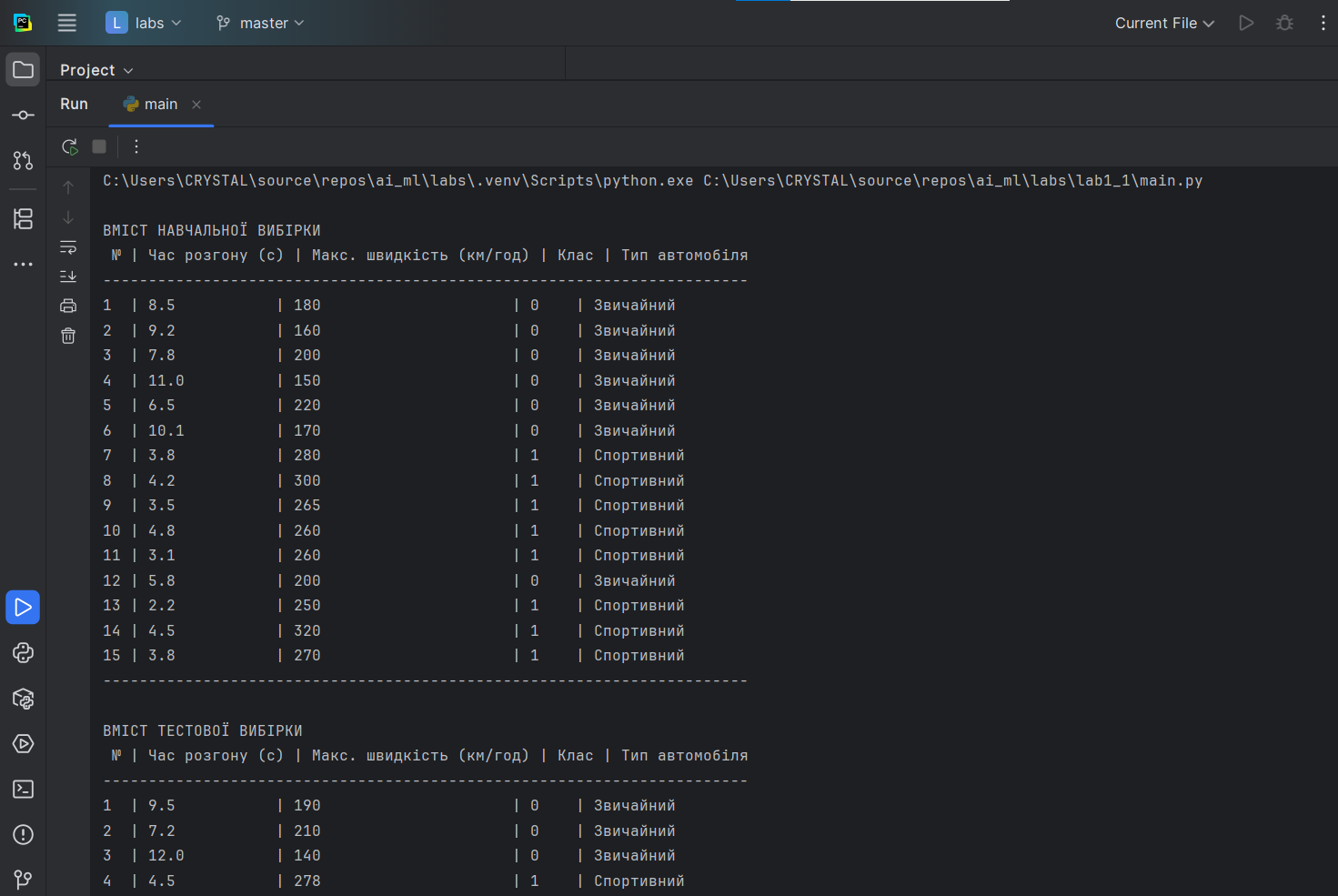
ПОРІВНЯЛЬНА ТАБЛИЦЯ З ОТРИМАНИМИ ПОКАЗНИКАМИ ТОЧНОСТІ ДЛЯ ВСІХ РЕЗУЛЬТАТІВ МОДЕЛЮВАННЯ

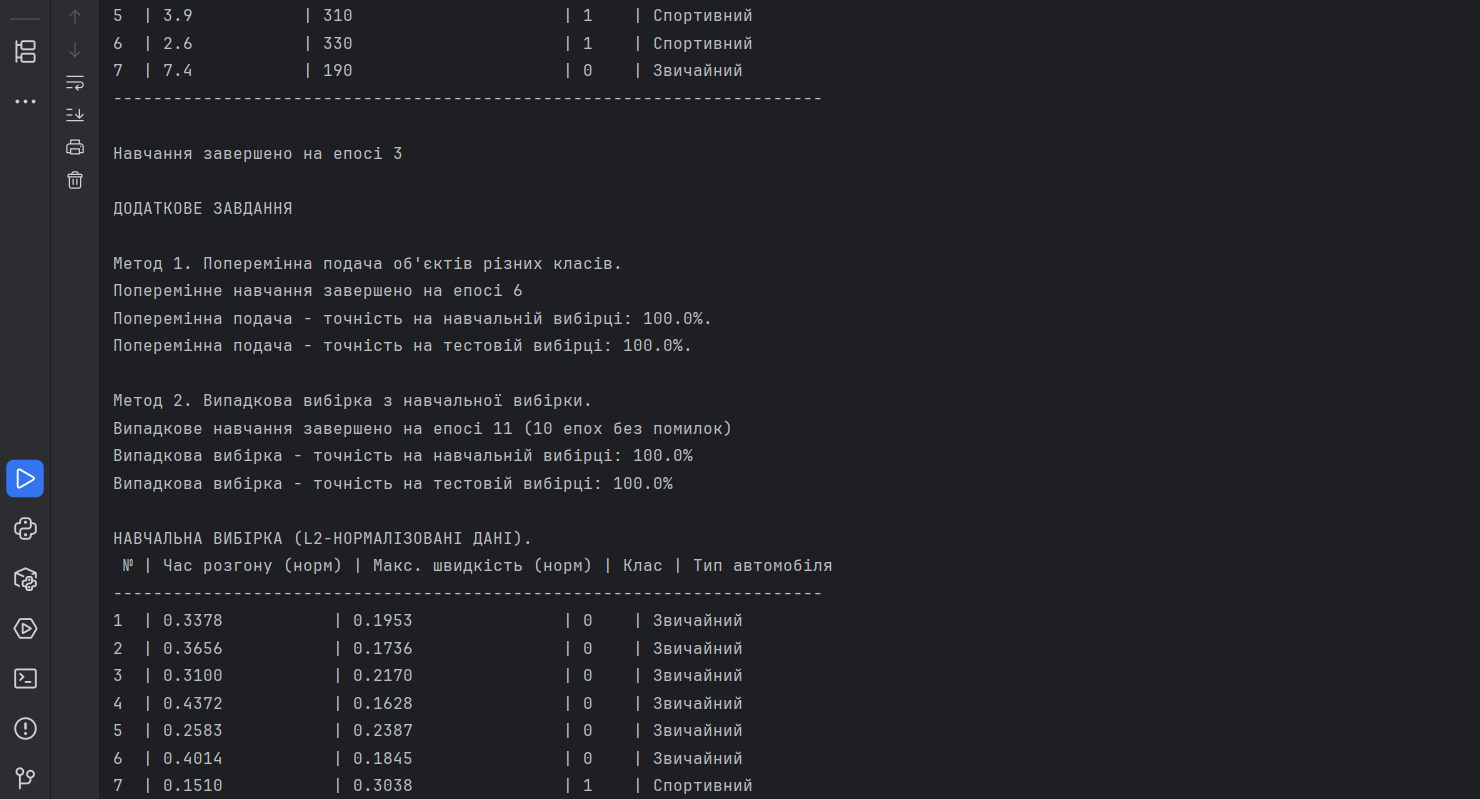
|  |  |  |  |
| --- | --- | --- | --- |
| Метод навчання | Нормалізація | Точність (навчальна) | Точність (тестова) |
| Послідовна подача | Min-Max | 100% | 100% |
| Поперемінна подача | Min-Max | 100% | 100% |
| Випадкова вибірка | Min-Max | 100% | 100% |
| Базовий | L2 | 100% | 100% |

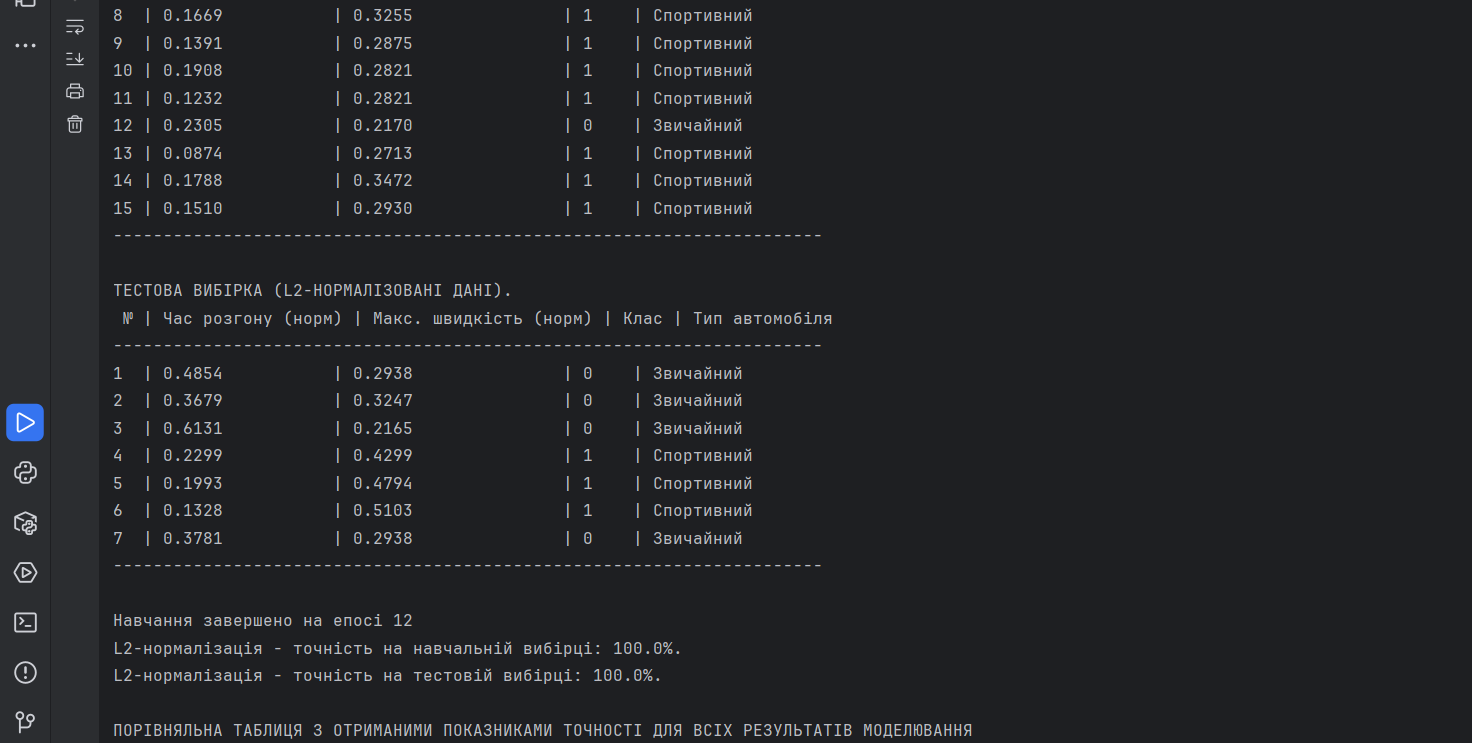
АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

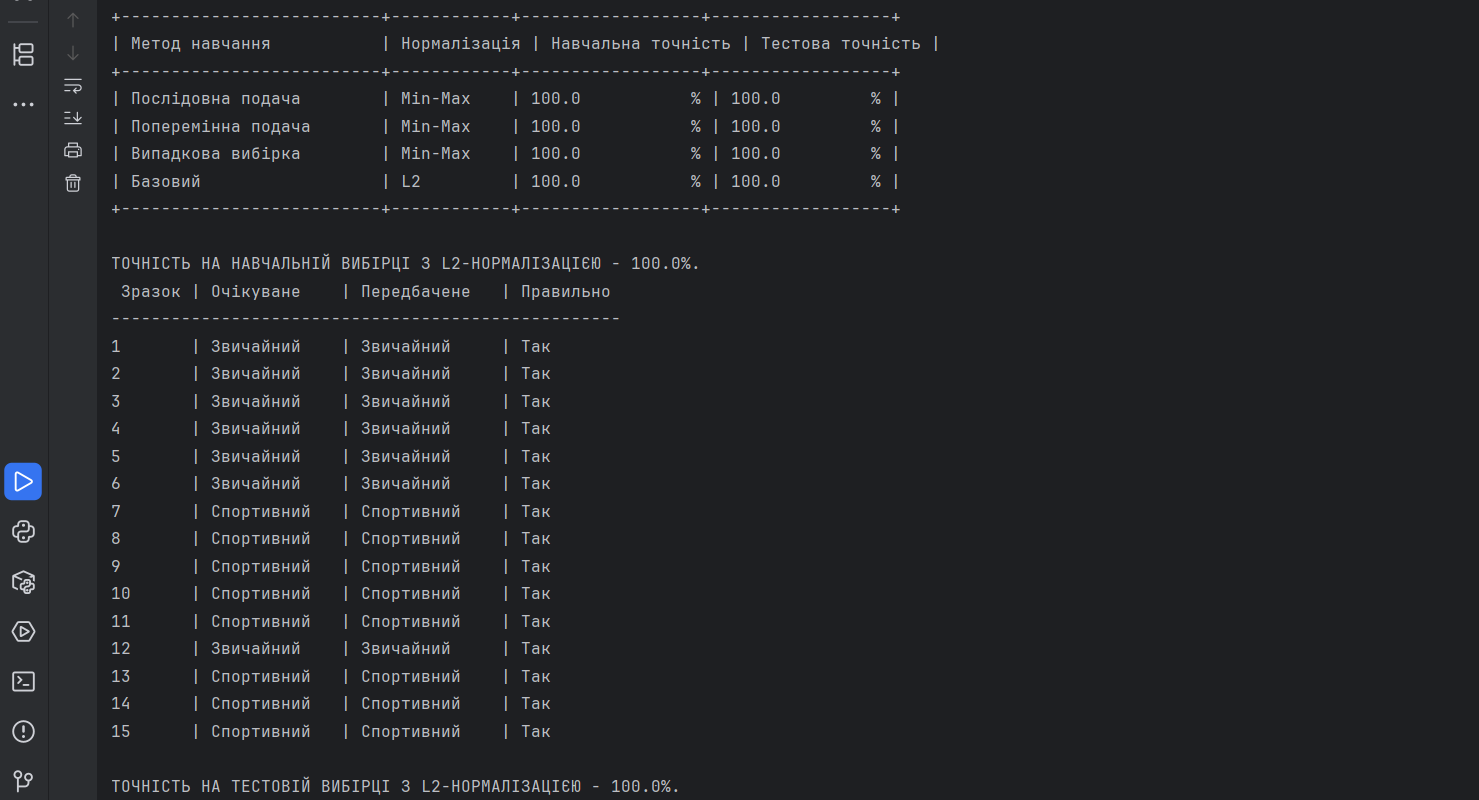
Всі методи показали відмінні результати з точністю 100%. А це свідчить про те, що задача класифікації автомобілів за обраними ознаками є лінійно відокремлюваною. Точність залишилась стабільною незалежно від способу подачі даних чи нормалізації, що підтверджує надійність перцептрона. L2-нормалізація та випадкова вибірка не дали помітного ефекту.  
Висока точність зумовлена лінійною відокремлюваністю класів і спортивні авто легко відрізнити за часом розгону та максимальною швидкістю.

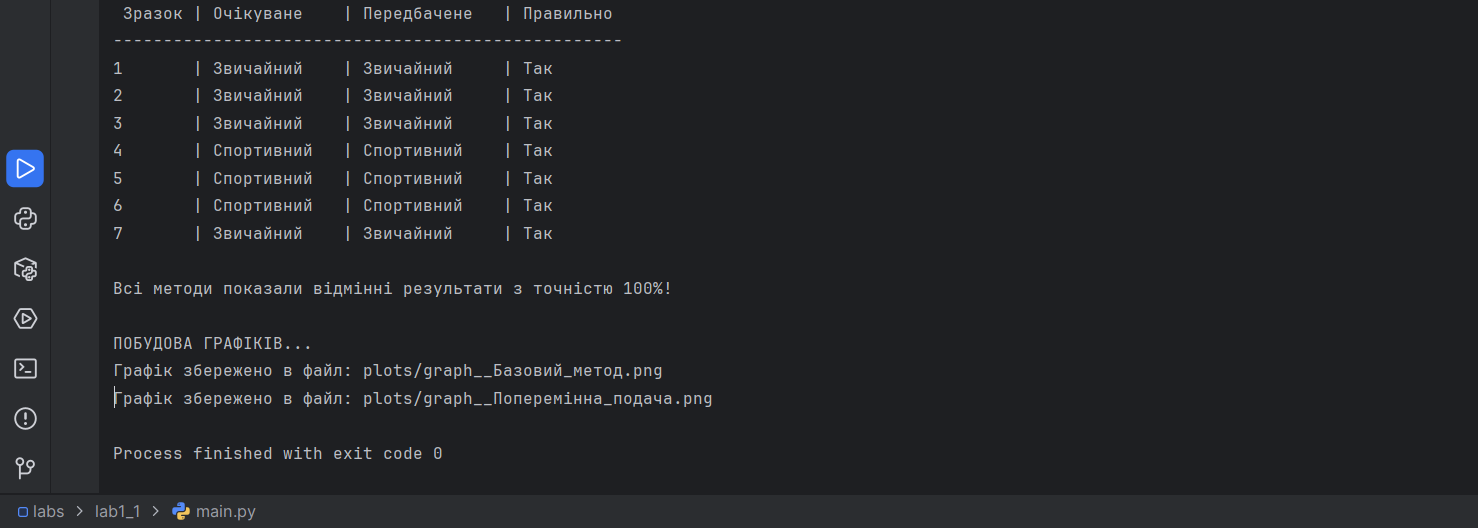
ЕКРАННІ ФОРМИ З РЕЗУЛЬТАТАМИ РОБОТИ ПРОГРАМИ











ЛІСТИНГ ПРОГРАМИ

main.py

def normalize\_minmax(data):  
 min\_vals = data.min(axis=0)  
 max\_vals = data.max(axis=0)  
 return (data - min\_vals) / (max\_vals - min\_vals), min\_vals, max\_vals  
  
def normalize\_l2(data):  
 norms = np.sqrt(np.sum(data \*\* 2, axis=0))  
 return data / norms

perceptron.py

class PerceptronAlternating(Perceptron):  
 def \_\_init\_\_(self, input\_size):  
 super().\_\_init\_\_(input\_size)  
  
 def train\_alternating(self, training\_data, training\_labels, max\_epochs=1000):  
 class0\_indices = np.where(training\_labels == 0)[0]  
 class1\_indices = np.where(training\_labels == 1)[0]  
  
 max\_class\_size = max(len(class0\_indices), len(class1\_indices))  
  
 if len(class0\_indices) < max\_class\_size:  
 class0\_indices = np.pad(class0\_indices,  
 (0, max\_class\_size - len(class0\_indices)),  
 mode='wrap')  
 if len(class1\_indices) < max\_class\_size:  
 class1\_indices = np.pad(class1\_indices,  
 (0, max\_class\_size - len(class1\_indices)),  
 mode='wrap')  
  
 alternating\_indices = []  
 for i in range(max\_class\_size):  
 alternating\_indices.append(class0\_indices[i])  
 alternating\_indices.append(class1\_indices[i])  
  
 epoch = 0  
 while epoch < max\_epochs:  
 total\_error = 0  
 for idx in alternating\_indices:  
 inputs = training\_data[idx]  
 expected = training\_labels[idx]  
 prediction = self.predict(inputs)  
 error = expected - prediction  
 total\_error += abs(error)  
 if error != 0:  
 inputs\_with\_bias = np.append(inputs, 1)  
 for j in range(len(self.weights)):  
 self.weights[j] += self.learning\_rate \* error \* inputs\_with\_bias[j]  
 self.training\_errors.append(total\_error)  
 if total\_error == 0:  
 print(f"Поперемінне навчання завершено на епосі {epoch + 1}")  
 break  
 epoch += 1  
 if epoch == max\_epochs:  
 print(f"Поперемінне навчання завершено після {max\_epochs} епох")  
  
  
class PerceptronRandom(Perceptron):  
 def \_\_init\_\_(self, input\_size):  
 super().\_\_init\_\_(input\_size)  
  
 def train\_random(self, training\_data, training\_labels, max\_epochs=10000):  
 n\_samples = len(training\_data)  
 epoch = 0  
 consecutive\_zero\_errors = 0  
  
 while epoch < max\_epochs:  
 total\_error = 0  
 indices = np.random.choice(n\_samples, n\_samples, replace=True)  
  
 for idx in indices:  
 inputs = training\_data[idx]  
 expected = training\_labels[idx]  
 prediction = self.predict(inputs)  
 error = expected - prediction  
 total\_error += abs(error)  
 if error != 0:  
 inputs\_with\_bias = np.append(inputs, 1)  
 for j in range(len(self.weights)):  
 self.weights[j] += self.learning\_rate \* error \* inputs\_with\_bias[j]  
  
 self.training\_errors.append(total\_error)  
  
 if total\_error == 0:  
 consecutive\_zero\_errors += 1  
 if consecutive\_zero\_errors >= 10:  
 print(f"Випадкове навчання завершено на епосі {epoch + 1} (10 епох без помилок)")  
 break  
 else:  
 consecutive\_zero\_errors = 0  
  
 epoch += 1  
  
 if epoch == max\_epochs:  
 print(f"Випадкове навчання завершено після {max\_epochs} епох")

ВИСНОВКИ

У ході цієї лабораторної роботи було успішно розроблено та навчено одношаровий перцептрон для класифікації автомобілів на "звичайні" та "спортивні" за двома ознаками: часом розгону та максимальною швидкістю. Модель продемонструвала 100% точність як на навчальній, так і на тестовій вибірках, що свідчить про високу ефективність для даної задачі.

Ключовим фактором такого результату є те, що класи є

лінійно відокремлюваними, тобто їх можна легко розділити прямою лінією на графіку. Були протестовані додаткові методи навчання, такі як поперемінна подача зразків, випадкова вибірка та L2-нормалізація даних. Жодна з цих модифікацій не змінила кінцевої точності, що ще раз підтверджує надійність простого перцептрона для цього завдання.

Таким чином, робота доводить, що навіть найпростіші нейронні архітектури можуть бути надзвичайно ефективними, якщо дані мають чітку структуру.