МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені Тараса Шевченка ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра програмних систем і технологій

Дисципліна
«Спеціалізоване програмування автоматизованих систем»

Лабораторна робота № 2

«Алгоритми класифікації»

Виконав:	Гоша Давід	**Перевірив**:	
Група	ІПЗ-33	Дата перевірки	
Форма навчання	денна	Оцінка	
Спеціальність	121		
2022			

Завдання:

Розробіть класифікатор за варіантом. Підготуйте тренувальний (навчальний) та тестовий набори даних для класифікатора. Дані можна взяти з файлу iris.csv та розділити на дві частини, де 90% віднести до тренувального (навчального) набору, а 10% до тестового. Навчіть класифікатор на тренувальному наборі даних. Застосуйте класифікатор до тестового набору даних і проаналізуйте точність результату – визначте відсоток правильних передбачень серед тестових даних.

Варіант 4

Хід роботи

Засуемо основнні функції програми:

1. Завантаження набору даних райдужної оболонки ока з файлу iris.csv.
2. Розділяє дані на навчальний і тестовий набори у співвідношенні 90/10.
3. Визначає функцію для обчислення евклідової відстані між двома точками.
4. Визначає функцію для отримання k найближчих сусідів тестової точки в навчальній вибірці.
5. Визначає функцію для прогнозування класу тестової точки на основі її k найближчих сусідів.
6. Використовує навчальну множину для прогнозування на тестовій множині та оцінює точність класифікатора.
7. Виводить точність класифікатора.

Спочатку ми завантажили набір даних ірисів з файлу iris.csv і розділили його на навчальний і тестовий набори у співвідношенні 90/10. Потім ми визначили MNN-класифікатор, використовуючи евклідову відстань з k=1, і навчили його на навчальній вибірці. Далі ми використали навчений класифікатор для прогнозування на тестовому наборі та оцінили точність класифікатора, порівнявши його прогнози з істинними мітками в тестовому наборі.

Код:

```
import csv
import random
import math

# Load the iris dataset from the iris.csv file
with open('iris.csv', 'r') as f:
    reader = csv.reader(f)
    next(reader)  # skip the first row (column names)
    iris_data = [row for row in reader]
```

```
# Split the data into a training set and a test set using a 90/10 split
random.shuffle(iris_data)
split_index = math.floor(len(iris_data) * 0.9)
train_data = iris_data[:split_index]
test_data = iris_data[split_index:]

# Define a function to calculate the Euclidean distance between two points
def euclidean_distance(point1, point2):
    distance = 0
    for i in range(len(point1)-1):
        distance += (float(point1[i]) - float(point2[i])) ** 2
    return math.sqrt(distance)

# Define a function to get the k nearest neighbors of a test point in the training set
def get_nearest_neighbors(train_data, test_point, k):
    distances = []
    for train_point in train_data:
        distance = euclidean_distance(train_point, test_point)
        distances.append((train_point, distance))
    distances.sort(key=lambda x: x[1])
    neighbors = [x[0] for x in distances[:k]]
    return neighbors

# Define a function to predict the class of a test point based on its k nearest neighbors
def predict_class(train_data, test_point, k):
    neighbors = get_nearest_neighbors(train_data, test_point, k)
    class_votes = {}
    for neighbor in neighbors:
        label = neighbor[-1]
        if label in class_votes:
            class_votes[label] += 1
        else:
            class_votes[label] = 1
    sorted_votes = sorted(class_votes.items(), key=lambda x: x[1], reverse=True)
    return sorted_votes[0][0]

# Use the training set to make predictions on the test set and evaluate the accuracy of the classifier
correct_predictions = 0
for test_point in test_data:
    predicted_class = predict_class(train_data, test_point, 1)
    if predicted_class == test_point[-1]:
        correct_predictions += 1

accuracy = correct_predictions / len(test_data)
print("Accuracy:", accuracy)
```

Результати:

MNN-класифікатор досяг 100% точності на тестовому наборі, що означає, що він правильно передбачив клас усіх зразків у тестовому наборі. Це свідчить про те, що MNN-класифікатор здатен ефективно розділяти різні класи в наборі даних Iris на основі значень їхніх ознак.

Accuracy: 1.0

Висновок:

MNN-класифікатор, що використовує евклідову відстань, є простим, але ефективним алгоритмом для задач класифікації. У випадку з набором даних ірисів MNN-класифікатор зміг досягти ідеальної точності на тестовому наборі, що свідчить про те, що алгоритм зміг ефективно розділити різні класи квітів на основі їхніх значень ознак. Однак важливо зазначити, що набір даних ірисів є відносно невеликим і добре керованим, і продуктивність MNN-класифікатора може відрізнятися на більших і складніших наборах даних. Тим не менш, MNN класифікатор є корисним інструментом для задач класифікації і може бути використаний як базовий алгоритм для більш складних моделей.