UBND THÀNH PHỐ HỒ CHÍ MINH

Trường Đại Học Sài Gòn

BÁO CÁO THỰC HÀNH: K NEAREST NEIGHBORS

Họ Tên	MSSV
Lê Phước Thành	3123580045
Nguyễn Hoàng Long	3123580022
Đường Minh Đức	3123580010
Hà Tuấn Duy	3123580006

Thầy Hướng Dẫn: Đỗ Như Tài

THÀNH PHỐ HỒ CHÍ MINH

Mục Lục

1. Giới Thiệu Mô hình K-Nearest Neighbors (KNN)

Đây là lớp **K_Nearest_Neighbors** cốt lõi (từ file **K_nearest_model.py**) được sử dụng để giải quyết cho cả hai câu hỏi bài tập.

```python
import numpy as np
from collections import Counter

class K_Nearest_Neighbors:
    def __init__(self, data_set, k):
        self.ds = data_set
        self.k = k
        self.confidence = 0 # Thêm thuộc tính này cho nhất quán

    def predict(self, feature_set):
        distances = []
        for group in self.ds:
            for feature in self.ds[group]:
                # Tính khoảng cách euclidean
                e_d = np.linalg.norm(np.array(feature) - np.array(feature_set))
                distances.append([e_d, group])
```

```python
# Lấy tên lớp (group) của các hàng xóm
votes = [d[1] for d in nearest]

# Xử lý trường hợp không tìm thấy (ví dụ k=0 hoặc list rỗng)
if not votes:
    print("Cảnh báo: Không có 'votes' nào. Trả về None.")
    return None # Hoặc một giá trị mặc định

# Đếm số phiếu bầu và chọn lớp có nhiều phiếu nhất
nearest_group = Counter(votes).most_common(1)[0]
feature_set_group, self.confidence = nearest_group[0], nearest_group[1] / self.k

return feature_set_group
```

```python
def test(self, test_data):
    correct = 0
    total = 0
    for group in test_data:
        for feature_set in test_data[group]:
            group_prediction = self.predict(feature_set)
            if group_prediction == group:
                correct += 1
            total += 1

    if total > 0:
        accuracy = correct / total
        # Cập nhật print để dễ đọc hơn
        print(f"Total: {total}, Correct: {correct}")
        print(f"Accuracy = {accuracy * 100:.2f}%")
    else:
        print("Không có dữ liệu trong test_data.")
```

2.Giải Quyết Câu 1: Phân loại hoa Iris

2.1 Mục Tiêu

Cài đặt một chương trình demo thuật toán KNN để phân loại 3 loài hoa (Iris-setosa, Iris-versicolor, Iris-virginica) dựa trên tập dữ liệu Iris.

2.2 Code Thực Thi

Đoạn code này sẽ tải dữ liệu, xử lý và sử dụng lớp K_Nearest_Neighbors đã định nghĩa ở trên để chạy phân loại.

File code: **k_nearest_cau1.ipynb**

2.3 Giải Thích

- **Tải và xử lý dữ liệu**: Code sử dụng thư viện pandas để đọc file Iris.csv. Cột 'Id' được loại bỏ vì không phải là đặc trưng. Dữ liệu sau đó được chuyển thành dạng list và xáo trộn ngẫu nhiên.
- **Chia dữ liệu**: Dữ liệu được chia thành 2 phần: 120 mẫu đầu tiên (80%) dùng để huấn luyện (train_data) và 30 mẫu còn lại (20%) dùng để kiểm tra (test_data).
- **Định dạng dữ liệu**: Dữ liệu được chuyển đổi thành định dạng dictionary mà lớp KNN yêu cầu (ví dụ: {'Iris-setosa': [[5.1, 3.5, 1.4, 0.2], ...], ...}).
- **Huấn luyện và Kiểm tra**: Mô hình KNN được khởi tạo với k=7. Hàm knn.test() được gọi để lặp qua 30 mẫu trong test_data, dự đoán lớp cho từng mẫu và so sánh với nhãn thực tế để tính toán độ chính xác cuối cùng.

2.4 Kết Quả

Kết quả của chương trình sẽ in ra độ chính xác của mô hình trên tập 30 mẫu kiểm tra là Accuracy = 0.9666666666666667. (Kết quả có thể thay đổi một chút sau mỗi lần chạy do dữ liệu được xáo trộn ngẫu nhiên).

3.Giải Quyết Câu 2: Nhận Dạng ký tự

3.1 Mục Tiêu

Áp dụng thuật toán KNN để nhận dạng ký tự (26 lớp A-Z) từ bộ dữ liệu UCI Letter-Recognition, bao gồm 20.000 mẫu với 16-D feature vectors.

3.2 Code Thực Thi

Đoạn code này sẽ tải dữ liệu, xử lý và sử dụng lớp K_Nearest_Neighbors đã định nghĩa ở trên để chạy phân loại.

File code: **k_nearest.ipynb**

3.3 Giải Thích

- **Tải và xử lý dữ liệu**: Code sử dụng pandas để đọc file letter-recognition.data. Vì file này không có header, ta cung cấp một danh sách features_name để đặt tên cho các cột, trong đó cột đầu tiên lettr là nhãn (ký tự A-Z).
- **Định dạng dữ liệu**: Tương tự như Câu 1, nhưng vì dữ liệu ban đầu ở dạng list [['A', ...], ['B', ...]], một hàm trợ giúp format_data_for_knn được tạo ra để chuyển đổi nó về định dạng dictionary mà lớp KNN yêu cầu.
- **Chia dữ liệu**: Toàn bộ 20.000 mẫu được xáo trộn. Sau đó, chúng được chia theo tỷ lệ 80/20, tương ứng với 16.000 mẫu cho huấn luyện và 4.000 mẫu cho kiểm tra.
- **Huấn luyện và Kiểm tra**: Mô hình KNN được khởi tạo với k=7. Hàm knn.test() được gọi để chạy dự đoán trên 4.000 mẫu kiểm tra.

3.4 Kết Quả

Kết quả thực tế thu được từ file Jupyter Notebook của bạn cho thấy độ chính xác rất cao:

- Total: 4000, Correct: 3812
- Accuracy = 95.30%