Mai Đức Khiêm

Nhóm 11

CODE: HỆ THỐNG PHÂN LOẠI RƯỢU VANG

# Bước 1: Chuẩn bị dữ liệu

# Đọc dữ liệu từ file CSV chứa thông tin về rượu vang

url = "./wine.data" # Đường dẫn tới file dữ liệu

# Đặt tên cho các cột trong dữ liệu

columns = ['Class', 'Alcohol', 'Malic\_Acid', 'Ash', 'Alcalinity\_of\_Ash',

'Magnesium', 'Total\_Phenols', 'Flavanoids', 'Nonflavanoid\_Phenols',

'Proanthocyanins', 'Color\_Intensity', 'Hue',

'OD280\_OD315\_of\_Diluted\_Wines', 'Proline']

# Đọc dữ liệu vào dataframe

data = pd.read\_csv(url, header=None, names=columns)

print(data.head()) # Hiển thị 5 dòng đầu tiên của dữ liệu

# Bước 2: Chia dữ liệu thành tập huấn luyện và kiểm tra

# Định nghĩa hàm để phân chia dữ liệu

def split\_data(data):

# Tạo dataframe rỗng cho tập huấn luyện và kiểm tra

train\_data = pd.DataFrame()

test\_data = pd.DataFrame()

# Lặp qua từng lớp dữ liệu

for label in data['Class'].unique():

class\_data = data[data['Class'] == label] # Lọc dữ liệu theo từng lớp

test\_sample = class\_data.sample(n=1) # Lấy ngẫu nhiên 1 mẫu từ mỗi lớp làm tập kiểm tra

train\_samples = class\_data.drop(test\_sample.index) # Phần còn lại làm tập huấn luyện

# Kết hợp các mẫu huấn luyện và kiểm tra

test\_data = pd.concat([test\_data, test\_sample])

train\_data = pd.concat([train\_data, train\_samples])

return train\_data, test\_data # Trả về tập huấn luyện và tập kiểm tra

# Gọi hàm để chia dữ liệu

# Bước 2: Chia dữ liệu thành tập huấn luyện và kiểm tra với tỷ lệ 70-30

#train\_data, test\_data = train\_test\_split(data, test\_size=0.3, stratify=data['Class'], random\_state=42)

train\_data, test\_data = split\_data(data)

# Bước 3: Chuẩn hóa dữ liệu

# Chuẩn hóa dữ liệu giúp đưa các đặc trưng về cùng một đơn vị đo để mô hình học hiệu quả hơn

scaler = StandardScaler() # Khởi tạo bộ chuẩn hóa

# Chuẩn hóa các đặc trưng của tập huấn luyện

train\_features = scaler.fit\_transform(train\_data.drop('Class', axis=1))

# Chuẩn hóa tập kiểm tra dựa trên thông số của tập huấn luyện

test\_features = scaler.transform(test\_data.drop('Class', axis=1))

# Bước 4: Xây dựng mô hình Template Matching

# Định nghĩa hàm phân loại bằng phương pháp template matching

def classify\_template\_matching(train\_features, train\_labels, test\_features):

results = [] # Tạo danh sách lưu kết quả dự đoán

# Lặp qua từng mẫu kiểm tra

for test\_sample in test\_features:

# Tính khoảng cách Euclid giữa mẫu kiểm tra và tất cả các mẫu huấn luyện

distances = euclidean\_distances([test\_sample], train\_features)

# Tìm mẫu huấn luyện gần nhất

nearest\_index = distances.argmin()

# Dự đoán lớp của mẫu kiểm tra dựa trên mẫu gần nhất

predicted\_class = train\_labels.iloc[nearest\_index]

# Thêm dự đoán vào danh sách kết quả

results.append(predicted\_class)

return results # Trả về danh sách các kết quả dự đoán

# Tách nhãn lớp từ tập huấn luyện và tập kiểm tra

train\_labels = train\_data['Class']

test\_labels = test\_data['Class']

# Gọi hàm để phân loại và lưu kết quả vào predictions

predictions = classify\_template\_matching(train\_features, train\_labels, test\_features)

# Bước 5: Đánh giá mô hình

# Tính số lượng dự đoán chính xác

correct = sum(true == pred for true, pred in zip(test\_labels, predictions))

# Tính độ chính xác của mô hình

accuracy = correct / len(test\_labels)

print(f"Accuracy: {accuracy:.2f}") # In ra độ chính xác

# Nhận xét kết quả

for true, pred in zip(test\_labels, predictions):

print(f"Actual: {true}, Predicted: {pred}") # In ra kết quả từng mẫu

# Nhận xét kết quả

# Phương án 1: Hiển thị toàn bộ kết quả

results\_df = pd.DataFrame({

'Actual': test\_labels,

'Predicted': predictions

})

print(results\_df) # In ra toàn bộ kết quả

# Phương án 2: Hiển thị chỉ 10 kết quả đầu tiên

print(results\_df.head(10)) # In ra 10 dòng đầu tiên