Bài tập Thực hành môn Khai phá Dữ liệu

Họ và tên: Huỳnh Nguyễn Thế Dân

MSSV: 21110256

Lớp: 21TTH1

Cài đặt thuật toán K_medians

Định nghĩa hàm MDE

```
def MDE(a, b):
    # Hàm MDE nhận vào hai tham số, a và b.
    return np.sum(np.abs(a - b))  # Trả về tổng của giá trị tuyệt đối của hiệu giữa các phần tử tương ứng của a và b.
```

Tóm tắt thuật toán:

- **Chức năng của hàm**: Hàm MDE tính toán khoảng cách Manhattan giữa hai mảng (hoặc hai vector) a và b. Khoảng cách Manhattan, còn được biết đến với tên gọi khoảng cách L1, là tổng các giá trị tuyệt đối của hiệu giữa các phần tử tương ứng của hai mảng.
- **Cách thức hoạt động**:
  - a - b: Tính hiệu giữa từng cặp phần tử tương ứng của hai mảng a và b.
  - np.abs(...): Lấy giá trị tuyệt đối của từng phần tử trong mảng kết quả từ bước trước.
  - np.sum(...): Tính tổng của các giá trị tuyệt đối vừa thu được, đây chính là giá trị của khoảng cách Manhattan.

Định nghĩa hàm k_median_centroid_assignation

```
def k_median_centroid_assignation(dset, centroids):
    '''
        Given a dataframe 'dset' and a set of 'centroids', we assign each
        data point in 'dset' to a centroid.
        - dset - pandas dataframe with observations
        - centroids - pandas dataframe with centroids
    '''
    # Số lượng centroids
    k = centroids.shape[0]
    # Số lượng quan sát trong dset
```

```
    n = dset.shape[0]
    # Danh sách để lưu trữ thông tin về centroid gần nhất cho mỗi quan
sát
    assignation = []
    # Danh sách để lưu trữ khoảng cách từ mỗi quan sát đến centroid gần
nhất của nó
    assign_errors = []

    # Duyệt qua mỗi quan sát trong dset
    for obs in range(n):
        # Khởi tạo mảng để lưu trữ các khoảng cách từ quan sát hiện tại
đến mỗi centroid
        all_errors = np.array([])
        # Duyệt qua mỗi centroid
        for centroid in range(k):
            # Tính khoảng cách Manhattan từ centroid này đến quan sát
hiện tại
            error = MDE(centroids.iloc[centroid,:], dset.iloc[obs,:])
            # Lưu trữ khoảng cách này vào mảng all_errors
            all_errors = np.append(all_errors, error)

        # Xác định centroid có khoảng cách nhỏ nhất đến quan sát hiện
tại
        nearest_centroid = np.where(all_errors == np.amin(all_errors))
[0].tolist()[0]
        # Khoảng cách nhỏ nhất từ quan sát hiện tại đến centroid
        nearest_centroid_error = np.amin(all_errors)

        # Thêm centroid gần nhất và khoảng cách của nó vào các danh
sách tương ứng
        assignation.append(nearest_centroid)
        assign_errors.append(nearest_centroid_error)
    return assignation, assign_errors
```

Tóm tắt thuật toán:

Hàm này được sử dụng trong bối cảnh phân cụm, nơi bạn có một tập hợp dữ liệu và một số centroids đã được xác định. Mục đích của hàm là gán mỗi điểm dữ liệu trong tập hợp dset tới centroid gần nhất dựa trên khoảng cách Manhattan, được tính toán bởi hàm MDE.

- **Duyệt Qua Dữ Liệu**: Hàm duyệt qua mỗi quan sát trong dataframe dset.
- **Tính Khoảng Cách**: Với mỗi quan sát, hàm tính toán khoảng cách từ quan sát đến từng centroid trong centroids.
- **Chọn Centroid Gần Nhất**: Sau khi tính toán, hàm xác định centroid có khoảng cách nhỏ nhất đến quan sát đó.
- **Lưu Kết Quả**: Kết quả được lưu trữ dưới dạng danh sách các centroids gần nhất và khoảng cách tới các centroids đó từ mỗi quan sát.

Hàm trả về hai danh sách: một danh sách các chỉ số của centroids gần nhất cho mỗi quan sát và một danh sách các khoảng cách từ mỗi quan sát đến centroid gần nhất của nó. Hàm này hữu ích trong việc thiết lập các bước đầu tiên của thuật toán K-Medians hoặc các thuật toán phân cụm tương tự.

Định nghĩa hàm k_median_centroid_assignation

```
def Kmedians(dset, k=2, tol=1e-4):
    '''
        K-medians implementation for a
        'dset': DataFrame with observations
        'k': Number of cluster, default k=2
        'tol': Tolerance = 1E-4
    '''


    # Tạo một bản sao của tập dữ liệu để tránh chỉnh sửa trực tiếp lên
dữ liệu gốc
    working_dset = dset.copy()
    # Khởi tạo các biến để lưu trữ lỗi, tín hiệu dừng và đếm số lần lặp
    error = []
    goAHead = True
    iteration = 0

    # Bước 2: Khởi tạo các cluster bằng cách xác định centroids
    centroids = initiate_centroids(k, dset)

    # Vòng lặp chính của thuật toán
    while (goAHead):
        # Bước 3 và 4: Gán centroids và tính toán lỗi
        working_dset['centroid'], iteration_error =
k_median_centroid_assignation(working_dset, centroids)
        error.append(sum(iteration_error))

        # Bước 5: Cập nhật vị trí của các centroids
        centroids =
working_dset.groupby('centroid').agg('median').reset_index(drop=True)

        # Bước 6: Khởi động lại vòng lặp
        if iteration > 0:
            # Kiểm tra nếu lỗi giảm ít hơn ngưỡng cho phép (1E-4)
            if error[iteration-1] - error[iteration] <= tol:
                goAHead = False
        iteration += 1

    # Cuối cùng, thực hiện một lần gán centroid và tính toán lỗi
    working_dset['centroid'], iteration_error =
k_median_centroid_assignation(working_dset, centroids)
    centroids =
working_dset.groupby('centroid').agg('median').reset_index(drop=True)
    return working_dset['centroid'], iteration_error, centroids
```

Tóm tắt thuật toán:

Hàm Kmedians là một cài đặt của thuật toán phân cụm K-medians, sử dụng khoảng cách Manhattan để gán mỗi điểm dữ liệu vào một trong số k centroids. Thuật toán lặp lại quá

trình này cho đến khi sự thay đổi trong tổng lỗi giữa các lần lặp nhỏ hơn một giá trị ngưỡng nhất định (tol), cho thấy sự ổn định của các centroids.

- **Khởi tạo và chuẩn bị**: Hàm bắt đầu bằng cách tạo một bản sao của tập dữ liệu để tránh sửa đổi dữ liệu gốc và khởi tạo các biến cần thiết cho quá trình lặp.
- **Xác định Centroids ban đầu**: Hàm gọi initiate_centroids để tạo các centroids ban đầu.
- **Phân Centroids và Tính toán Lỗi**: Sử dụng hàm k_median_centroid_assignation, mỗi điểm dữ liệu được gán cho centroid gần nhất, và lỗi được tính toán dựa trên khoảng cách từ điểm đến centroid của nó.
- **Cập nhật Centroids**: Centroids được cập nhật bằng cách tính trung vị của các điểm dữ liệu đã được gán cho từng centroid.
- **Kiểm tra Điều kiện Dừng**: Nếu sự thay đổi trong lỗi nhỏ hơn ngưỡng cho phép, quá trình lặp sẽ dừng lại.
- **Trả về kết quả**: Cuối cùng, hàm trả về danh sách centroids, các lỗi tương ứng và các nhóm (centroid gán cho mỗi điểm).

Hàm này hữu ích cho việc phân tích dữ liệu không giám sát, giúp phát hiện các nhóm tự nhiên trong dữ liệu dựa trên các tính năng được cung cấp.

Hết.