KHAI THÁC DỮ LIỆU

Nguyễn Xuân Việt Đức - 22280012

Bài tập thực hành - Lần 6

Bài 6: SỰ PHÂN TÍCH NHÓM

III. Nội dung thực hành:

1. Thuật Toán K-means

K-means là một thuật toán phân cụm lặp, nhằm chia n quan sát thành k cụm sao cho mỗi quan sát thuộc về cụm có trung bình gần nhất (tâm cụm).

Các bước:

- **Khởi tạo:** Chọn ngẫu nhiên k điểm dữ liệu từ tập dữ liệu làm các tâm cụm ban đầu.
- **Gán cụm:** Gán mỗi điểm dữ liệu vào tâm cụm gần nhất. Khoảng cách thường dùng là khoảng cách Euclid và sai số được tính là tổng bình phương sai số (SSE) từ mỗi điểm đến tâm cụm của nó.
- **Cập nhật:** Tính lại các tâm cụm bằng trung bình của tất cả các điểm dữ liệu trong cụm đó.
- **Lặp:** Lặp lại hai bước trên cho đến khi tâm cụm không còn thay đổi đáng kể (nghĩa là sai số tổng thay đổi dưới một ngưỡng xác định) hoặc đạt đến số lần lặp tối đa.

Hàm sai số: Tổng căn bậc hai sai số bình phương:

```
rsserr(a, b) = np.sum(np.square(a - b))
```

Listing 1: Hàm sai số

Implementation:

1.1. Khởi tạo ngẫu nhiên centroid:

```
def initiate_centroids(k, dset):
    '''
    Select k data points as centroids
    k: number of centroids
    dset: pandas dataframe
    '''
    centroids = dset.sample(k)
    return centroids

np.random.seed(42)
k = 3
df = blobs[['x', 'y']]
centroids = initiate_centroids(k, df)
print(centroids)
```

	x	y
0	24.412	32.932
5	25.893	31.515
36	26.878	36.609

1.2. Hàm tính sai số:

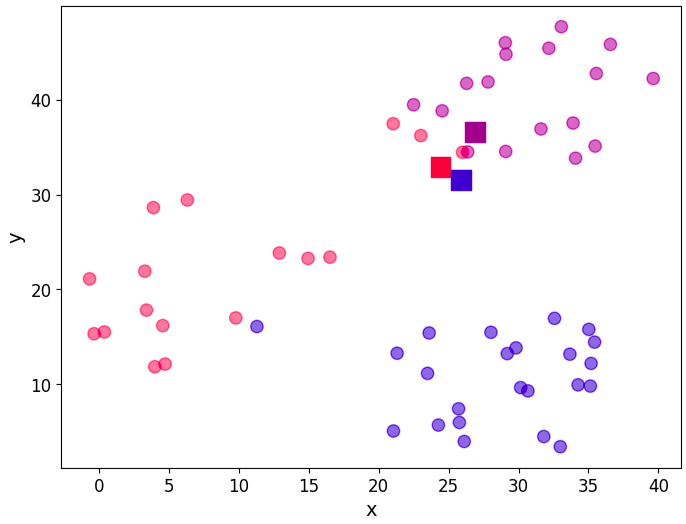
```
def rsserr(a, b):
    '''
    Calculate the root of sum of squared errors.
    a and b are numpy arrays
    '''
    return np.sum(np.square(a - b))

for i, centroid in enumerate(range(centroids.shape[0])):
    err = rsserr(centroids.iloc[centroid, :], df.iloc[36, :])
    print('Error for centroid {0}: {1:.2f}'.format(i, err))
```

```
Error for centroid 0: 19.60
Error for centroid 1: 26.92
Error for centroid 2: 0.00
```

1.3. Hàm gán tâm cụm:

```
def centroid_assignation(dset, centroids):

    k = centroids.shape[0]
    n = dset.shape[0]
    assignation = []
    assign_errors = []

    for obs in range(n):
        all_errors = np.array([])
        for centroid in range(k):
            err = rsserr(centroids.iloc[centroid, :], dset.iloc[obs
    , :])
            all_errors = np.append(all_errors, err)

        nearest_centroid = np.where(all_errors == np.amin(
    all_errors))[0].tolist()[0]
        nearest_centroid_error = np.amin(all_errors)

        assignation.append(nearest_centroid)
        assign_errors.append(nearest_centroid_error)

    return assignation, assign_errors

df.loc[:, 'centroid'], df.loc[:, 'error'] = centroid_assignation(df
    , centroids)
print(df.head())
```

Đồ thị phân tán cùng với cụm ngẫu nhiên:




```
print("The total error is {0:.2f}".format(df['error'].sum()))
```

```
The total error is 20606.95
```

1.4. Hàm tình K-means:

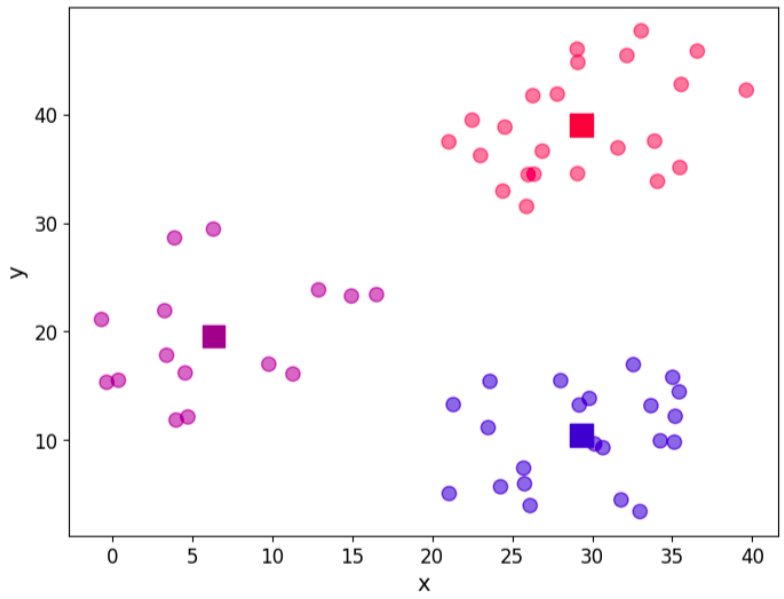
```
def kmeans(dset, k=2, tol=1e-4):
    '''
    K-means implementation for a
    'dset': DataFrame with observations
    'k': number of clusters, default k=2
    'tol': tolerance=1E-4
    '''
    # Let us work in a copy, so we don't mess the original
    working_dset = dset.copy()
    # We define some variables to hold the error, the
    # stopping signal and a counter for the iterations
    err = []
    goahead = True
    j = 0

    # Step 2: Initiate clusters by defining centroids
    centroids = initiate_centroids(k, dset)

    while(goahead):
        # Step 3 and 4 - Assign centroids and calculate error
        working_dset['centroid'], j_err = centroid_assignation(
    working_dset, centroids)
        err.append(sum(j_err))

        # Step 5 - Update centroid position
        centroids = working_dset.groupby('centroid').agg('mean').
    reset_index(drop=True)

        # Step 6 - Restart the iteration
        if j > 0:
            # Is the error less than a tolerance (1E-4)
            if err[j-1] - err[j] <= tol:
                goahead = False

        j += 1

    return working_dset['centroid'], j_err, centroids
```

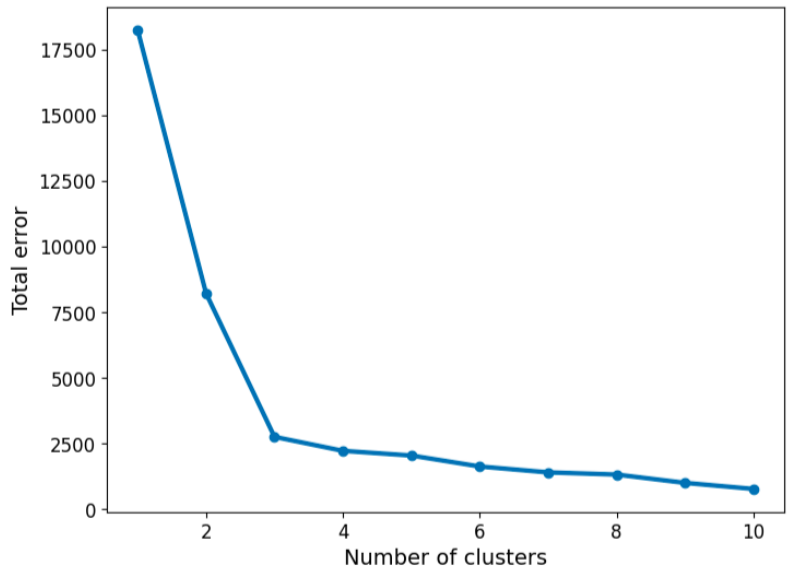
```
np.random.seed(432)
df['centroid'], df['error'], centroids = kmeans(df[['x', 'y']], 3)
print(df.head())
```

	x	y	centroid	error
0	24.412	32.932	0	61.380524
1	35.190	12.189	1	37.415091
2	26.288	41.718	0	16.216075
3	0.376	15.506	2	51.798518
4	26.116	3.963	1	52.188602

	x	y
0	29.304957	39.050783
1	29.330864	10.432409
2	6.322867	19.559800

1.5. Kết quả sau áp dụng K-means Biểu đồ khủy tay của k-means:





Từ biểu đồ trên ta thấy được khủy tay được xác định tại vị trí số lượng cụm là $3 => k = 3$, điều này hòan toàn hợp lý với dữ liệu ban đầu.

2. Thuật Toán K-medians

K-medians tương tự K-means nhưng khác ở cách cập nhật tâm cụm và cách tính khoảng cách (và sai số). Thuật toán này tối thiểu hóa tổng khoảng cách từ các điểm đến trung vị của cụm. Nó thường bền vững hơn với các giá trị ngoại lai so với K-means.

Các bước:

- **Khởi tạo:** Chọn ngẫu nhiên k điểm dữ liệu làm tâm cụm ban đầu (giống phương pháp của K-means).
- **Gán cụm:** Gán mỗi điểm vào tâm cụm gần nhất. Khoảng cách dùng là khoảng cách Manhattan (Tổng độ lệch tuyệt đối - SAD).
- **Cập nhật:** Tính lại các tâm cụm bằng trung vị của tất cả các điểm trong cụm.
- **Lặp:** Lặp lại quá trình gán cụm và cập nhật cho đến khi các tâm cụm không còn thay đổi đáng kể hoặc đạt đến số lần lặp tối đa.

Hàm sai số: Tổng độ lệch tuyệt đối:

```
sad_err(a, b) = np.sum(np.abs(a - b))
```

Listing 2: Hàm sai số

Implementation:

2.1. Hàm tính sai số cho K-medians:

```
def sad_err(a, b):
    '''
    Calculate the sum of absolute differences (Manhattan distance).
    a and b are numpy arrays or pandas series
    '''
    return np.sum(np.abs(a - b))
```

2.2. Hàm gắn tâm cụm cho thuật toán K-medians:

```

def centroid_assignation_median(dset, centroids):
    '''
    Given a dataframe `dset` and a set of `centroids`, we assign
    each
    data point in `dset` to a centroid based on Sum of Absolute
    Differences.
    - dset: pandas dataframe with observations
    - centroids: pandas dataframe with centroids
    '''
    k = centroids.shape[0]
    n = dset.shape[0]
    assignation = []
    assign_errors = []

```

```
    for obs_idx in range(n):
        # Estimate error
        all_errors = np.array([])
        for centroid_idx in range(k):
            err = sad_err(centroids.iloc[centroid_idx, :], dset.
    iloc[obs_idx, :])
            all_errors = np.append(all_errors, err)

        # Get the nearest centroid and the error
        nearest_centroid = np.where(all_errors == np.amin(
    all_errors))[0].tolist()[0]
        nearest_centroid_error = np.amin(all_errors)

        # Add values to corresponding lists
        assignation.append(nearest_centroid)
        assign_errors.append(nearest_centroid_error)

    return assignation, assign_errors
```

2.3. Hàm thuật toán K-medians:

```
def kmedians(dset, k=2, tol=1e-4):
    '''
    K-medians implementation for a
    `dset`: DataFrame with observations
    `k`: number of clusters, default k=2
    `tol`: tolerance=1E-4
    '''
    working_dset = dset.copy()
    err = []
    goahead = True
    j = 0
    centroids = initiate_centroids(k, dset)

    while(goahead):
        working_dset['centroid'], j_err =
    centroid_assignation_median(working_dset, centroids)
        err.append(sum(j_err))

        centroids = working_dset.groupby('centroid').median().
    reset_index(drop=True)


        if j > 0:
            if err[j-1] - err[j] <= tol:
                goahead = False
        j += 1

    return working_dset['centroid'], j_err, centroids

np.random.seed(432)
df_kmedians = blobs[['x', 'y']].copy()
df_kmedians['centroid'], df_kmedians['error'], centroids_kmedians =
     kmedians(df_kmedians[['x', 'y']], 3)
print("Data points with assigned centroids (K-medians):")
print(df_kmedians.head())
print("\nFinal centroids (K-medians):")
print(centroids_kmedians)
```

2.4. Kết quả

Data points with assigned centroids (K-medians):

	x	y	centroid	error
0	24.412	32.932	0	9.2470
1	35.190	12.189	1	6.8565
2	26.288	41.718	0	6.9510
3	0.376	15.506	2	6.4780
4	26.116	3.963	1	10.4435

Final centroids (K-medians):

	x	y
0	29.056	37.5350
1	29.986	10.5365
2	4.550	17.8100