Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
Московский государственный технический университет имени Н.Э. Баумана

Домашняя работа №2
«Кластерный анализ методом k-средних»
по курсу:
«Моделирование»

Выполнил:
студент группы ИУ9-82
Иванов Георгий

Проверила:
Домрачева А.Б.

Москва, 2019

Содержание

1 Цель задачи

Имеются данные с информацией о ВИЧ-зараженных в различных странах за временной промежуток с 2000 года по 2014 год.

Необходимо исследовать связь между оценкой страны по ВИЧ (для каждой страны оценка вычисляется в зависимости от населения, уровня страны и количество зараженных) и количеством ВИЧ-зараженных с помощью кластерного анализа.

2 Теоретические сведения

2.1 Кластерный анализ (кластеризация)

Кластерный анализ — многомерная статистическая процедура, выполняющая сбор данных, содержащих информацию о выборке объектов, и затем упорядочивающая объекты в сравнительно однородные группы. Внутри каждой группы должны оказаться «похожие» объекты, а объекты разных группы должны быть как можно более отличны.

2.2 Метод k-средних

Метод k-средних наиболее популярный метод кластеризации. Целью этого метода является разделение m наблюдений (из пространства R^n) на k кластеров ($k \leq m$), при этом каждое наблюдение относится к тому кластеру, к центру (центроиду) которого оно ближе всего. В качестве меры близости используется Евклидово расстояние.

Основная идея заключается в том, что на каждой итерации перевычисляется центр масс для каждого кластера, полученного на предыдущем шаге, затем векторы разбиваются на кластеры вновь в соответствии с тем, какой из новых центров оказался ближе по выбранной метрике.
Алгоритм завершается, когда на какой-то итерации не происходит изменения внутрикластерного расстояния. Это происходит за конечное число итераций, так как количество возможных разбиений конечного множества конечно, а на каждом шаге суммарное квадратичное отклонение V уменьшается, поэтому зацикливание невозможно.

3 Практическая реализация

```
from collections import defaultdict
from random import uniform
from math import sqrt
from matplotlib import pyplot as plt

import pandas as pd
import random


def point_avg(points):
    """
    Accepts a list of points, each with the same number of dimensions.
    NB. points can have more dimensions than 2

    Returns a new point which is the center of all the points.
    """
    dimensions = len(points[0])

    new_center = []

    for dimension in range(dimensions):
        dim_sum = 0  # dimension sum
        for p in points:
            dim_sum += p[dimension]

        # average of each dimension
        new_center.append(dim_sum / float(len(points)))

    return new_center


def update_centers(data_set, assignments):
    """
    Accepts a dataset and a list of assignments; the indexes
    of both lists correspond to each other.

    Compute the center for each of the assigned groups.

    Return `k` centers where `k` is the number of unique assignments.
    """
    new_means = defaultdict(list)
    centers = []
    for assignment, point in zip(assignments, data_set):
        new_means[assignment].append(point)
```

```
45
46      for points in new_means.values():
47          centers.append(point_avg(points))
48
49      return centers
50
51
52  def assign_points(data_points, centers):
53      """
54      Given a data set and a list of points betweeen other points,
55      assign each point to an index that corresponds to the index
56      of the center point on it's proximity to that point.
57      Return a an array of indexes of centers that correspond to
58      an index in the data set; that is, if there are N points
59      in `data_set` the list we return will have N elements. Also
60      If there are Y points in `centers` there will be Y unique
61      possible values within the returned list.
62      """
63      assignments = []
64      for point in data_points:
65          shortest = float("+inf")  # positive infinity
66          shortest_index = 0
67          for i in range(len(centers)):
68              val = distance(point, centers[i])
69              if val < shortest:
70                  shortest = val
71                  shortest_index = i
72          assignments.append(shortest_index)
73      return assignments
74
75
76  def distance(a, b):
77      """
78      """
79      dimensions = len(a)
80
81      _sum = 0
82      for dimension in range(dimensions):
83          difference_sq = (a[dimension] - b[dimension]) ** 2
84          _sum += difference_sq
85      return sqrt(_sum)
86
87
88  def generate_k(data_set, k):
89      """
90      Given `data_set`, which is an array of arrays,
```

```
    find the minimum and maximum for each coordinate, a range.
    Generate `k` random points between the ranges.
    Return an array of the random points within the ranges.
    """
    centers = []
    dimensions = len(data_set[0])
    min_max = defaultdict(int)

    for point in data_set:
        for i in range(dimensions):
            val = point[i]
            min_key = 'min_%d' % i
            max_key = 'max_%d' % i
            if min_key not in min_max or val < min_max[min_key]:
                min_max[min_key] = val
            if max_key not in min_max or val > min_max[max_key]:
                min_max[max_key] = val

    for _k in range(k):
        rand_point = []
        for i in range(dimensions):
            min_val = min_max['min_%d' % i]
            max_val = min_max['max_%d' % i]

            rand_point.append(uniform(min_val, max_val))

        centers.append(rand_point)

    return centers


def k_means(dataset, k):
    k_points = generate_k(dataset, k)
    assignments = assign_points(dataset, k_points)
    old_assignments = None
    while assignments != old_assignments:
        new_centers = update_centers(dataset, assignments)
        old_assignments = assignments
        assignments = assign_points(dataset, new_centers)

    result = dict()
    for assign_num, point in zip(assignments, dataset):
        if assign_num not in result:
            result[assign_num] = [[point[i]] for i in range(len(point))]
        else:
            for i in range(len(point)):
```

```
                result[assign_num][i].append(point[i])
    return result, new_centers


def visualize_clusters(clusters, information):
    """
    Visualizes the first 2 dimensions of the data as a 2-D scatter plot.
    """
    plt.figure()
    colors = ["#%06x" % random.randint(0, 0xFFFFFF) for _ in range(len(clusters))]
    for num_cluster in clusters:
        plt.plot(clusters[num_cluster][0], clusters[num_cluster][1], 'o',
        ↪  color=colors[num_cluster], markersize=2)

    for num_cluster in clusters:
        for i in range(0, len(clusters[num_cluster][0]), 111):

            point = (clusters[num_cluster][0][i], clusters[num_cluster][1][i])

            for j in range(len(information)):

                if point[0] == information[j][0] and point[1] == information[j][1]:
                    dx = random.randint(-1, 1)
                    dy = random.randint(-1, 1)

                    plt.annotate(information[j][2], xy=(clusters[num_cluster][0][i],
                    ↪  clusters[num_cluster][1][i]),
                                 xytext=(clusters[num_cluster][0][i] + (dx * 40),
                                 ↪  clusters[num_cluster][1][i] + (dy * 6)))

                    break



    plt.show()


fixed_df = pd.read_csv('aids.csv', sep=',', encoding='utf8')


cases = [int(case.replace(",", "")) for case in fixed_df['Cases'].to_list()]
rates = [rate for rate in fixed_df['Rate'].to_list()]
population = [p.replace(",", "").replace('\'',"") for p in
↪  fixed_df['Population'].to_list()]


```

```
clusters, centers = k_means(list(zip(cases, rates)), 2)
visualize_clusters(clusters, list(zip(cases, rates, population)))
```

4 Результаты

Примеры результаты работы программы приведены на рисунках для разного количества кластеров (рис. 1-3).

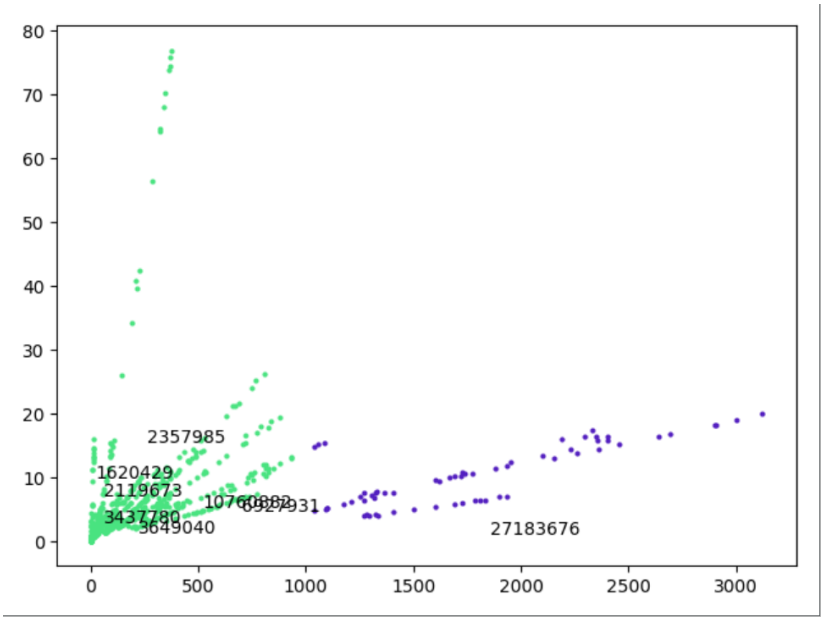

Рис. 1: 2 кластера

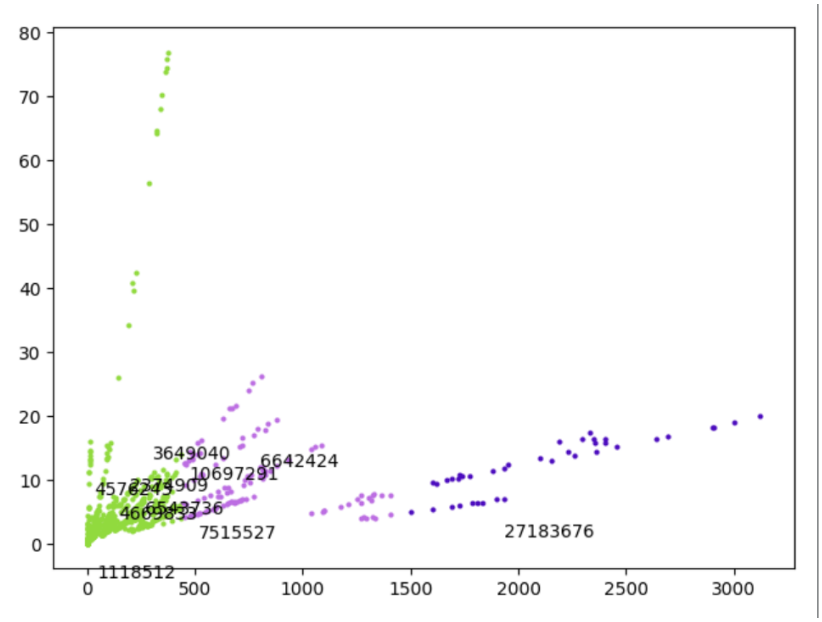

Рис. 2: 3 кластера

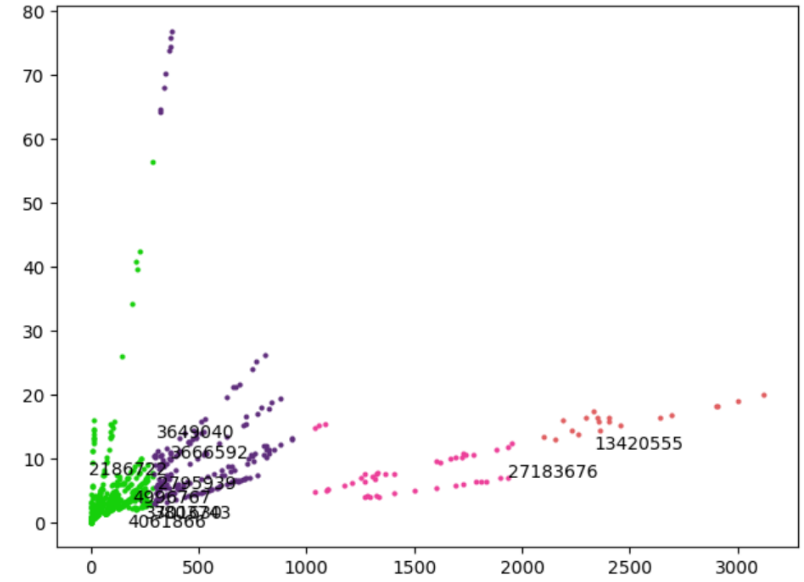

Рис. 3: 4 кластера

5 Вывод

На рисунке 1 более четко можно сформулировать гипотезу о то, что в мире есть тендеция увеличения числа ВИЧ-зараженных, при этом в развивающихся странах число более резко возрастает, нежели чем в развитых странах.