МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра алгоритмической математики

ОТЧЕТ

по практической работе №7

по дисциплине «Статистический анализ»

ТЕМА: КЛАСТЕРНЫЙ АНАЛИЗ. МЕТОД ПОИСКА СГУЩЕНИЙ.

Студент гр. 9372 _____ Иванов Р. С.

Преподаватель _____ Сучков А. И.

Санкт-Петербург

2021

Цель работы.

Освоение основных понятий и некоторых методов кластерного анализа.

Основные теоретические положения

Метод поиска сгущений:

Радиус R лежит в интервале (Rmin, Rmax)

Rmin = min {dij > 0}, Rmax = max {dij}

Радиус выбирается произвольно, среди всех расстояний между точками, не считая наибольшего расстояния (будет один кластер при любом центроиде) и наименьшего (число кластеров всегда равно числу точек при любых центроидах).

Алгоритм:

1. Выбираем случайную точку из множества.

2. Вокруг этой точки раздуваем окружность радиуса R(в многомерном пространстве многомерную сферу).

3. Добавляем в кластер все точки, попавшие в окружность(сферу).

4. Перемещаем центр из первоначально выбранной точки в центр масс получившегося кластера.

5. Если центр масс сместился, переходим к пункту 3.

6. Исключаем получившийся кластер из множества точек.

7. Если множество точек не пустое, переходим на следующую итерацию алгоритма - возвращаемся к пункту 1.

Постановка задачи.

Дано конечное множество из объектов, представленных двумя признаками (в качестве этого множества принимаем исходную двумерную выборку, сформированную ранее в практической работе №4). Выполнить разбиение исходного множества объектов на конечное число подмножеств (кластеров) с использованием метода поиска сгущений. Полученные результаты содержательно проинтерпретировать.

Выполнение работы.

Пункт 1

Нормализовали множество точек по формуле (1)

$$\overline{x}_i = \frac{\overline{x}_i - \overline{x}_в}{\sigma} \quad (1)$$

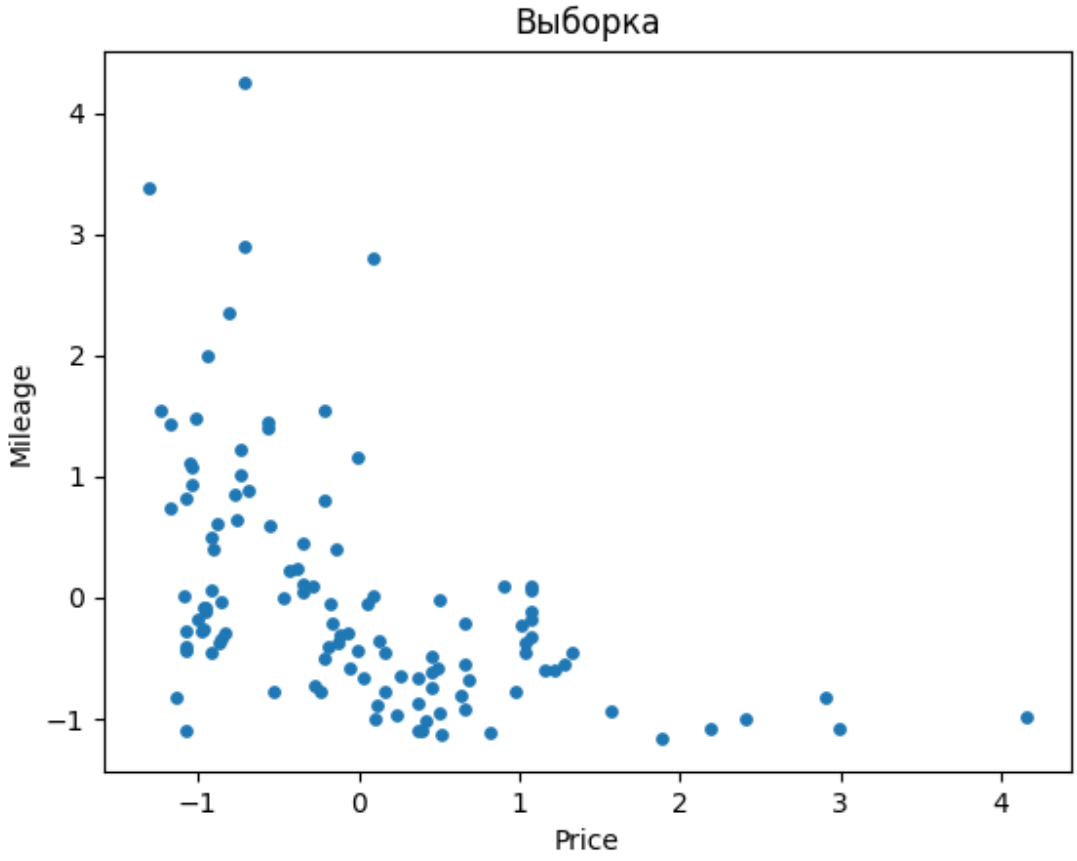

Рисунок 1 (Выборка)

Пункт 2

Реализовали алгоритм поиска сгущений, отобразили полученные кластеры, выделив каждый кластер разным цветом и отметив центроиды звездочкой для $R = 0{,}7$.

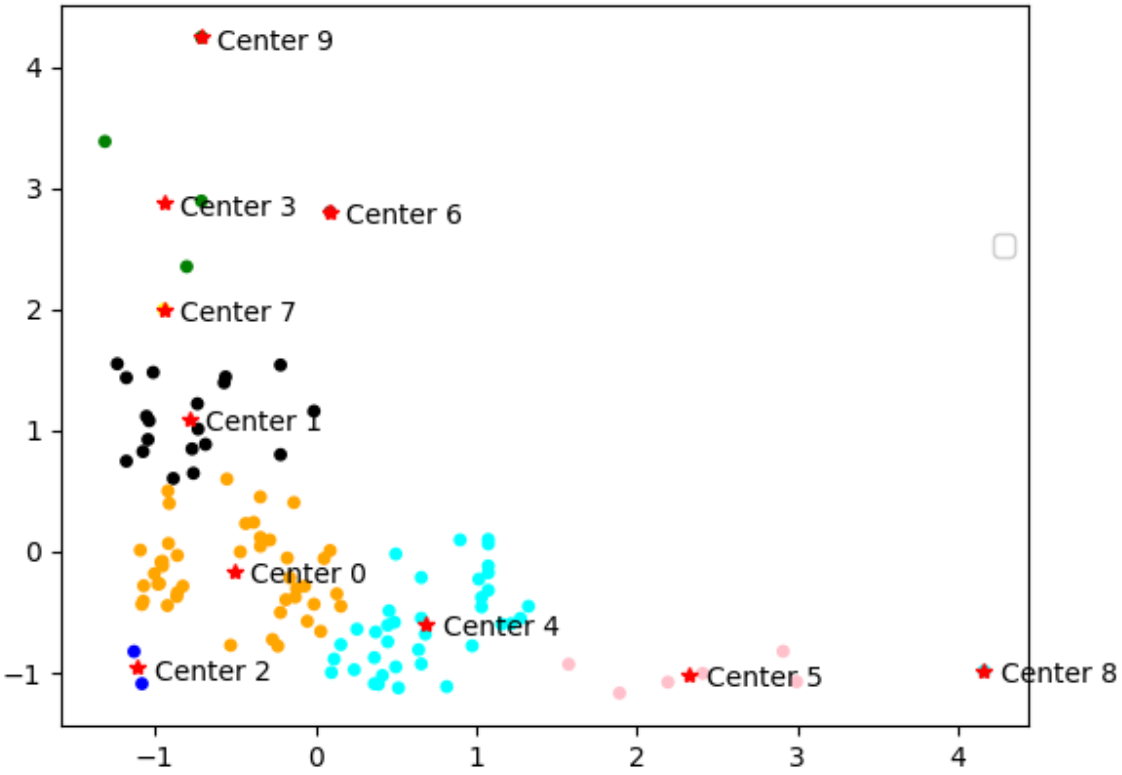

Рисунок 2 (Метод сгущений)

Пункт 3

Для проверки чувствительности метода берём те же начальные центроиды, сгенерированные еще в предыдущем пункте, добавляя погрешность к радиусу и прогоняя алгоритм.

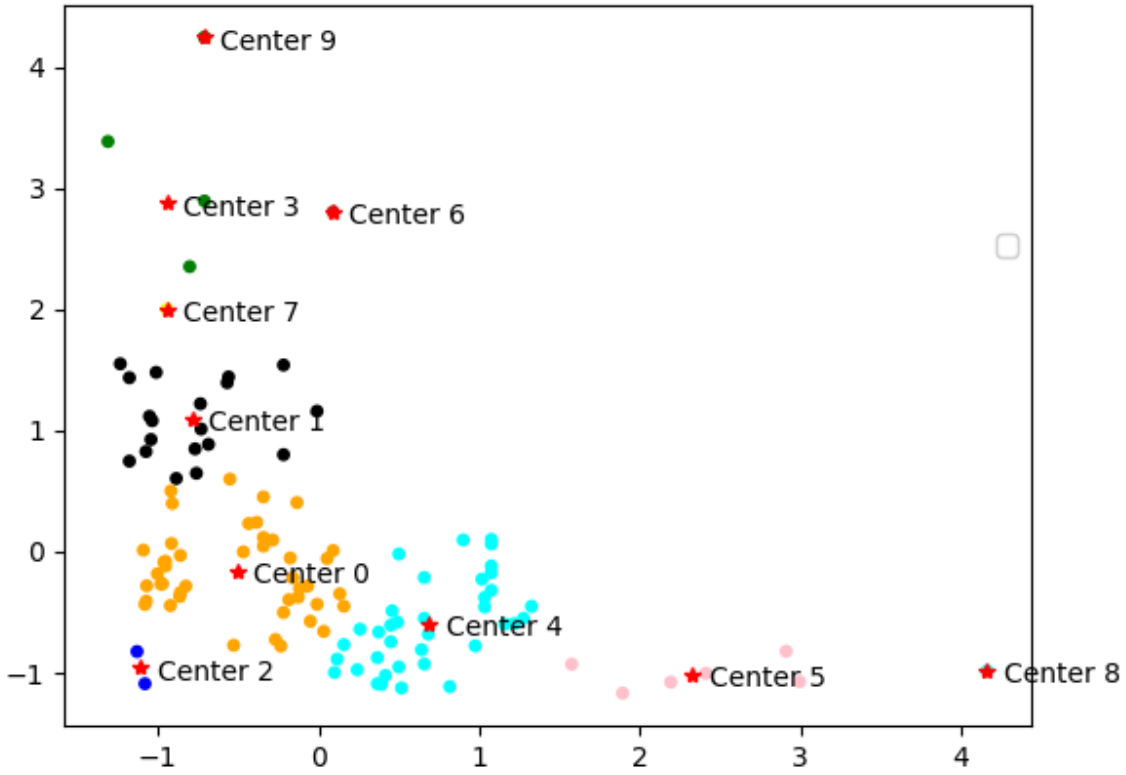

Рисунок 3 (Проверка чувствительности к погрешностям)

Метод к погрешностям не чувствителен.

Пункт 4

Сравним с методами из практической работы №6.

Рассчитаем оценку качества разбиения для всех трех методов кластеризации:

K-means	K-medoids	Forel
21.94	35.31	28.04

Таким образом, FOREL при R=0.7 показал приблизительно такое же качество разбиения, как алгоритмы k-means и k-medoids. Хуже всего из всех трёх методов себя показал метод k-medoids

Выводы.

В процессе выполнения практической работы был реализованы такой алгоритм разбиения на кластеры, как метод поиска сгущений Forel. А также установлена его чувствительность к погрешности и сравнение с таким алгоритмами, как k-means и k-medians, которое привело к выводу, что на данной выборке метод справляется лучше k-medoids, но хуже k-means.

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММЫ

```
import numpy as np
import matplotlib.pyplot as plt
import math
import csv
# import scipy
import random


# from openpyxl.drawing.image import Image


def calc_distance(coordinate_1, coordinate_2):
    return (coordinate_1[0] - coordinate_2[0]) ** 2 + (coordinate_1[1] -
coordinate_2[1]) ** 2


def recalc_medoids(s, clu, medoid):
    m = 10 ** 10
    ans = -1
    for i in range(len(clu)):
        med = medoid
        cluL = clu
        med = s[cluL[i]]
        rang = all_ranges(s, clu, med)
        if rang < m:
            m = rang
            ans = i
    if ans != -1:
        medoid = s[clu[ans]]
    return clu, medoid


def all_ranges(s, clu, medoid):
    ans = 0
    for i in range(len(clu)):
        ans += calc_distance(medoid, s[clu[i]])
    return ans


def cluster_index(D, cluster):
    return [D.index(coordinate) for coordinate in cluster]


def recalc_centroids(cluster):
    centroid = []
    temp_1 = 0
    temp_2 = 0
    len_cluster = len(cluster)
    for i in range(len_cluster):
        temp_1 += cluster[i][0]
        temp_2 += cluster[i][1]
```

```
    centroid.append(temp_1 / len_cluster)
    centroid.append(temp_2 / len_cluster)
    return centroid


def make_plot(centers, sample, clusters):
    mycolors = ['orange', 'black', 'blue', 'green', 'cyan', 'pink', 'brown',
'yellow', 'aqua','darkgreen','darkmagenta',
'darkseagreen','fuchsia','goldenrod','indianred','lightseagreen','palegreen',
'salmon','skyblue','turquoise','coral','darkblue','firebrick','gold','indigo'
,'lime']
    sample = np.array(sample)
    fig, ax = plt.subplots()
    for i, coordinate in enumerate(centers):
        ax.plot(coordinate[0], coordinate[1], 'r*')
        ax.text(coordinate[0] + 0.1, coordinate[1] - 0.1, 'Center ' + str(i))
    labels_ = [0] * len(sample)
    sum = 0
    for i in range(len(clusters)):
        for index in range(len(clusters[i])):
            labels_[clusters[i][index]] = mycolors[i]

    # for cluster in clusters:
    #     for index in cluster:
    #         ax.text(D[index, 0] + 0.01, D[index, 1] + 0.1, index)
    scatter = ax.scatter(sample[:, 0], sample[:, 1], s=15, c=labels_,
cmap='rainbow')
    legend = ax.legend(*scatter.legend_elements(fmt='Cluster {x:.0f}'),
bbox_to_anchor=(1, 0.7))
    ax.add_artist(legend)
    plt.show()


sample = []

with open('Price-Mileage.csv') as csv_file:  # Читаем выборку из файла 0
работы
    spam_reader = csv.reader(csv_file, quotechar='|')
    for row in spam_reader:
        x, y = row[0].split(';')
        if y.isdigit() and x.isdigit():
            sample.append([int(x), int(y)])

n = len(sample)
k = math.ceil(math.sqrt(n / 2))

Xs = 0  # Выброчное среднее по x
Ys = 0  # По y

Dsx = 0  # Дисперсия по x
Dsy = 0  # По s

sy = 0  # Исправленное СКО по y
sx = 0  # По x

for i in range(n):
```

```
    Xs += sample[i][0]
Xs = Xs / n

for i in range(n):
    Ys += sample[i][1]
Ys = Ys / n

for i in range(n):
    Dsx += (sample[i][0] - Xs) ** 2
Dsx = Dsx / n

for i in range(n):
    Dsy += (sample[i][1] - Ys) ** 2
Dsy = Dsy / n

sx = math.sqrt(Dsx) * n / (n - 1)
sy = math.sqrt(Dsy) * n / (n - 1)

for i in range(n):  # Нормализация точек
    sample[i][0] = (sample[i][0] - Xs) / sx
    sample[i][1] = (sample[i][1] - Ys) / sy

X = [sample[i][0] for i in range(n)]
Y = [sample[i][1] for i in range(n)]

plt.scatter(x=X, y=Y, s=15)
ax = plt.gca()
ax.set(title='Выборка')
#  Добавляем подписи к осям:
ax.set_xlabel('Price')
ax.set_ylabel('Mileage')
plt.show()

def ranges_clusters(clus, centrs, samp):
    ans = 0
    for i in range(len(clus)):
        ans += all_ranges(samp, clus[i], centrs[i])
    return ans


def rang_R(point, sam, R):
    clu = []
    for i in sam:
        ze = calc_distance(point, i)
        if ze <= R:
            clu.append(i)
    return clu


def find_thick(R, sam):
    samp = sam  # Список из которого мы будем изымать точки
    cluster = []
    cents = []
    c = -1
    for i in range(len(samp)):
        samp[i].append(i)
```

```
    while len(samp) != 0:
        c += 1
        thr = 1
        init = random.choice(samp)
        while thr != 0:
            clu = []
            cluN = []
            clu = rang_R(init, samp, R)
            for i in clu:
                cluN.append(i[2])
            new_init = recalc_centroids(clu)
            thr = calc_distance(init, new_init)
            init = new_init
        for i in clu:
            samp = list(filter(lambda e: e != i, samp))
        cluster.append(cluN)
        cents.append(init)
    return cluster, cents


def ranges_clusters(clus, centrs, samp):
    ans = 0
    for i in range(len(clus)):
        ans += all_ranges(samp,clus[i],centrs[i])
    return ans

centers = []
Radius = 0.7
clusters, centers = find_thick(Radius, sample)
make_plot(centers, sample, clusters)

range = ranges_clusters(clusters, centers, sample)
print('range = ', range)

b=0
```