**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №5**

**по дисциплине «Машинное обучение»**

Тема: **Кластеризация (k-средних, иерархическая)**

|  |  |  |
| --- | --- | --- |
| Студент гр. 8304 |  | Кирьянов Д. И. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

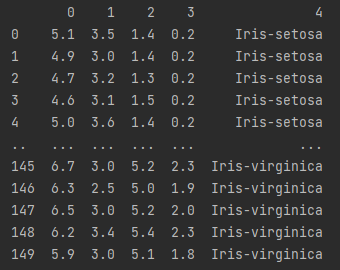
2021

**Цель работы.**

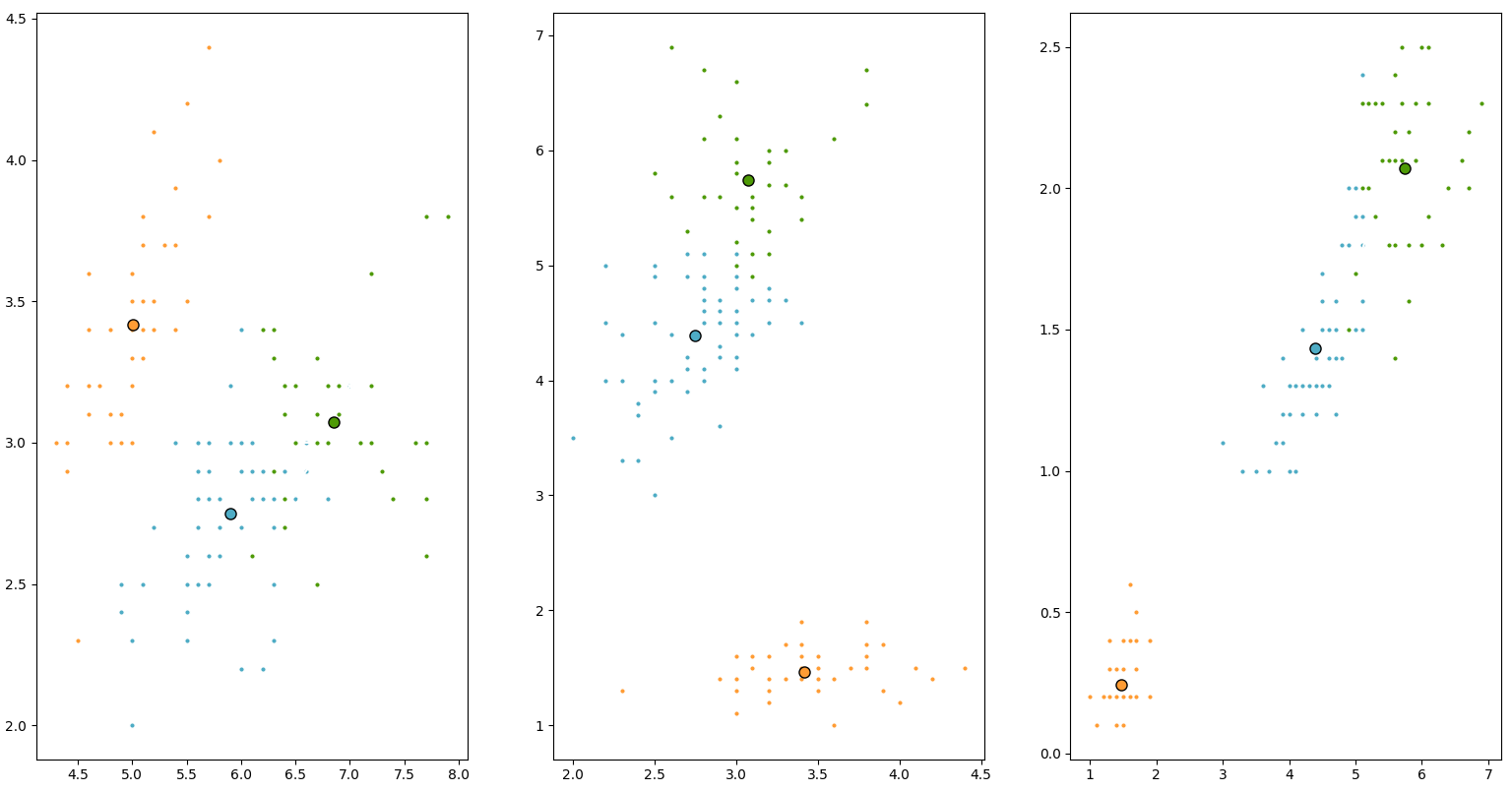
Ознакомиться с методами кластеризации модуля Sklearn

**Ход работы.**

1. **Загрузка данных**
   1. Загружен датасет.
   2. Создан Python скрипт. Данные загружены в датафрейм.

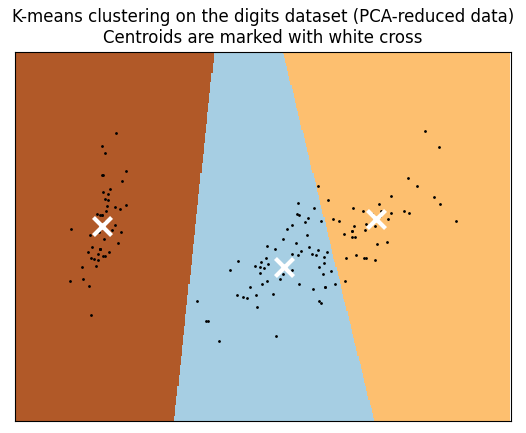


1. **K-means**
   1. Проведена кластеризация методов k-средних

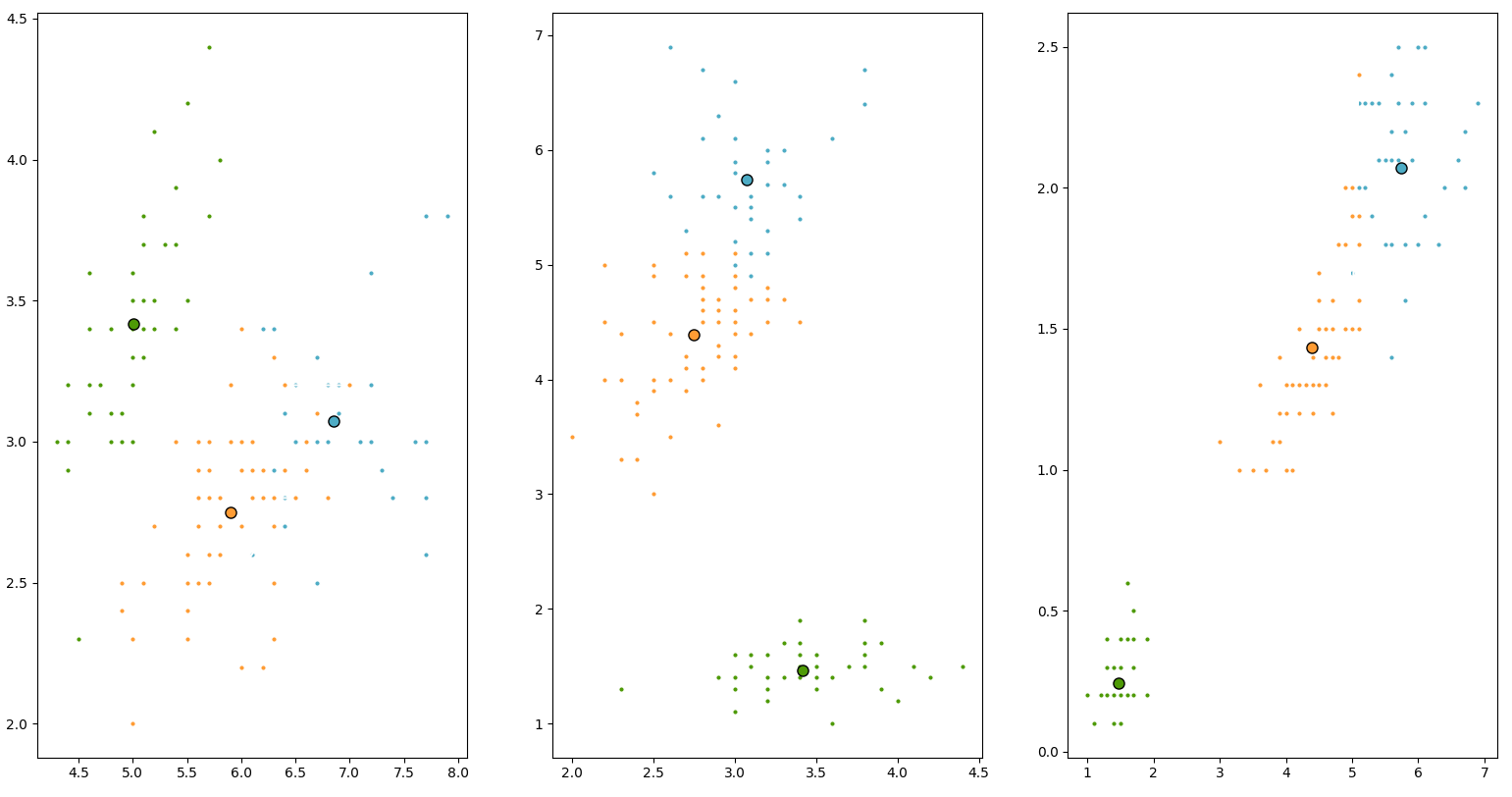


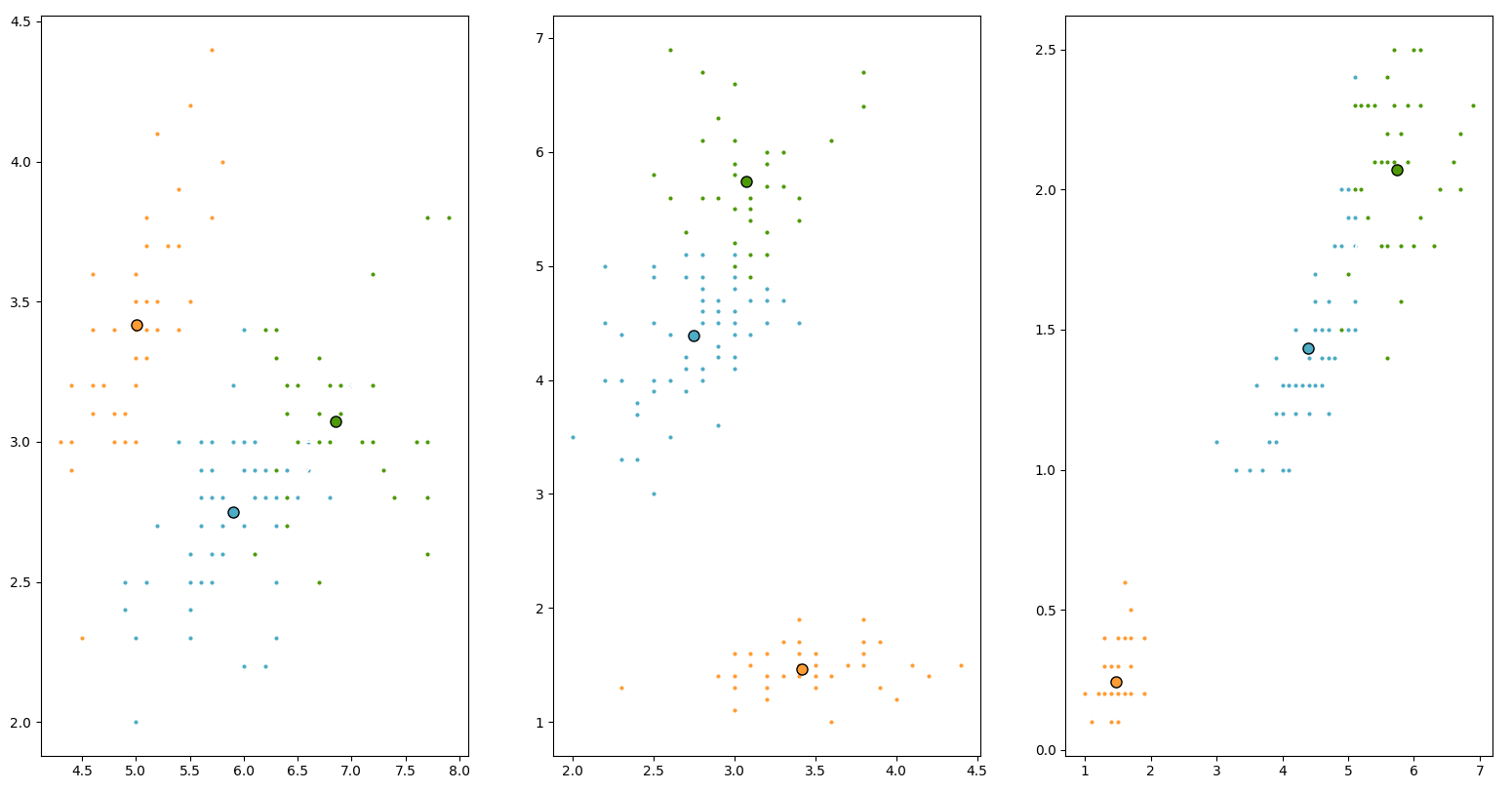
Исходя из рисунка, наилучшее разделение прошло по признакам 3 и 4. Параметр n\_init в данном случае не оказал видимого влияния на результаты.

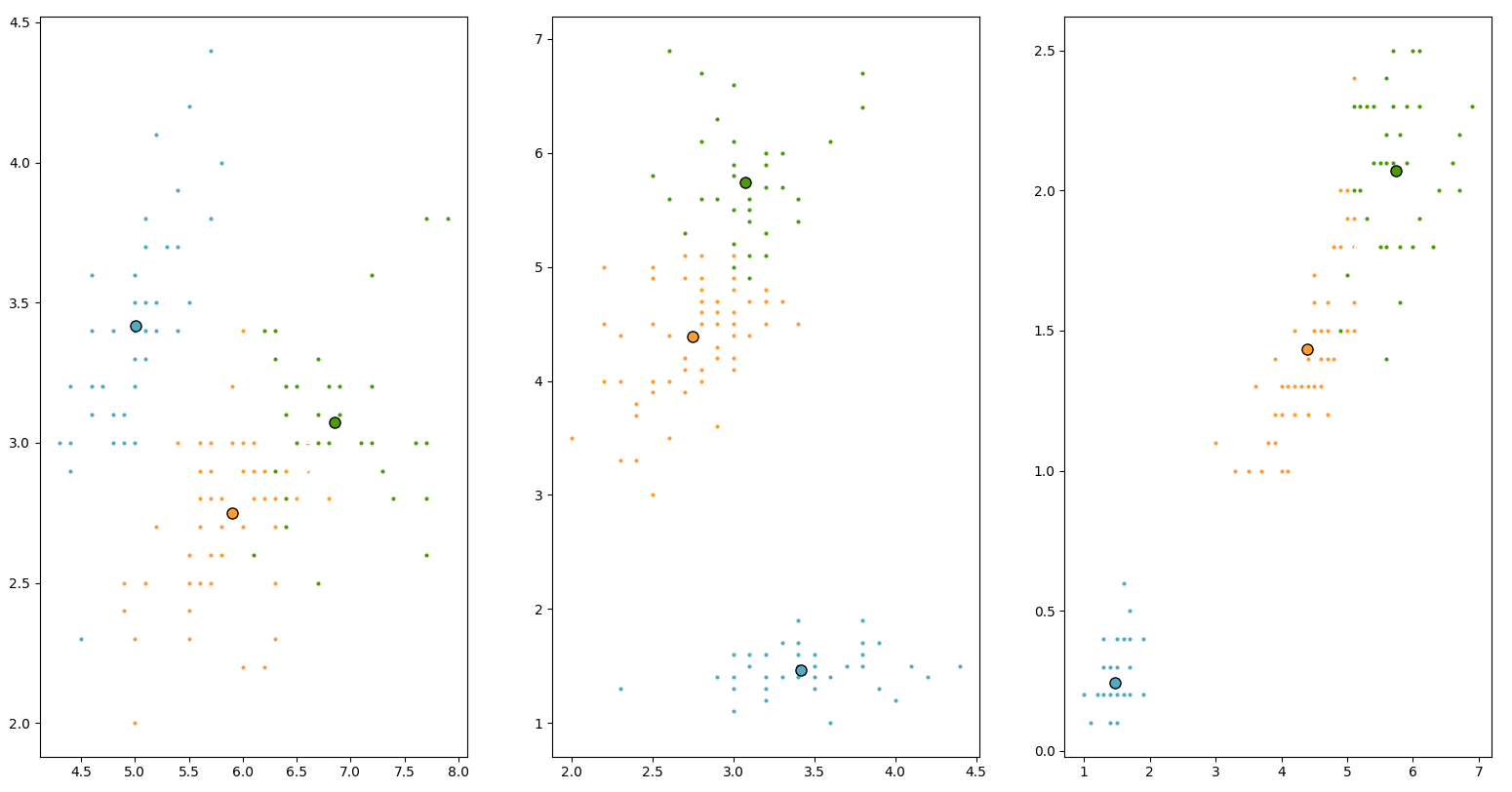
* 1. Размерность данных уменьшена до 2 используя метод главных компонент, нарисована карта для всей области значений, на которой каждый кластер занимает определенную область со своим цветом



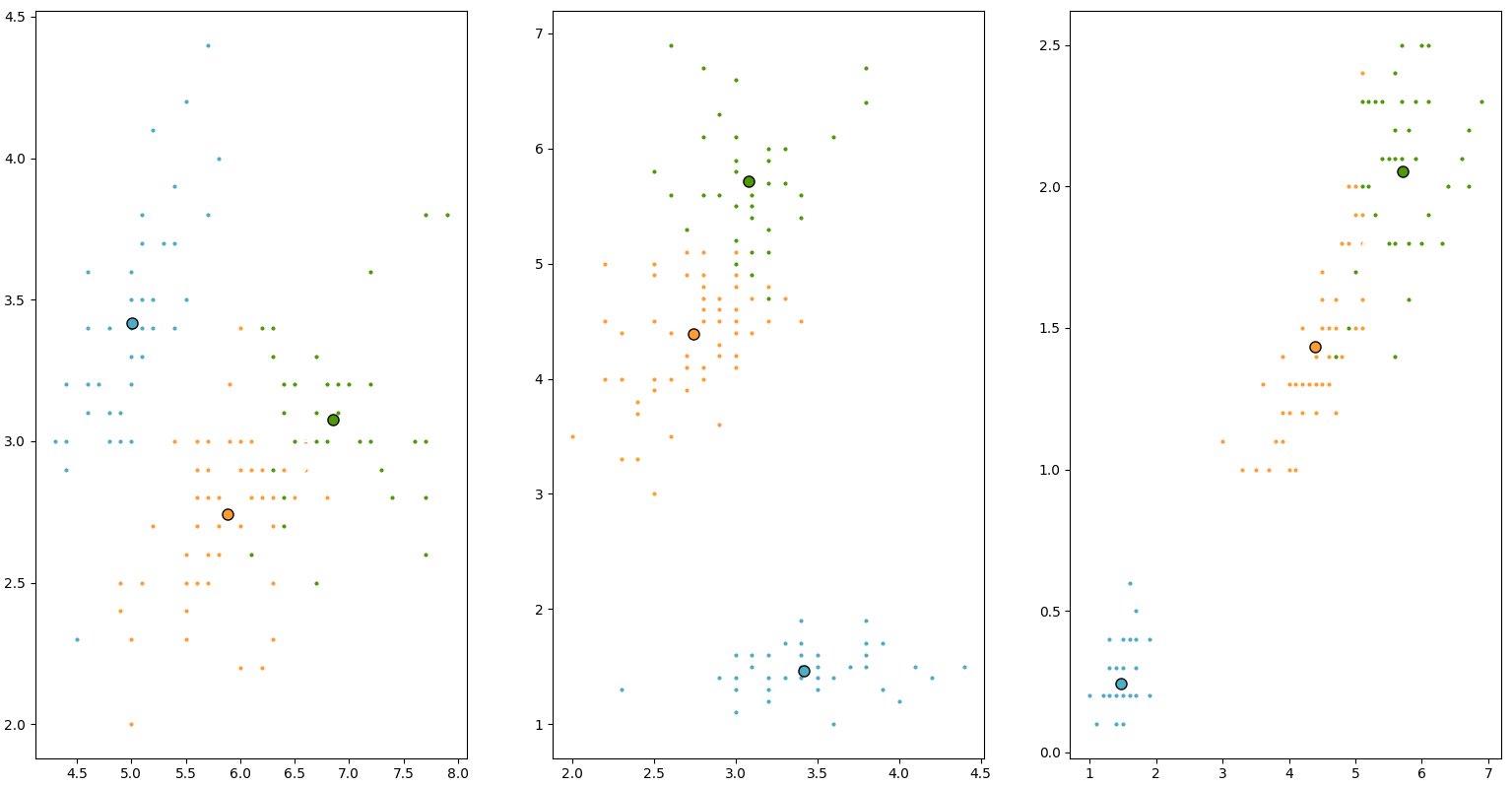
* 1. Исследование работы алгоритма k-средних при различных параметрах init
* init: random, 3 повторения



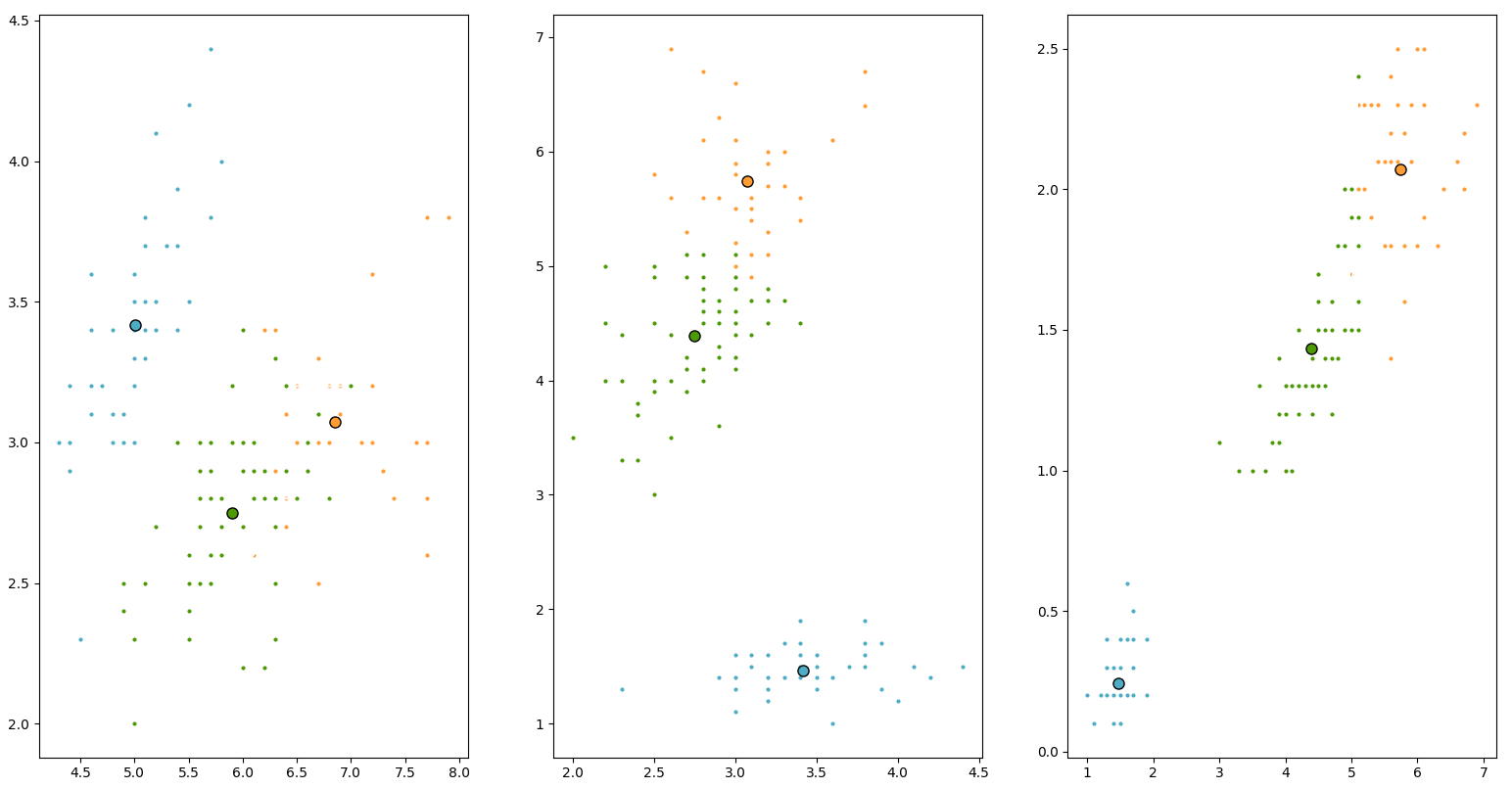




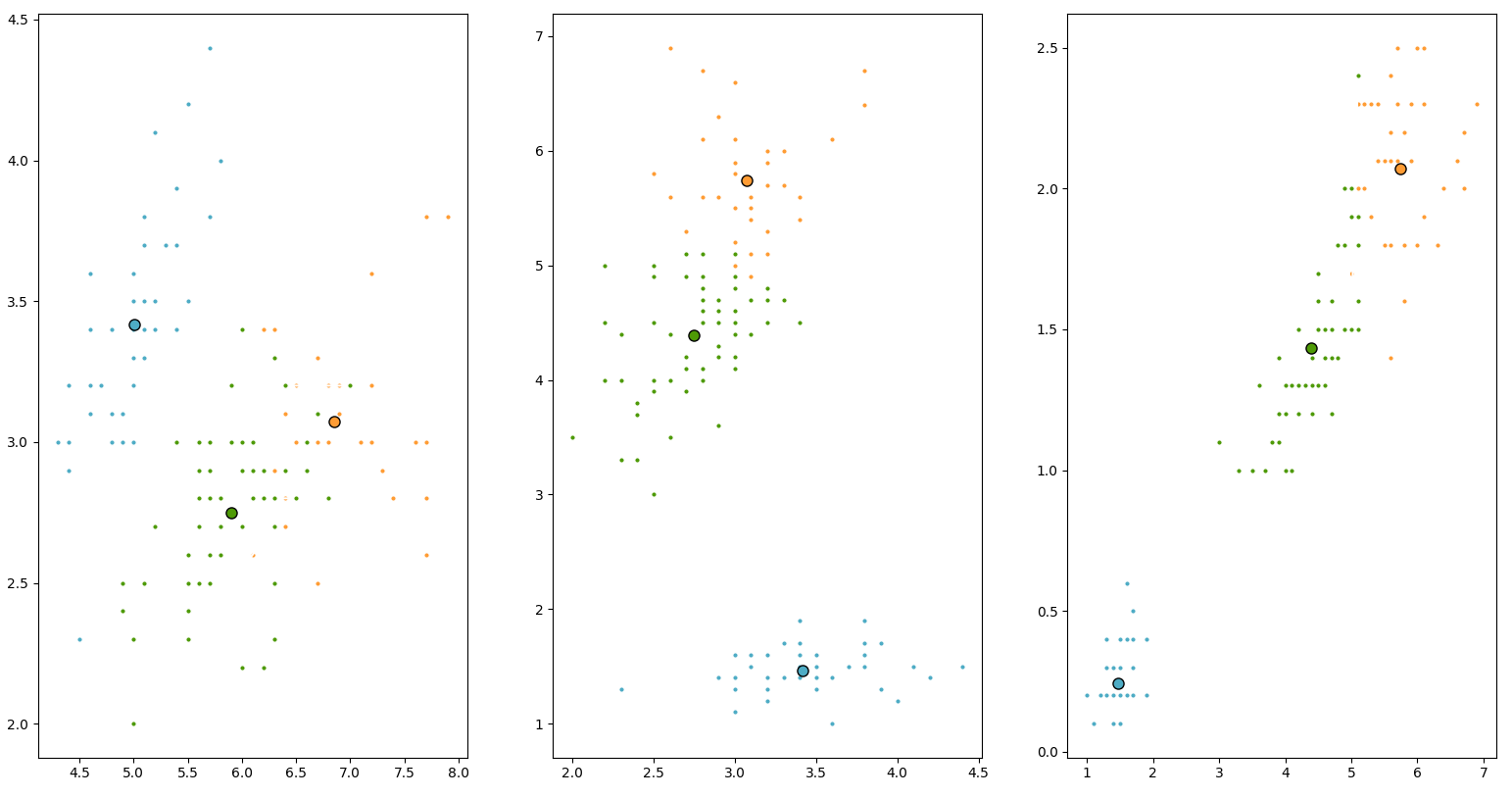
* init=np.array([[5.0, 3.4, 1.5, 0.2],[5.8, 2.2, 4.4, 1.5],[6.8, 3.1, 5.9, 2.2]])



* init=np.array([[1, 1, 1, 1], [50, 50, 50, 50], [100, 100, 100, 100]])



* init=np.array([[1, 1, 1, 1], [1, 1, 1, 1], [1, 1, 1, 1]])

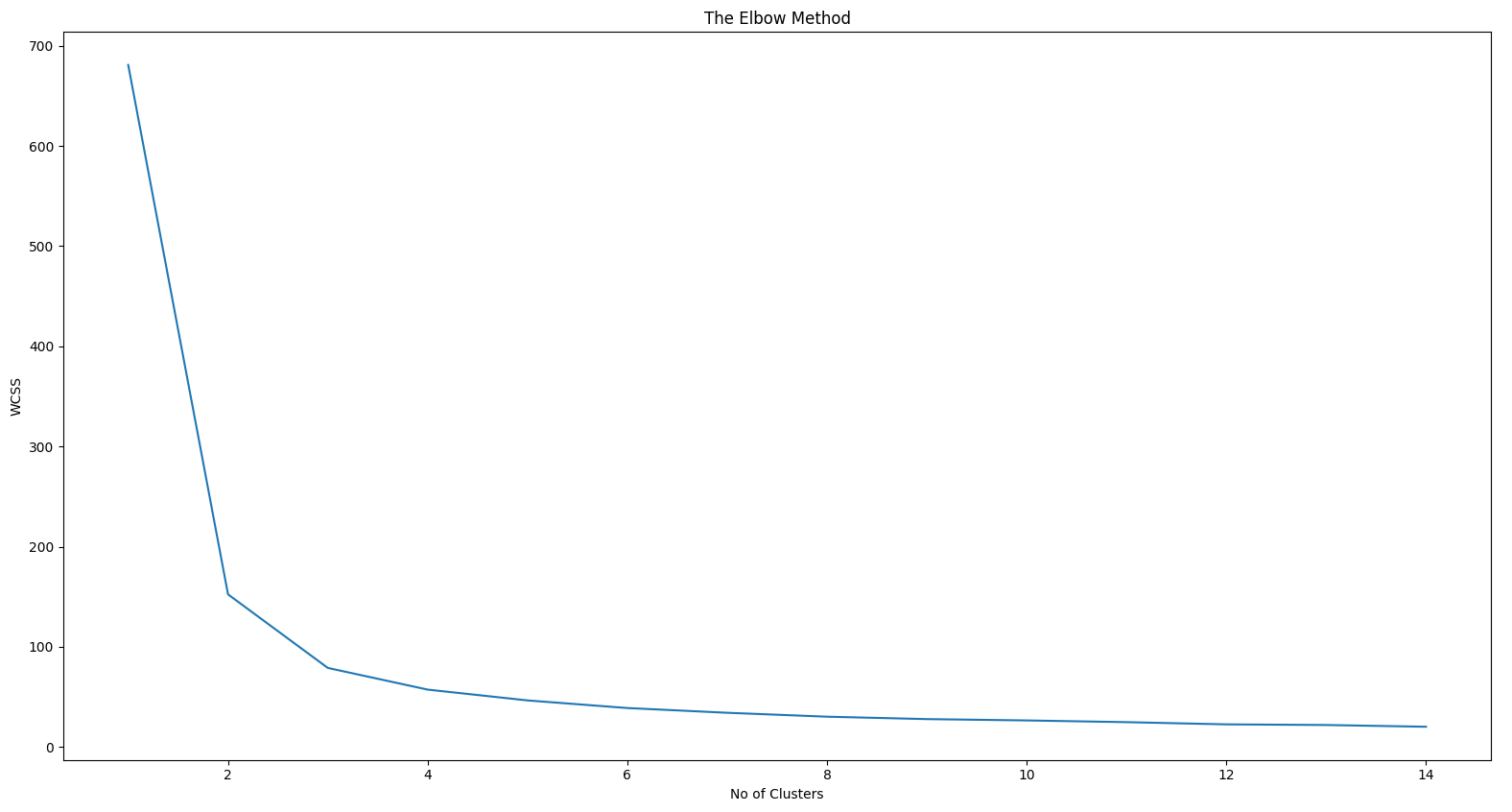


k-means++ выбирает первый центр случайно, последующий выбирается так, что вероятность выбора точки была пропорциональна вычисленному для нее квадрату расстояния.

random выбирает случайные центры.

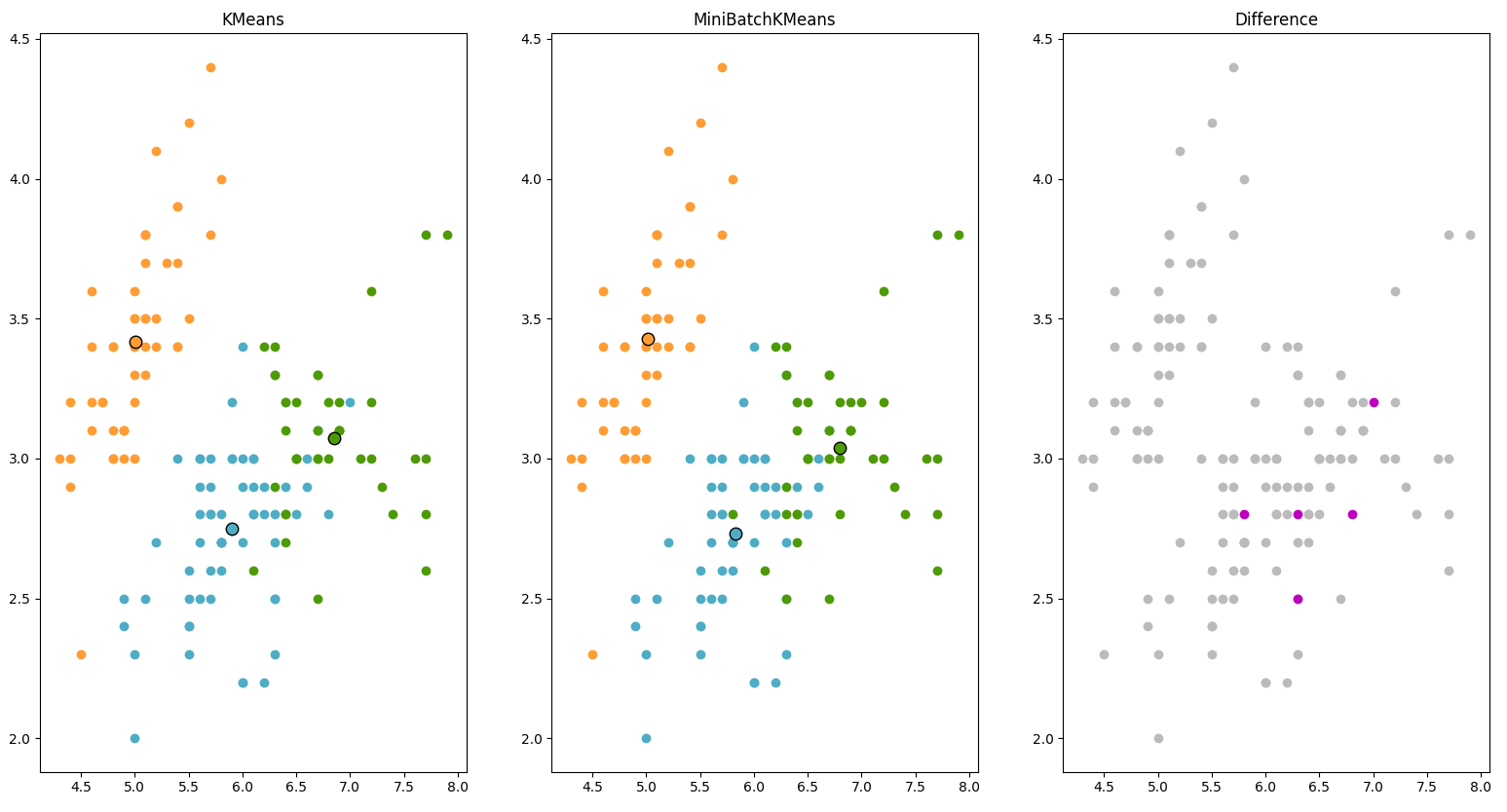
массив точек выбирает заданные центры.

* 1. Определено наилучшее количество кластеров методом локтя.

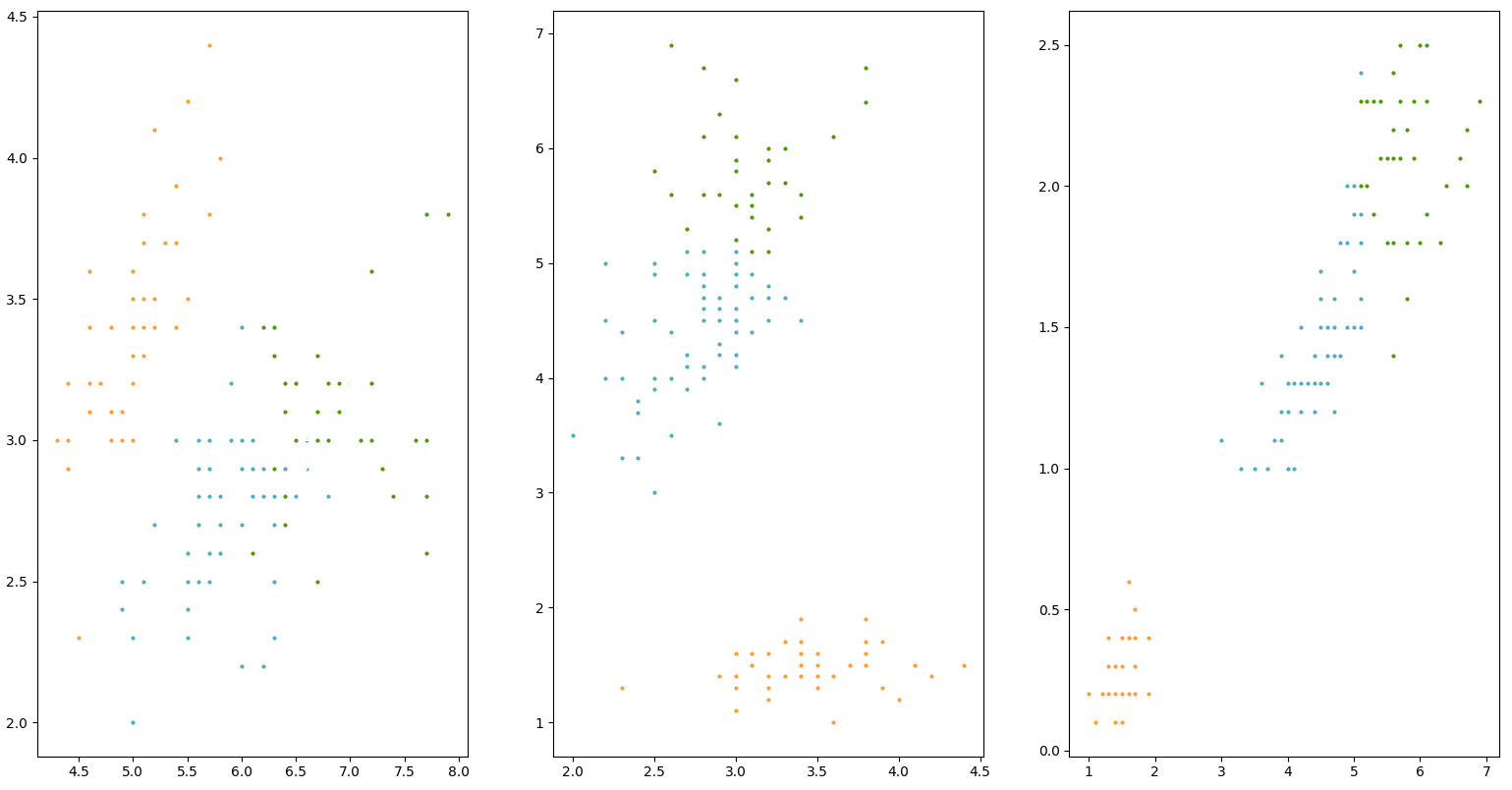


Можно сказать, что 2 кластера являются наилучшим.

* 1. Проведена кластеризация с использованием пакетной кластеризации методом k­-средних. Точки, которые для разных методов попали в разные кластеры отмечены фиолетовым.



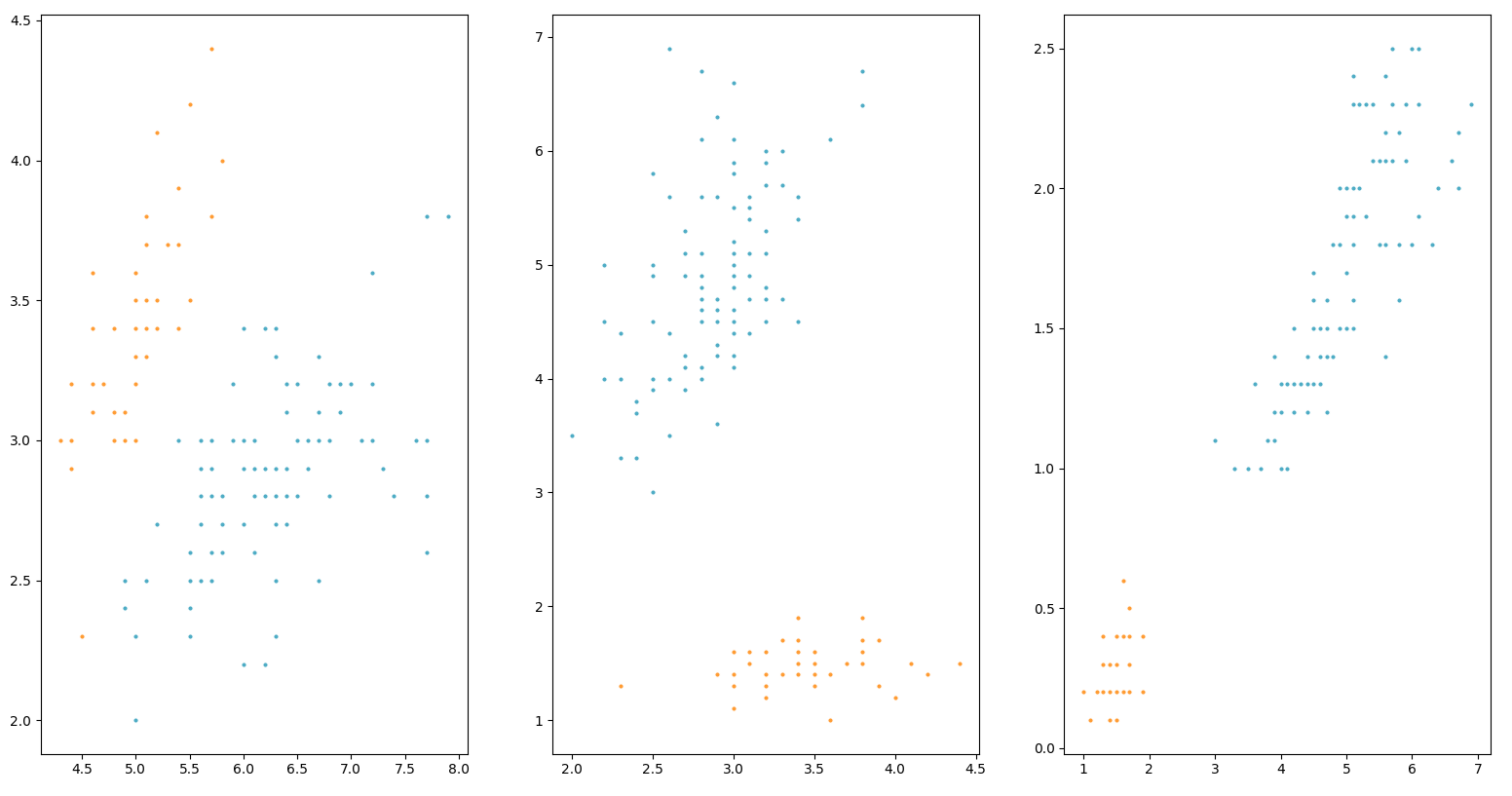
1. **Иерархическая кластеризация**
   1. Проведена иерархическая кластеризация



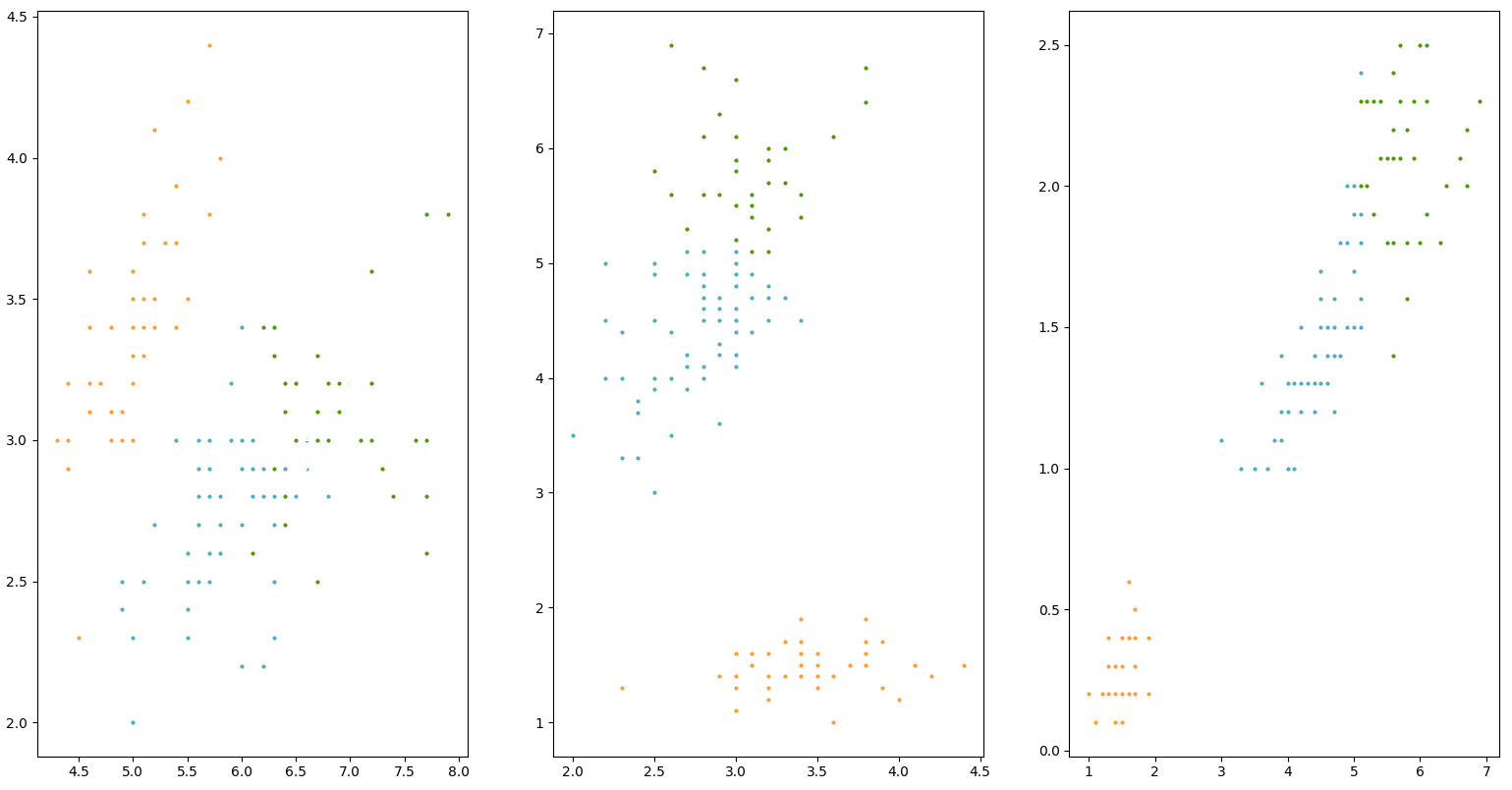
Отличие AgglomerativeClustering от KMeans в алгоритме. Первый начинает с состояния, где все точки принадлежат собственному кластеру, состоящей из одной точки, и объединяет ближайшие кластеры на основе выбранной метрики.

* 1. Проведено исследования для различного размера кластеров

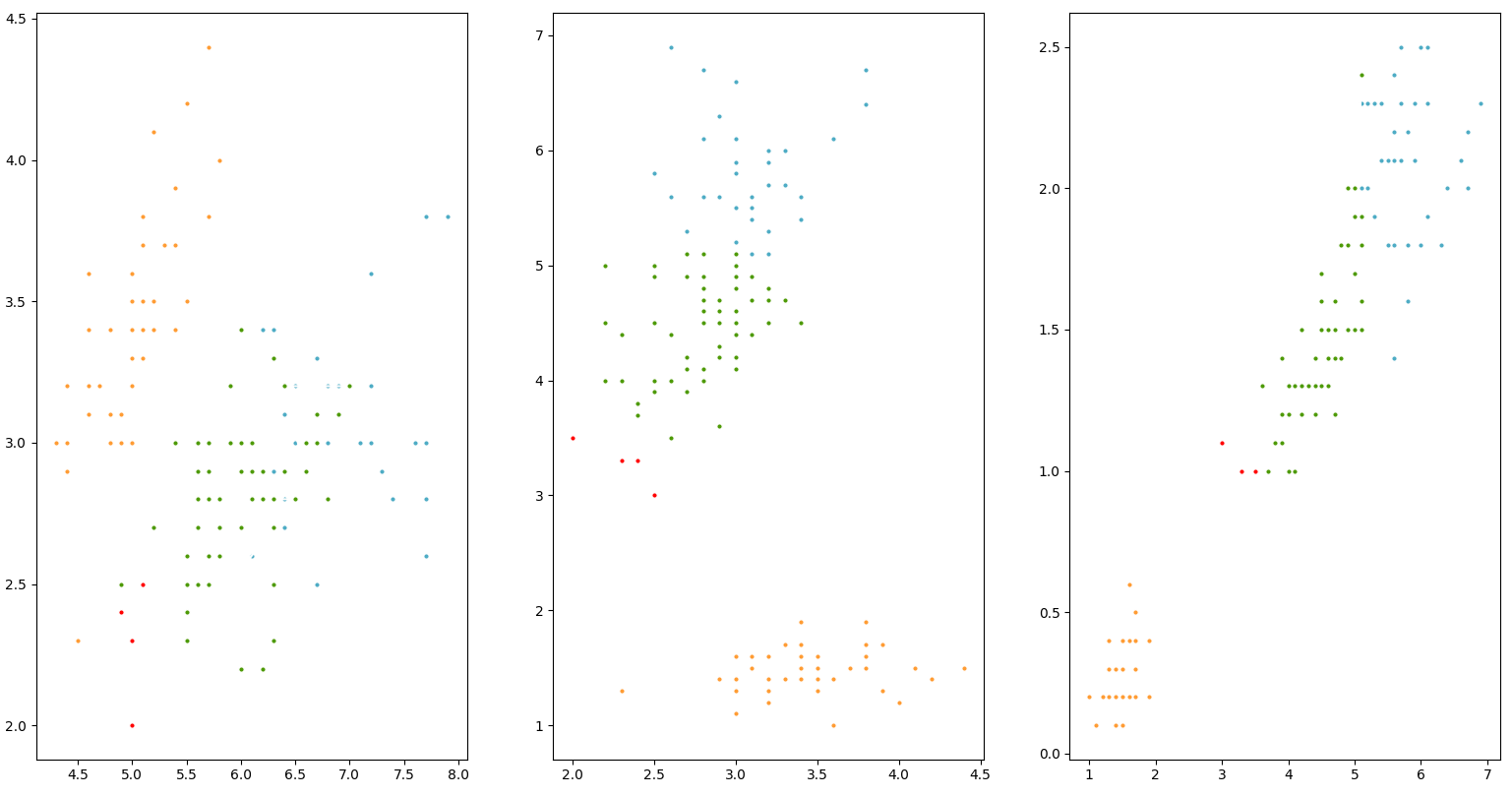
2 кластера:



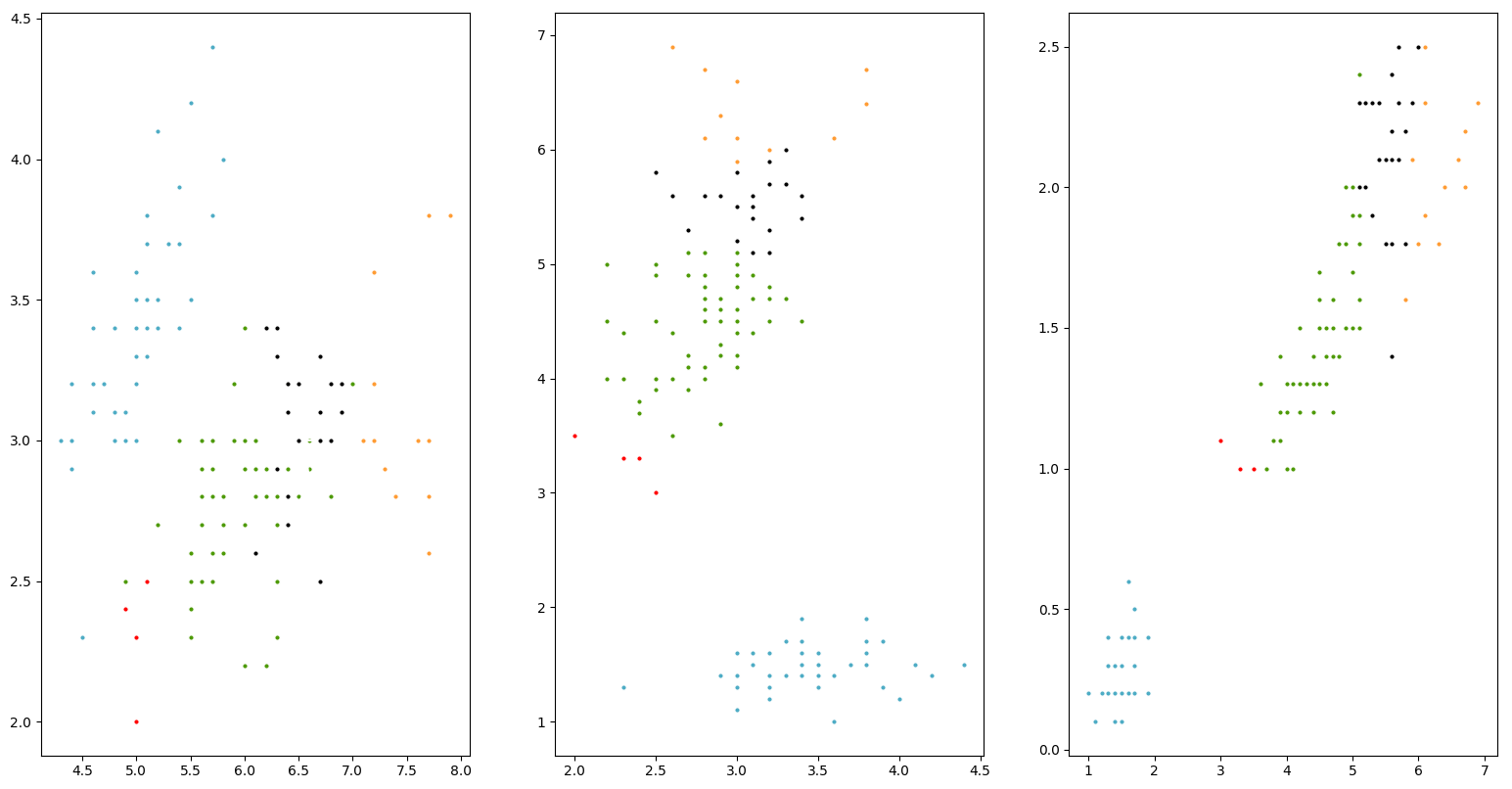
3 кластера:



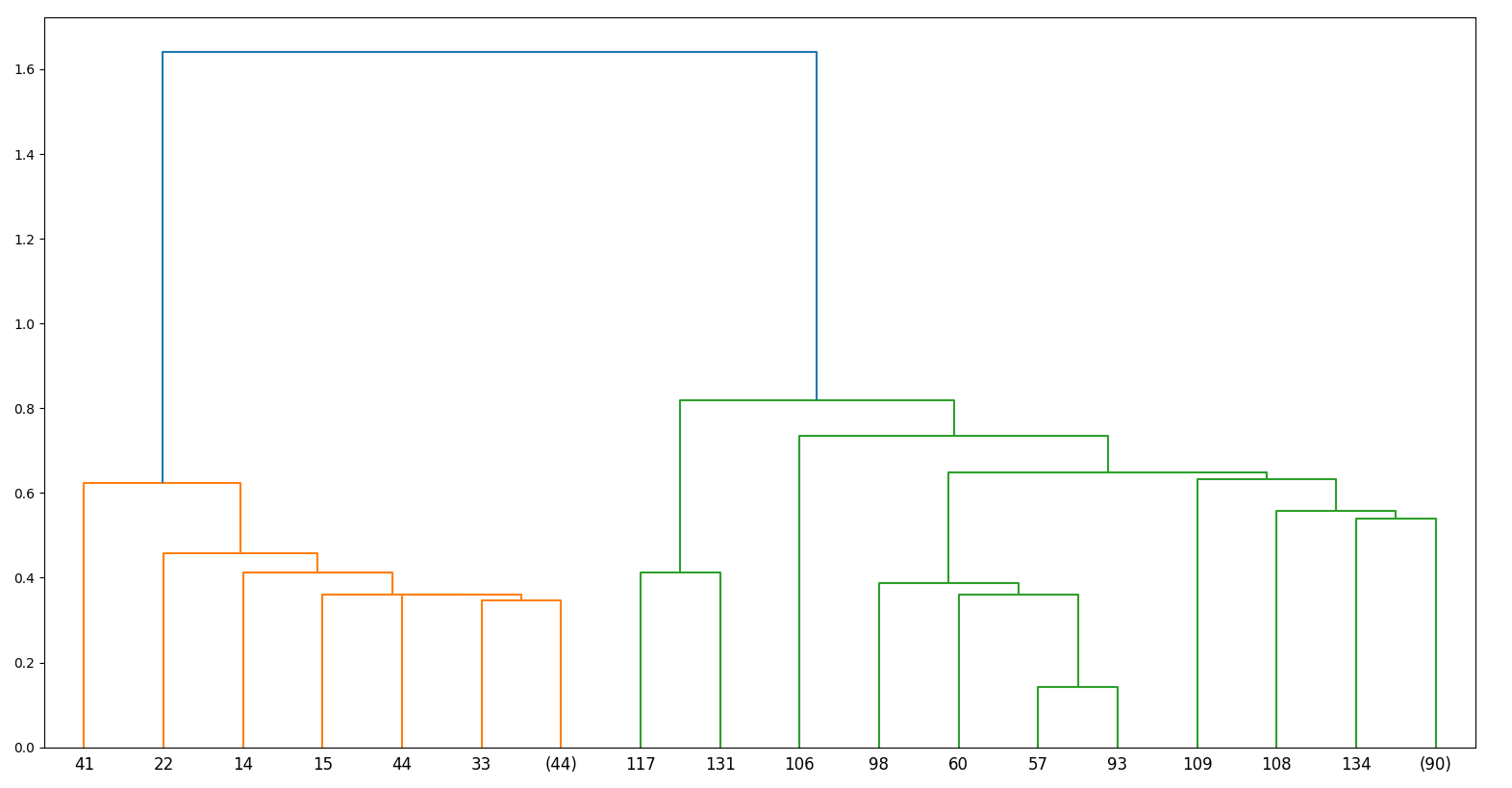
4 кластера:



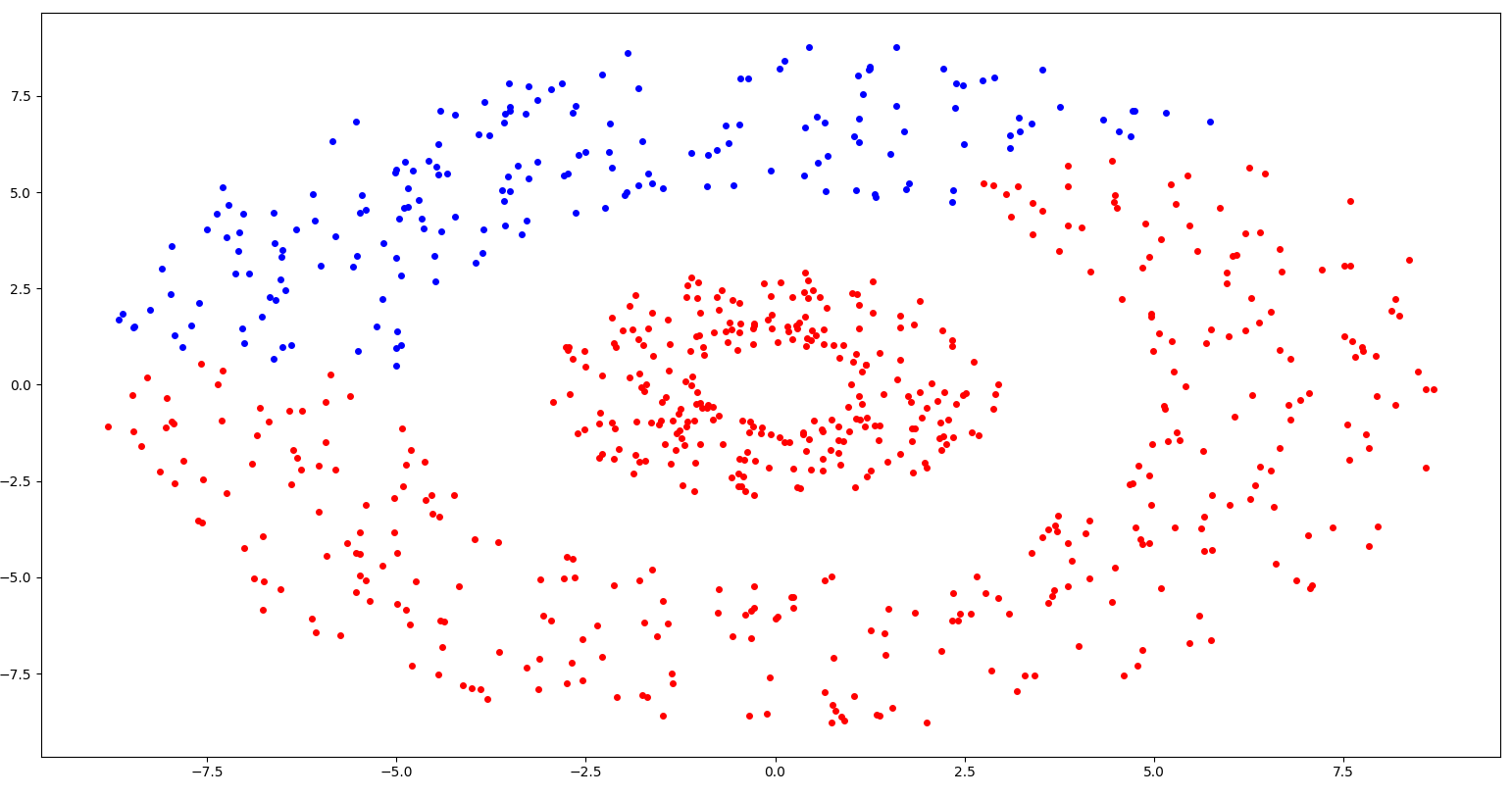
5 кластеров:



* 1. Построена дендрограмма

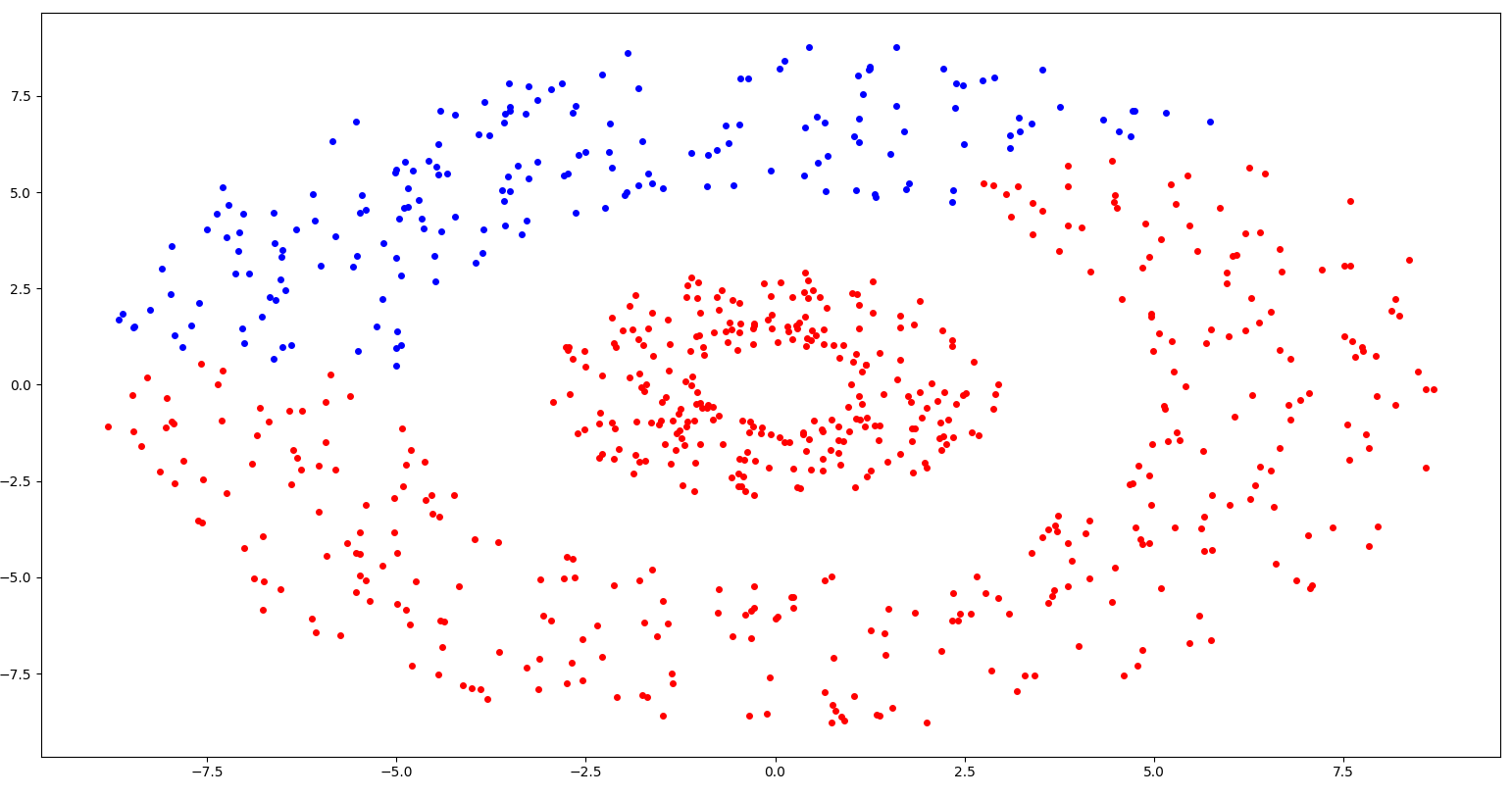


* 1. Сгенерированы случайные данные в виде двух колец. Проведена иерархическая кластеризация.

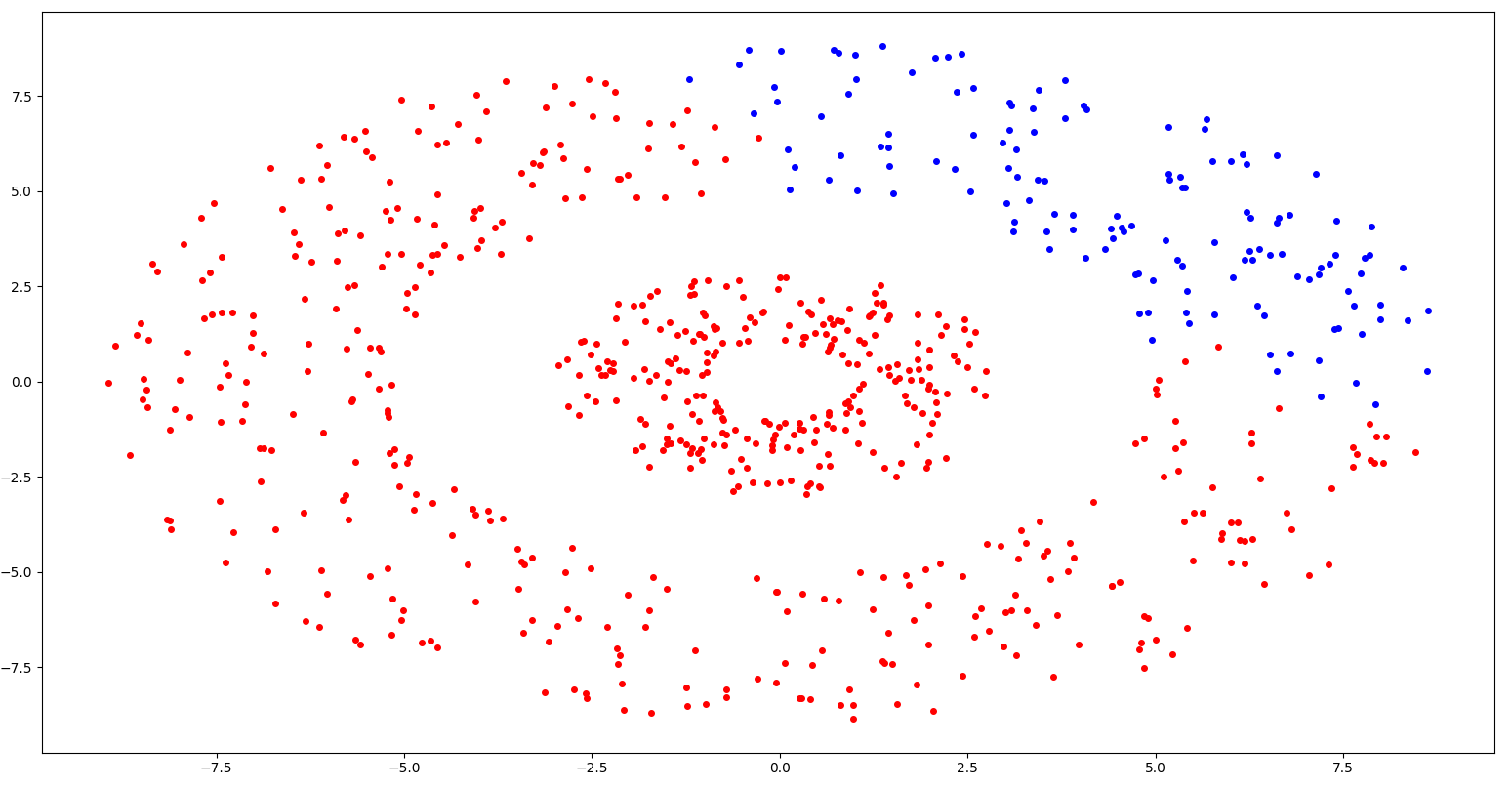


* 1. Проведены исследования для различных параметров linkage

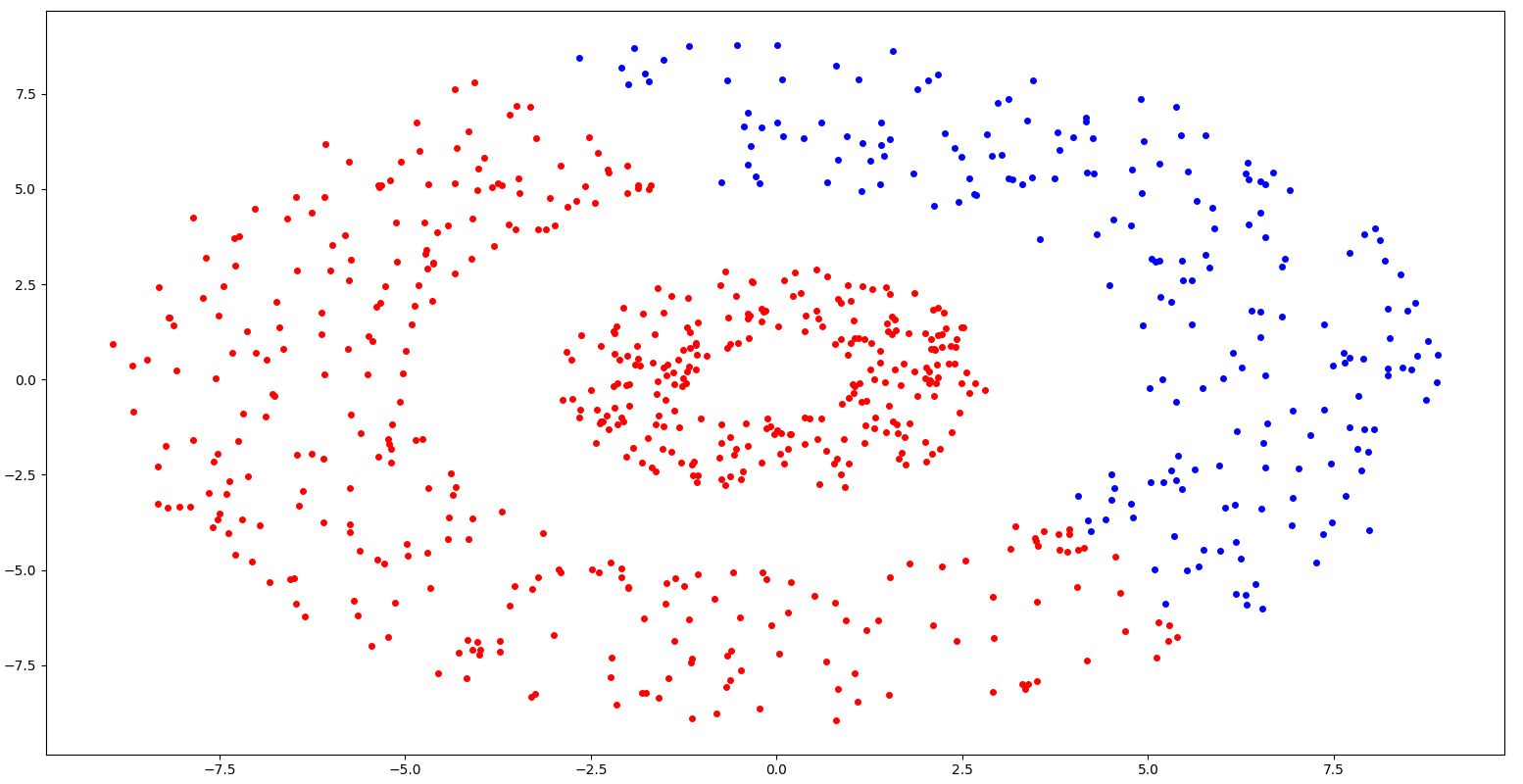
ward:



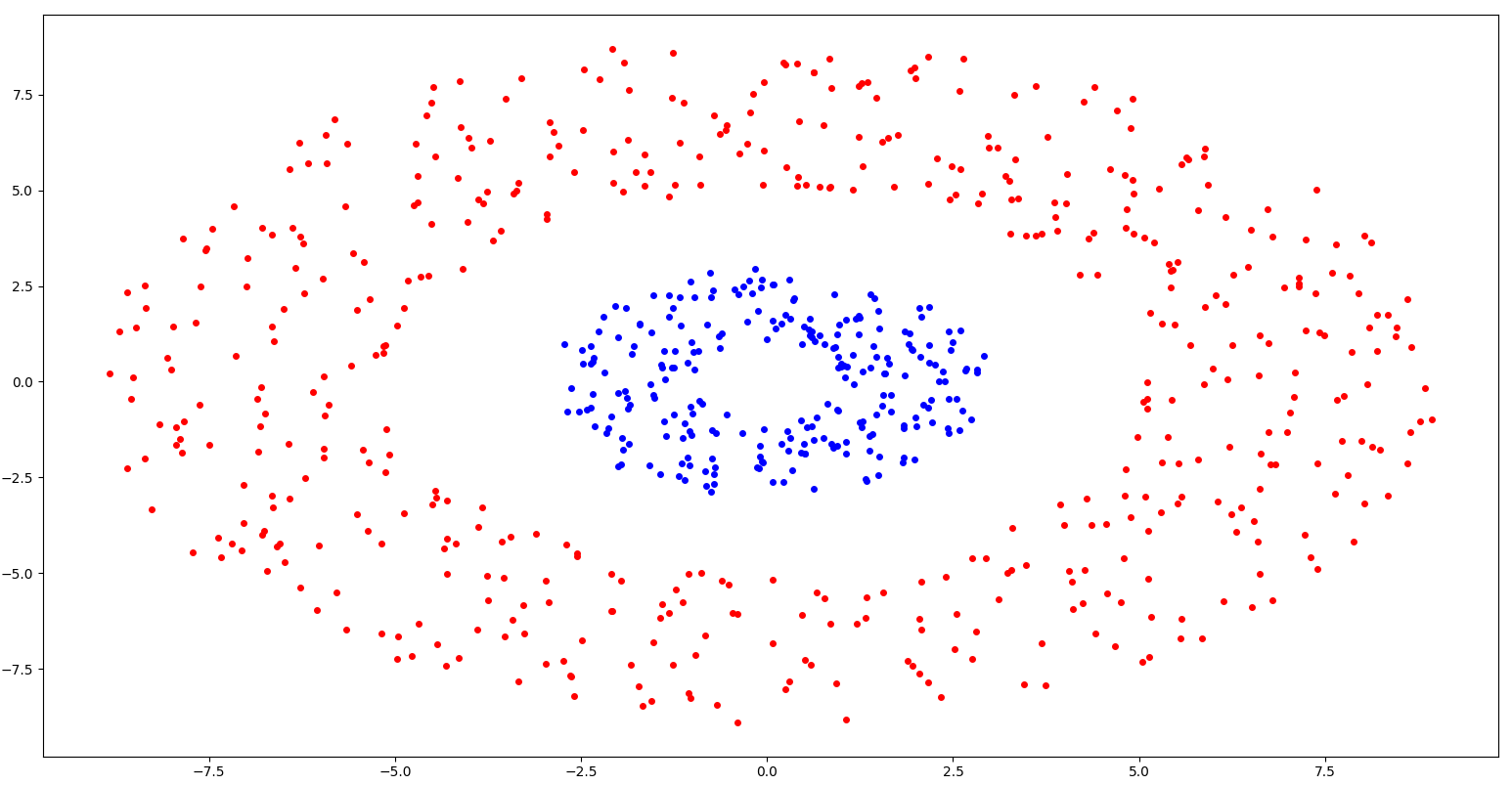
average:



complete:



single:



ward – минимизация суммы квадратов разностей

complete – минимизация максимального расстояния

average – минимизация среднего расстояния

single – минимизация расстояния

Лучший результат был достигнут при single, т.к. произошло разделение колец.

**Вывод**

Ознакомились с методами кластеризации модуля Sklearn.

**ПРИЛОЖЕНИЕ А**

**Исходный код программы**

import pandas as pd  
import numpy as np  
from sklearn.cluster import KMeans, MiniBatchKMeans, AgglomerativeClustering  
from sklearn.metrics.pairwise import pairwise\_distances\_argmin  
import matplotlib.pyplot as plt  
from sklearn.decomposition import PCA  
from scipy.cluster.hierarchy import dendrogram, linkage  
import random  
import math  
  
  
data = pd.read\_csv('iris.data', header=None)  
print(data)  
no\_labeled\_data = data.iloc[:,:4].to\_numpy()  
k\_means = KMeans(init='k-means++', n\_clusters=3, n\_init=15)  
k\_means.fit(no\_labeled\_data)  
k\_means\_cluster\_centers = k\_means.cluster\_centers\_  
k\_means\_labels = pairwise\_distances\_argmin(no\_labeled\_data, k\_means\_cluster\_centers)  
f, ax = plt.subplots(1, 3)  
colors = ['#4EACC5', '#FF9C34', '#4E9A06']  
print(ax)  
for i in range(3):  
 my\_members = k\_means\_labels == i  
 cluster\_center = k\_means\_cluster\_centers[i]  
 for j in range(3):  
 ax[j].plot(no\_labeled\_data[my\_members, j], no\_labeled\_data[my\_members, j + 1], 'w',  
 markerfacecolor=colors[i], marker='o', markersize=4)  
 ax[j].plot(cluster\_center[j], cluster\_center[j + 1], 'o', markerfacecolor=colors[i],  
 markeredgecolor='k', markersize=8)  
  
plt.show()  
  
pca = PCA(n\_components=2)  
reduced\_data = pca.fit\_transform(no\_labeled\_data)  
kmeans = KMeans(init='k-means++', n\_clusters=3, n\_init=15)  
kmeans.fit(reduced\_data)  
h = 0.02  
x\_min, x\_max = reduced\_data[:, 0].min() - 1, reduced\_data[:, 0].max() + 1  
y\_min, y\_max = reduced\_data[:, 1].min() - 1, reduced\_data[:, 1].max() + 1  
xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h), np.arange(y\_min, y\_max, h))  
Z = kmeans.predict(np.c\_[xx.ravel(), yy.ravel()])  
Z = Z.reshape(xx.shape)  
plt.figure(1)  
plt.clf()  
plt.imshow(Z,interpolation="nearest",extent=(xx.min(), xx.max(), yy.min(), yy.max()),  
 cmap=plt.cm.Paired,aspect="auto",origin="lower")  
plt.plot(reduced\_data[:, 0], reduced\_data[:, 1], "k.", markersize=2)  
centroids = kmeans.cluster\_centers\_  
plt.scatter(centroids[:, 0],centroids[:, 1],marker="x",s=169,  
 linewidths=3,color="w",zorder=10)  
plt.title("K-means clustering on the digits dataset (PCA-reduced data)\n"  
 "Centroids are marked with white cross")  
plt.xlim(x\_min, x\_max)  
plt.ylim(y\_min, y\_max)  
plt.xticks(())  
plt.yticks(())  
plt.show()  
k\_means = KMeans(init='random', n\_clusters=3, n\_init=15)  
k\_means.fit(no\_labeled\_data)  
k\_means\_cluster\_centers = k\_means.cluster\_centers\_  
k\_means\_labels = pairwise\_distances\_argmin(no\_labeled\_data, k\_means\_cluster\_centers)  
f, ax = plt.subplots(1, 3)  
colors = ['#4EACC5', '#FF9C34', '#4E9A06']  
print(ax)  
for i in range(3):  
 my\_members = k\_means\_labels == i  
 cluster\_center = k\_means\_cluster\_centers[i]  
 for j in range(3):  
 ax[j].plot(no\_labeled\_data[my\_members, j], no\_labeled\_data[my\_members, j + 1], 'w',  
 markerfacecolor=colors[i], marker='o', markersize=4)  
 ax[j].plot(cluster\_center[j], cluster\_center[j + 1], 'o', markerfacecolor=colors[i],  
 markeredgecolor='k', markersize=8)  
  
plt.show()  
k\_means = KMeans(init=np.array([[1, 1, 1, 1], [1, 1, 1, 1], [1, 1, 1, 1]]), n\_clusters=3, n\_init=1)  
k\_means.fit(no\_labeled\_data)  
k\_means\_cluster\_centers = k\_means.cluster\_centers\_  
k\_means\_labels = pairwise\_distances\_argmin(no\_labeled\_data, k\_means\_cluster\_centers)  
f, ax = plt.subplots(1, 3)  
colors = ['#4EACC5', '#FF9C34', '#4E9A06']  
print(ax)  
for i in range(3):  
 my\_members = k\_means\_labels == i  
 cluster\_center = k\_means\_cluster\_centers[i]  
 for j in range(3):  
 ax[j].plot(no\_labeled\_data[my\_members, j], no\_labeled\_data[my\_members, j + 1], 'w',  
 markerfacecolor=colors[i], marker='o', markersize=4)  
 ax[j].plot(cluster\_center[j], cluster\_center[j + 1], 'o', markerfacecolor=colors[i],  
 markeredgecolor='k', markersize=8)  
  
plt.show()  
wcss = []  
for i in range(1, 15):  
 kmean = KMeans(n\_clusters=i, init="k-means++")  
 kmean.fit\_predict(no\_labeled\_data)  
 wcss.append(kmean.inertia\_)  
plt.plot(range(1, 15), wcss)  
plt.title('The Elbow Method')  
plt.xlabel("No of Clusters")  
plt.ylabel("WCSS")  
plt.show()  
data = no\_labeled\_data  
n\_clusters = 3  
np.random.seed(1)  
kmeans = KMeans(init='k-means++', n\_clusters=3, n\_init=15)  
kmeans.fit(data)  
mbk = MiniBatchKMeans(init='k-means++', n\_clusters=3, batch\_size=100, n\_init=15)  
mbk.fit(data)  
fig = plt.figure(figsize=(25, 8))  
k\_means\_cluster\_centers = kmeans.cluster\_centers\_  
order = pairwise\_distances\_argmin(kmeans.cluster\_centers\_, mbk.cluster\_centers\_)  
mbk\_means\_cluster\_centers = mbk.cluster\_centers\_[order]  
k\_means\_labels = pairwise\_distances\_argmin(data, k\_means\_cluster\_centers)  
mbk\_means\_labels = pairwise\_distances\_argmin(data, mbk\_means\_cluster\_centers)  
print(mbk\_means\_labels)  
def ax\_fill(ax, data, labels, centers, title, plot\_centers=True, n\_clusters=3):  
 for k, col in zip(range(n\_clusters), colors):  
 my\_members = labels == k  
 ax.plot(data[my\_members, 0], data[my\_members, 1], 'o',  
 markerfacecolor=col, markersize=6, markeredgecolor=col)  
 if plot\_centers:  
 cluster\_center = centers[k]  
 ax.plot(cluster\_center[0], cluster\_center[1], 'o', markerfacecolor=col,  
 markeredgecolor='k', markersize=9)  
 ax.set\_title(title)  
  
ax = fig.add\_subplot(1, 3, 1)  
ax\_fill(ax,data, k\_means\_labels, k\_means\_cluster\_centers, 'KMeans')  
ax = fig.add\_subplot(1, 3, 2)  
ax\_fill(ax, data, mbk\_means\_labels, mbk\_means\_cluster\_centers, 'MiniBatchKMeans')  
  
different = (mbk\_means\_labels == 4)  
ax = fig.add\_subplot(1, 3, 3)  
  
for k in range(n\_clusters):  
 different += ((k\_means\_labels == k) != (mbk\_means\_labels == k))  
  
identic = np.logical\_not(different)  
ax.plot(data[identic, 0], data[identic, 1], 'o',  
 markerfacecolor='#bbbbbb', markersize=6, markeredgecolor='#bbbbbb')  
ax.plot(data[different, 0], data[different, 1], 'o',  
 markerfacecolor='m', markersize=6, markeredgecolor='m')  
ax.set\_title('Difference')  
plt.show()  
  
hier = AgglomerativeClustering(n\_clusters=5, linkage='average')  
hier = hier.fit(no\_labeled\_data)  
hier\_labels = hier.labels\_  
f, ax = plt.subplots(1, 3)  
colors = ['#4EACC5', '#FF9C34', '#4E9A06', '#FF0000', '#000000']  
for i in range(5):  
 my\_members = hier\_labels == i  
 for j in range(3):  
 ax[j].plot(no\_labeled\_data[my\_members, j], no\_labeled\_data[my\_members, j+1], 'w',  
 markerfacecolor=colors[i], marker='o', markersize=4)  
plt.show()  
  
linked = linkage(no\_labeled\_data)  
dendrogram(linked, truncate\_mode='level', p=6)  
plt.show()  
  
data1 = np.zeros([250,2])  
for i in range(250):  
 r = random.uniform(1, 3)  
 a = random.uniform(0, 2 \* math.pi)  
 data1[i,0] = r \* math.sin(a)  
 data1[i,1] = r \* math.cos(a)  
data2 = np.zeros([500,2])  
for i in range(500):  
 r = random.uniform(5, 9)  
 a = random.uniform(0, 2 \* math.pi)  
 data2[i,0] = r \* math.sin(a)  
 data2[i,1] = r \* math.cos(a)  
data = np.vstack((data1, data2))  
hier = AgglomerativeClustering(n\_clusters=2, linkage='single')  
hier = hier.fit(data)  
hier\_labels = hier.labels\_  
  
my\_members = hier\_labels == 0  
plt.plot(data[my\_members, 0], data[my\_members, 1], 'w', marker='o',markersize=4, color='red', linestyle='None')  
my\_members = hier\_labels == 1  
plt.plot(data[my\_members, 0], data[my\_members, 1], 'w', marker='o', markersize=4, color='blue', linestyle='None')  
plt.show()