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

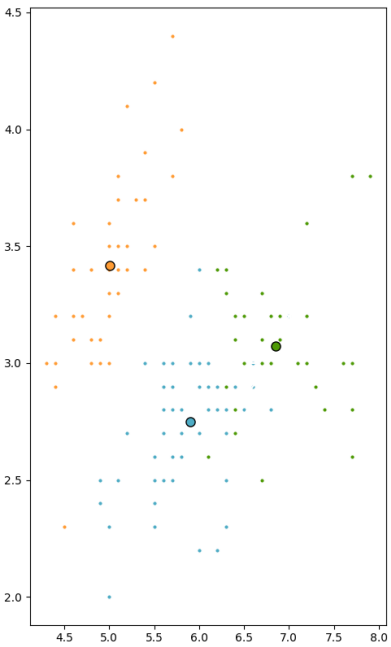
1.1. Загружен датасет.

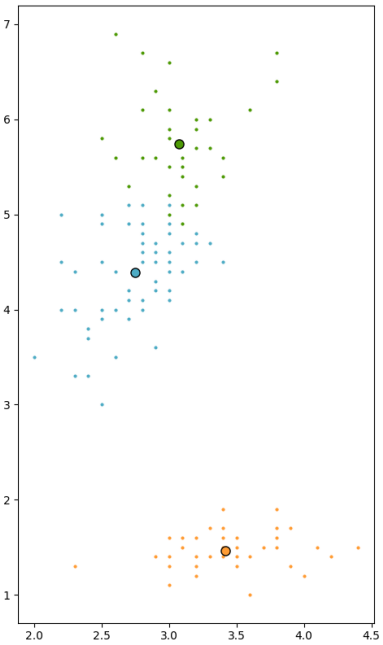
1.2. Создан Python скрипт. Данные загружены в датафрейм.

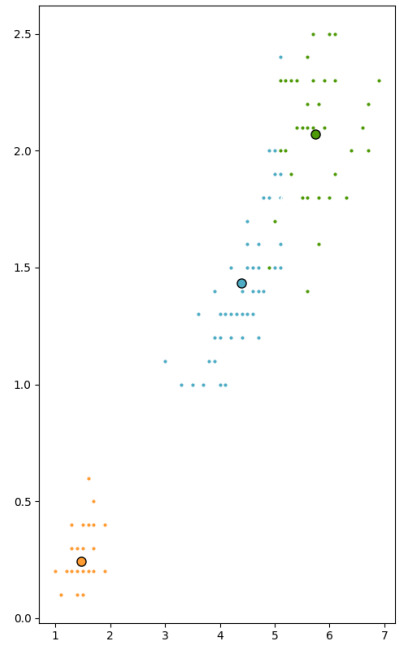
	0	1	2	3	4
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
..	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

2. K-means

2.1. Проведена кластеризация методов k-средних

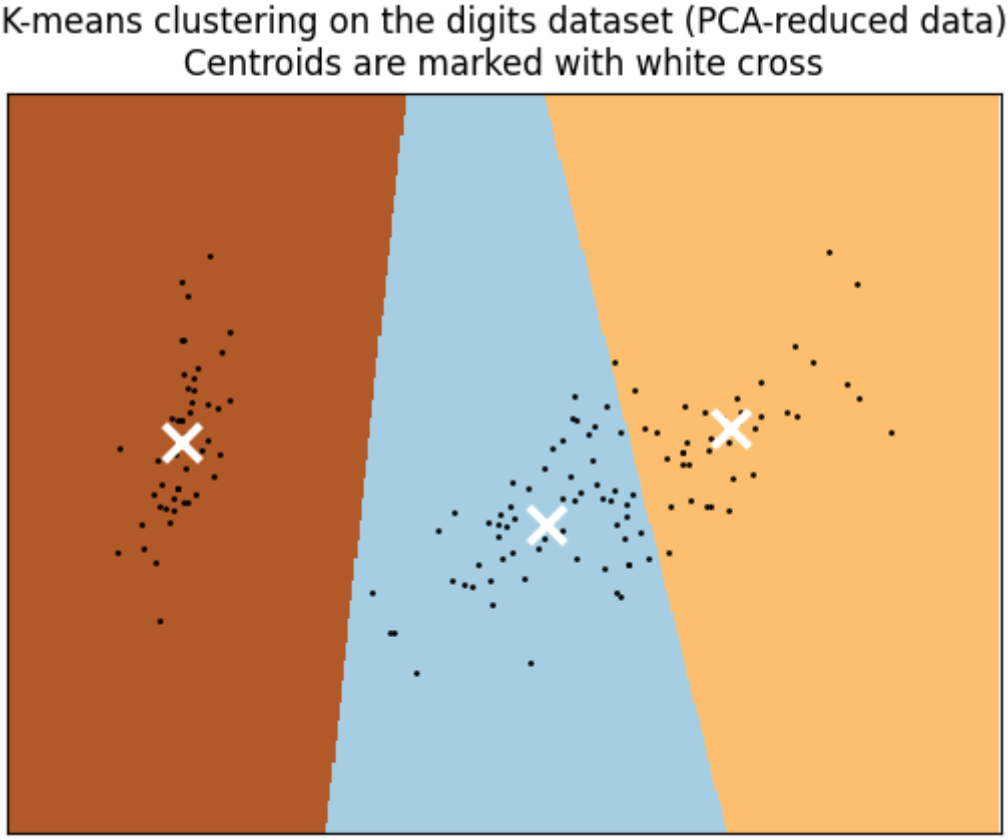






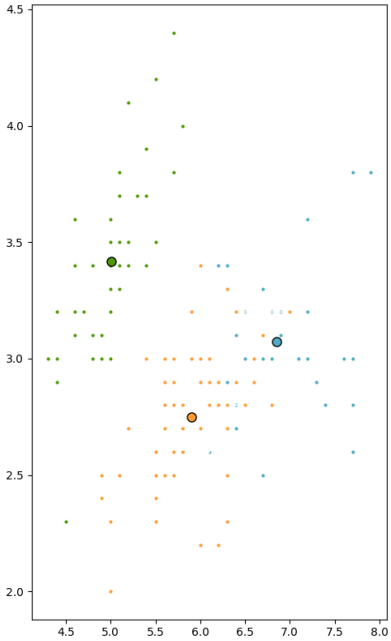
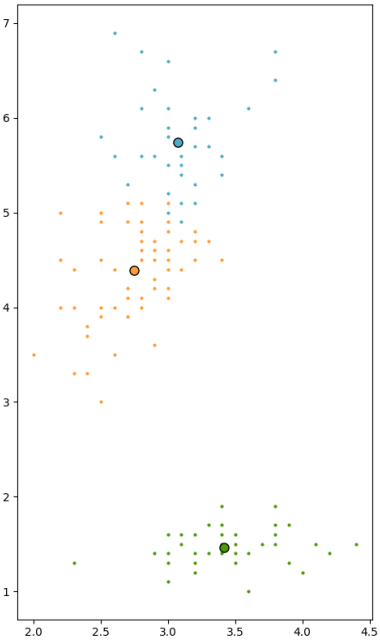
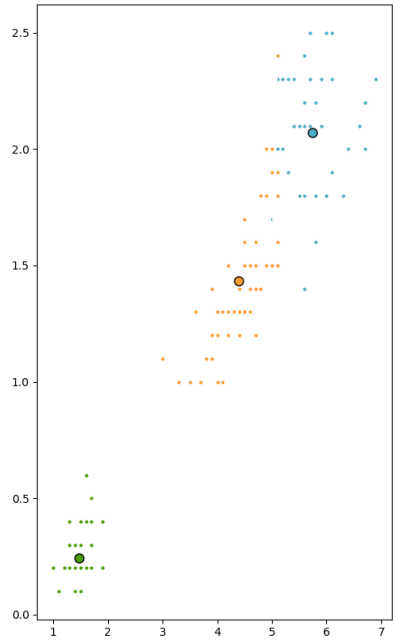
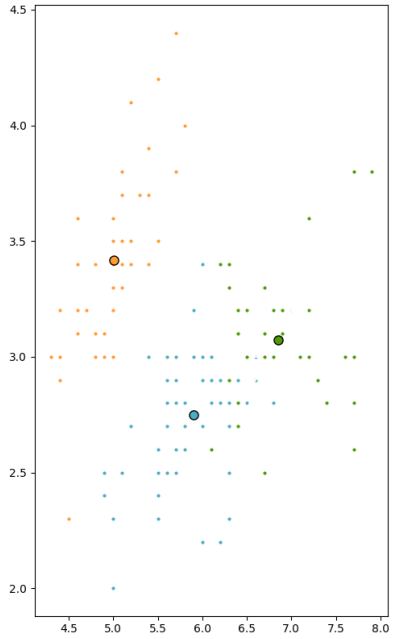
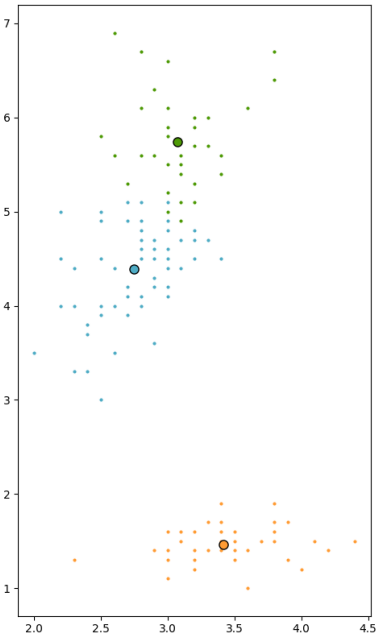
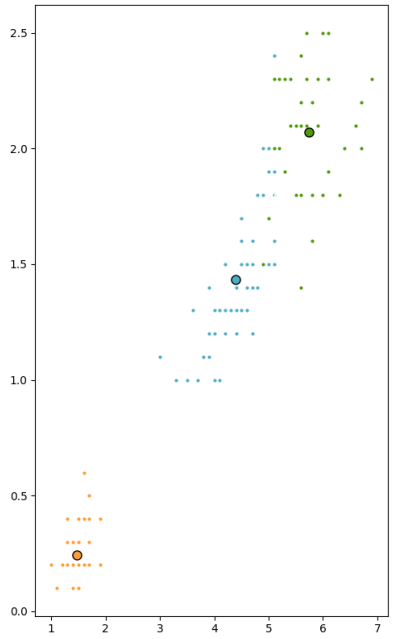
Исходя из рисунка, наилучшее разделение прошло по признакам 3 и 4. Параметр n_init в данном случае не оказал видимого влияния на результаты.

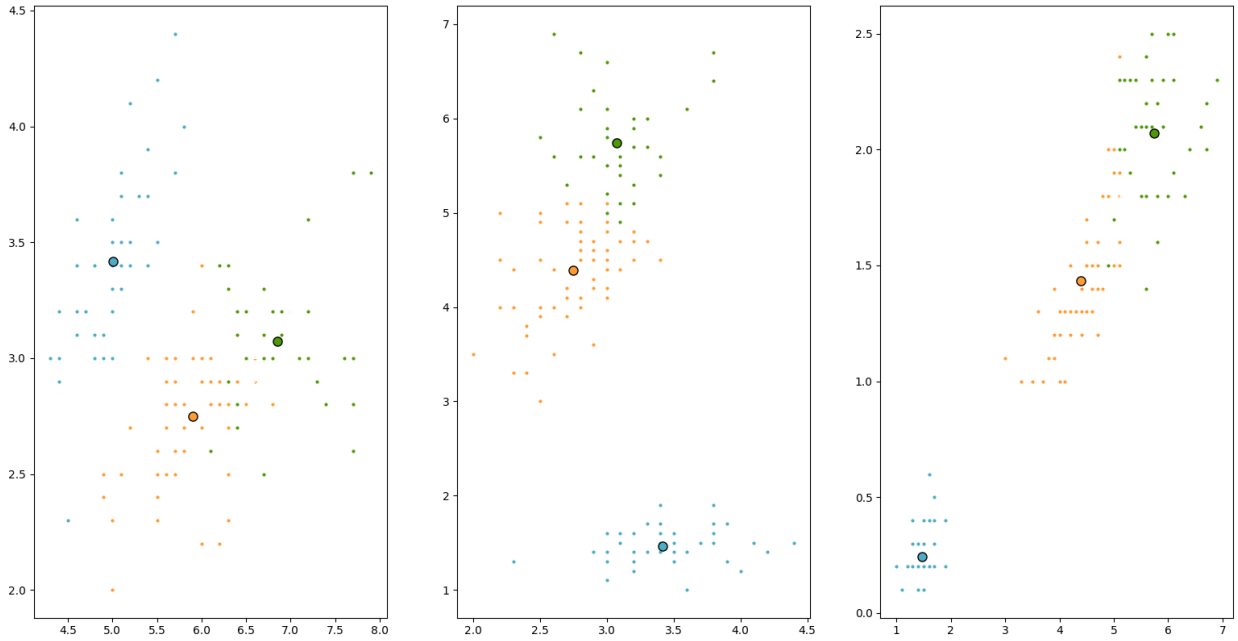
2.2. Размерность данных уменьшена до 2 используя метод главных компонент, нарисована карта для всей области значений, на которой каждый кластер занимает определенную область со своим цветом



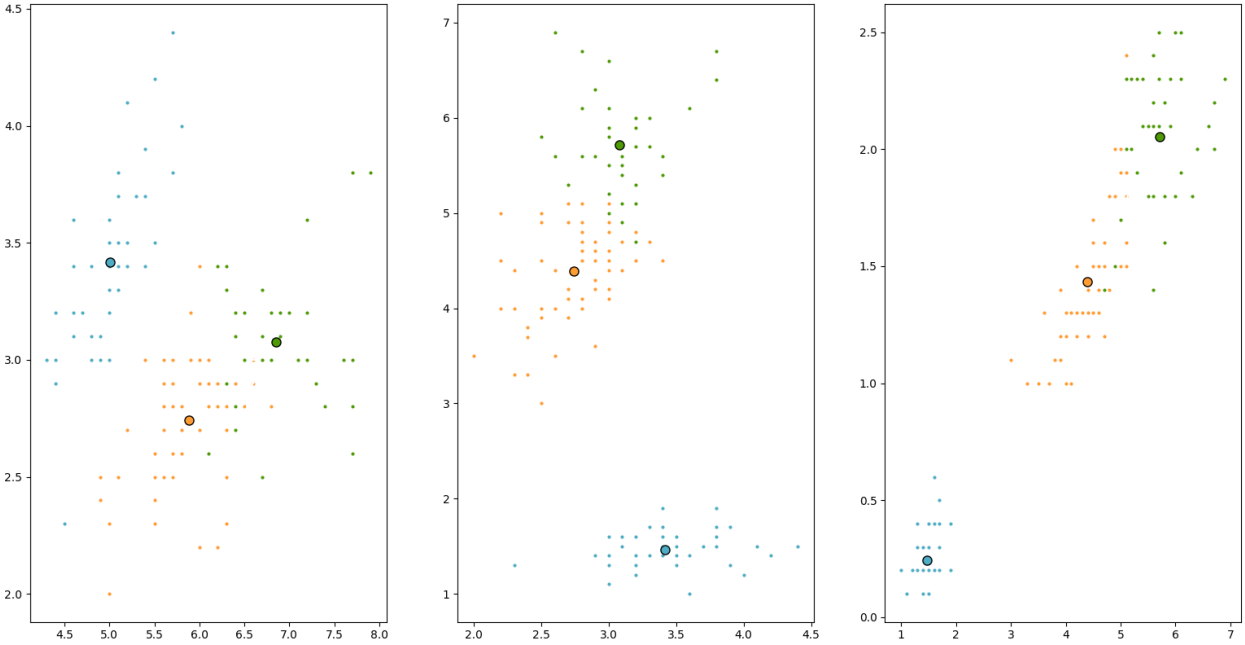
2.3. Исследование работы алгоритма k-средних при различных параметрах init

- init: random, 3 повторения

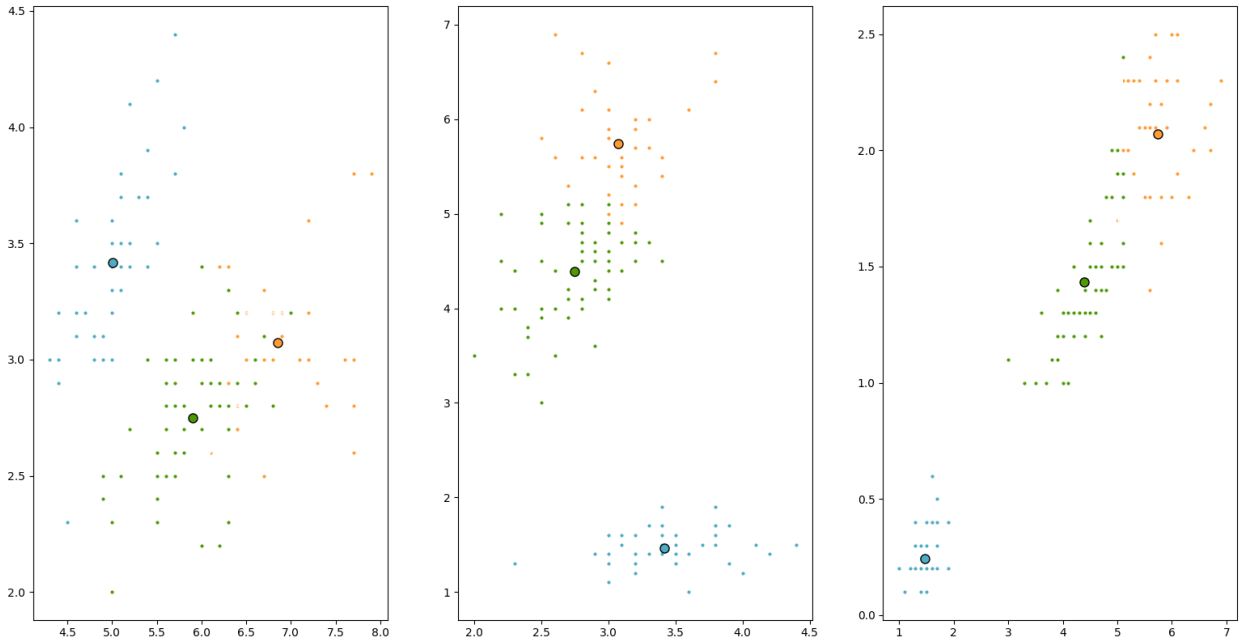









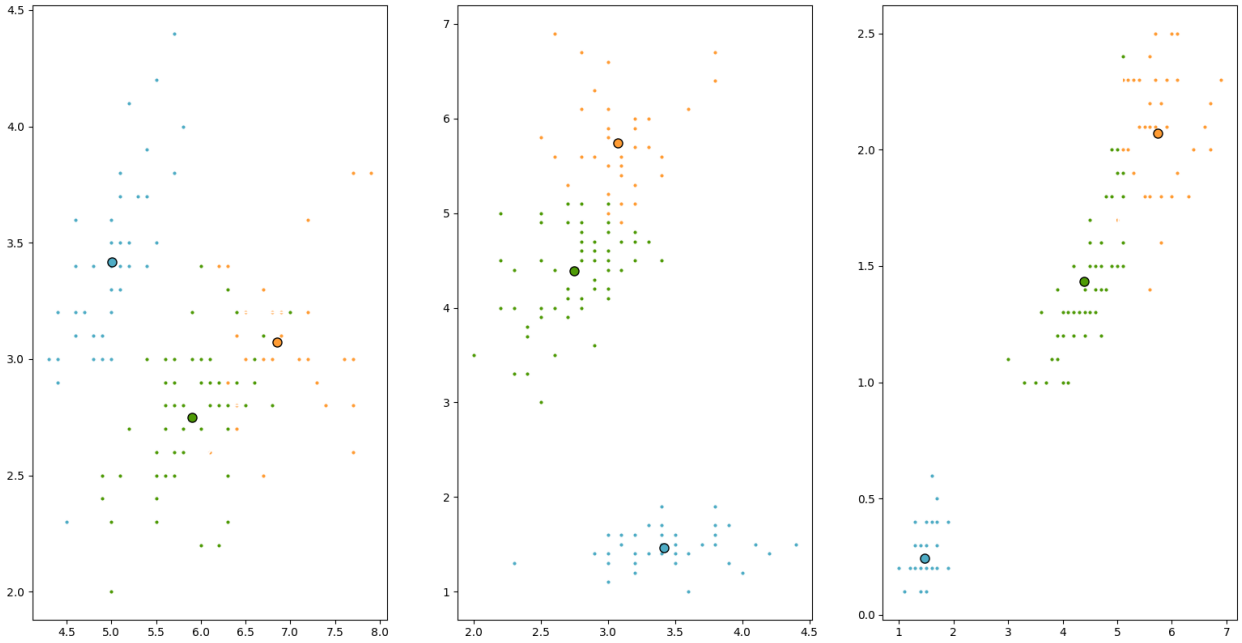
- init=np.array([[5.0, 3.4, 1.5, 0.2],[5.8, 2.2, 4.4, 1.5],[6.8, 3.1, 5.9, 2.2]])



- 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]])

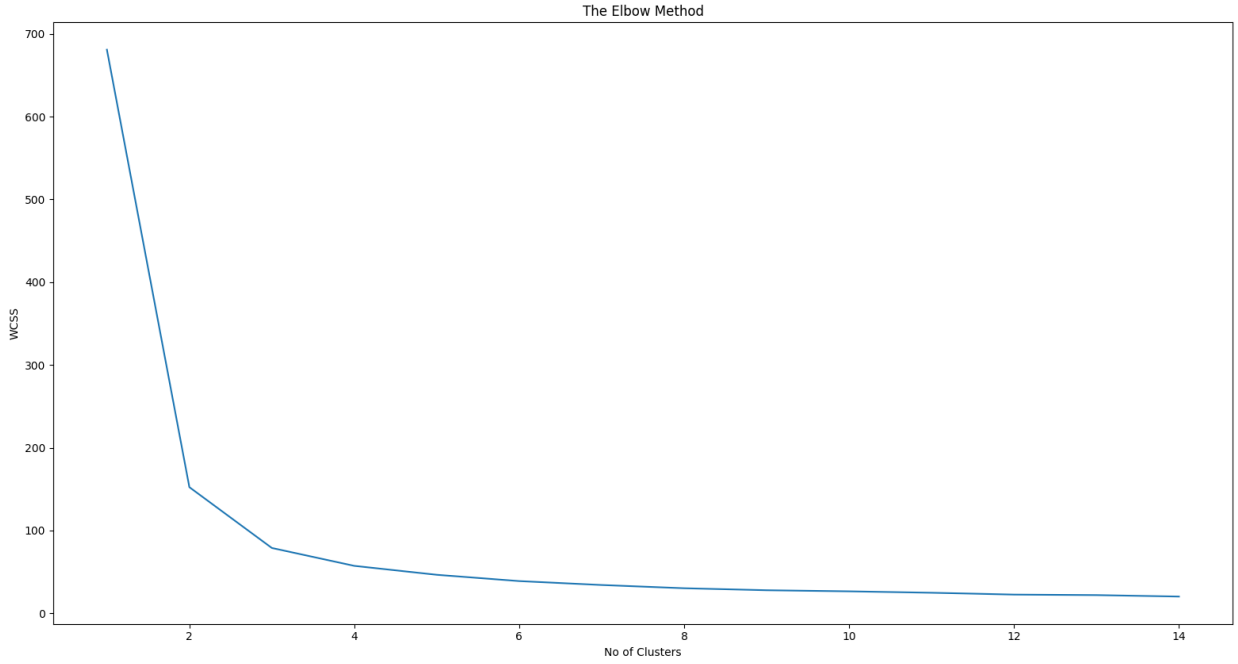


k-means++ выбирает первый центр случайно, последующий выбирается так, что вероятность выбора точки была пропорциональна вычисленному для нее квадрату расстояния.

random выбирает случайные центры.

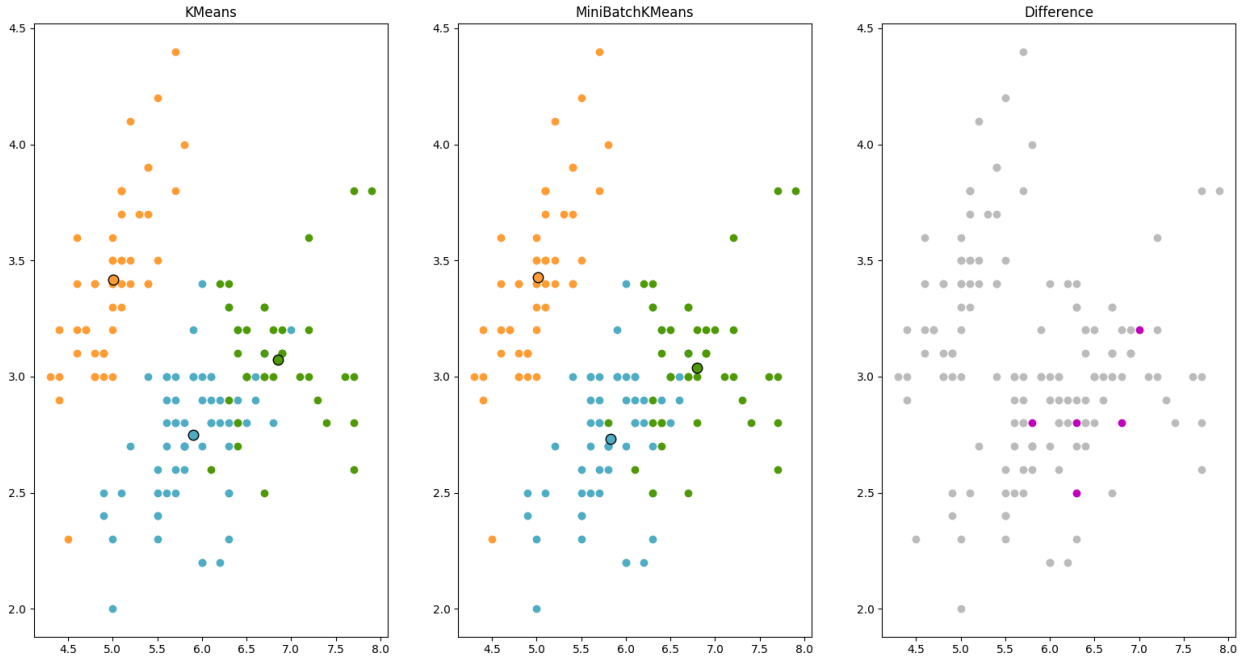
массив точек выбирает заданные центры.

2.4.Определено наилучшее количество кластеров методом локтя.



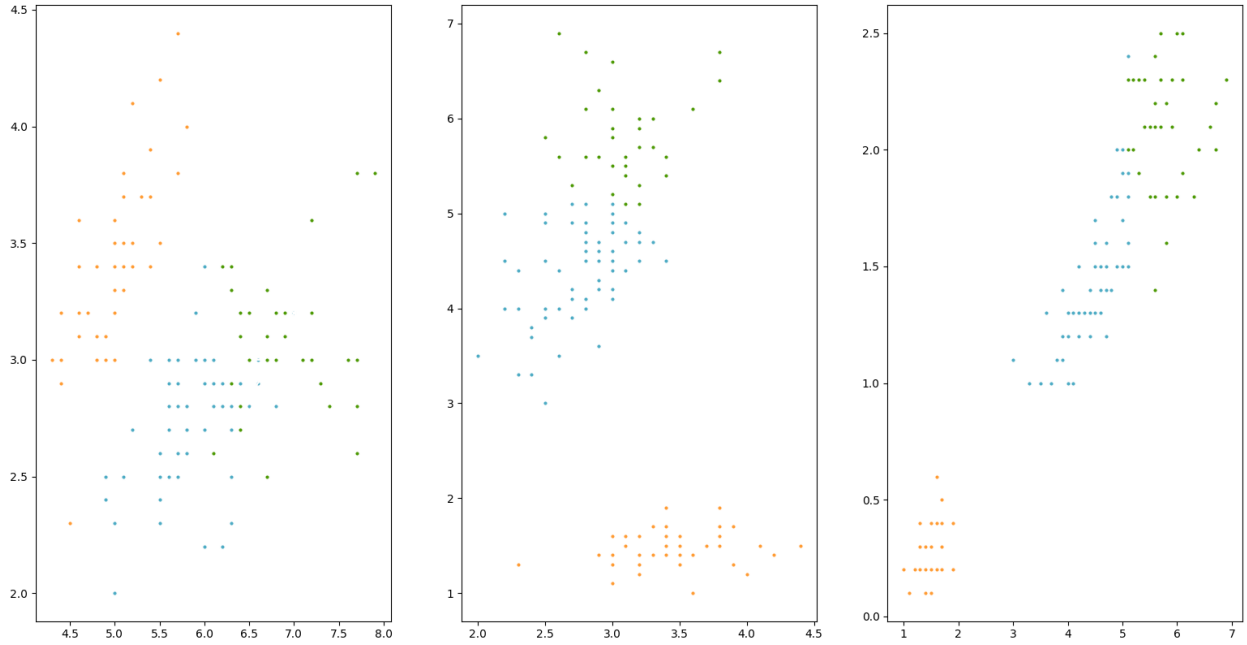
Можно сказать, что 2 кластера являются наилучшим.

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



3. Иерархическая кластеризация

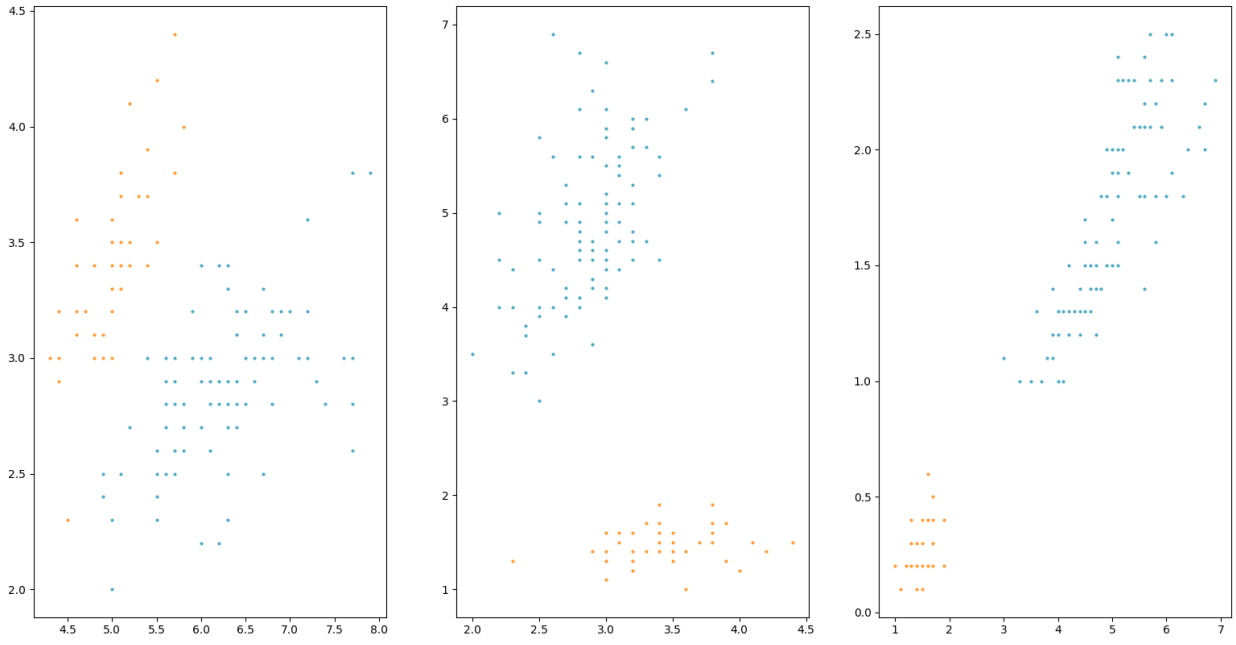
3.1. Проведена иерархическая кластеризация



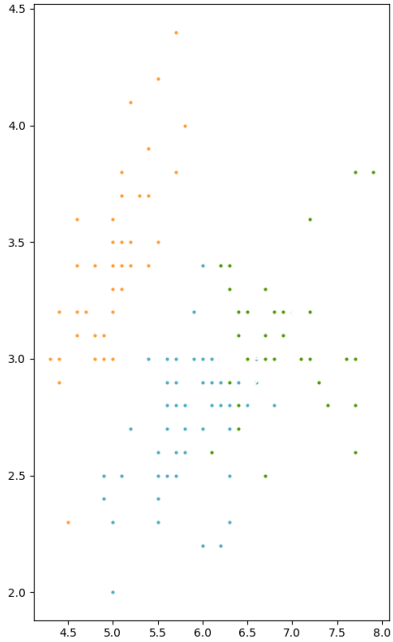
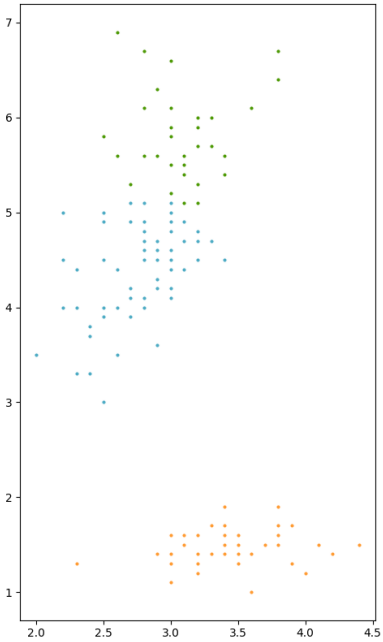
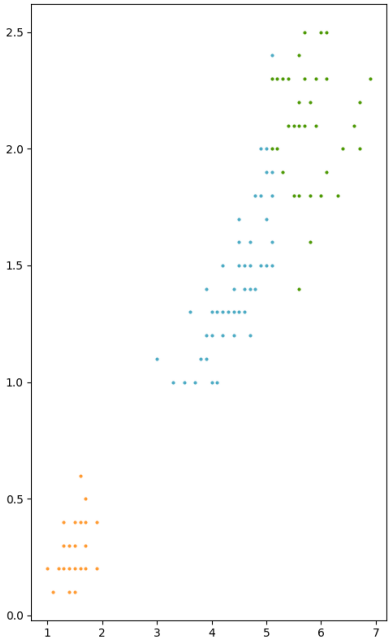
Отличие AgglomerativeClustering от KMeans в алгоритме. Первый начинает с состояния, где все точки принадлежат собственному кластеру, состоящей из одной точки, и объединяет ближайшие кластеры на основе выбранной метрики.

3.2. Проведено исследования для различного размера кластеров

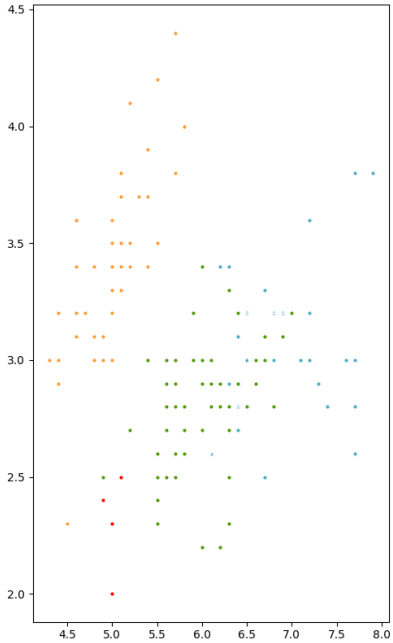
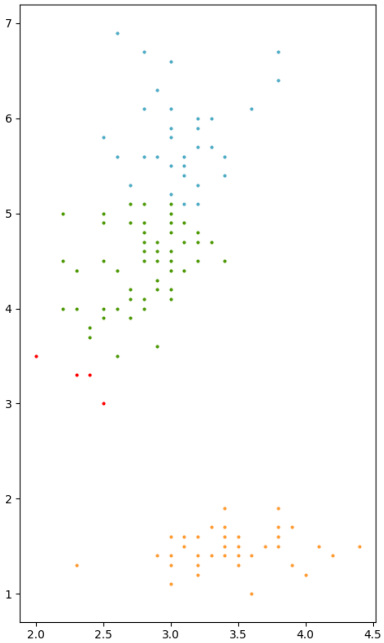
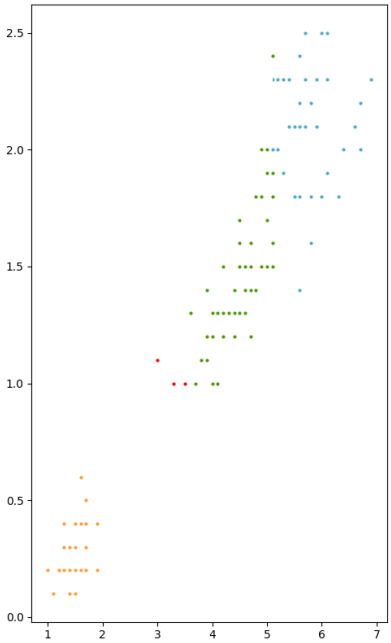
2 кластера:



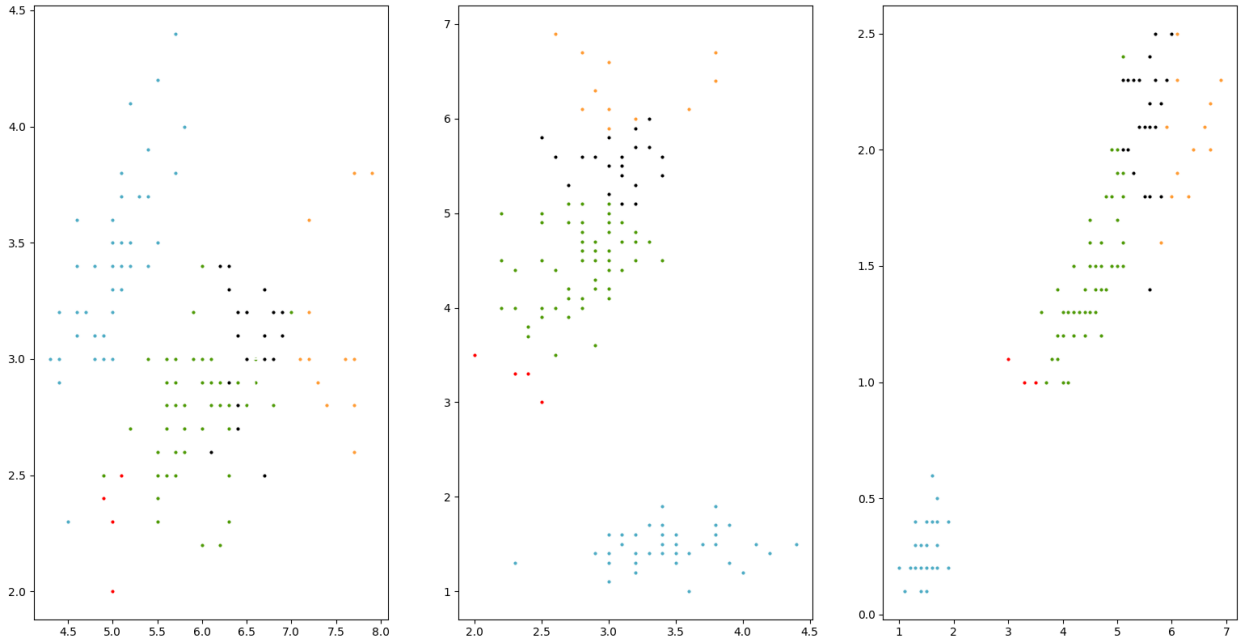
3 кластера:

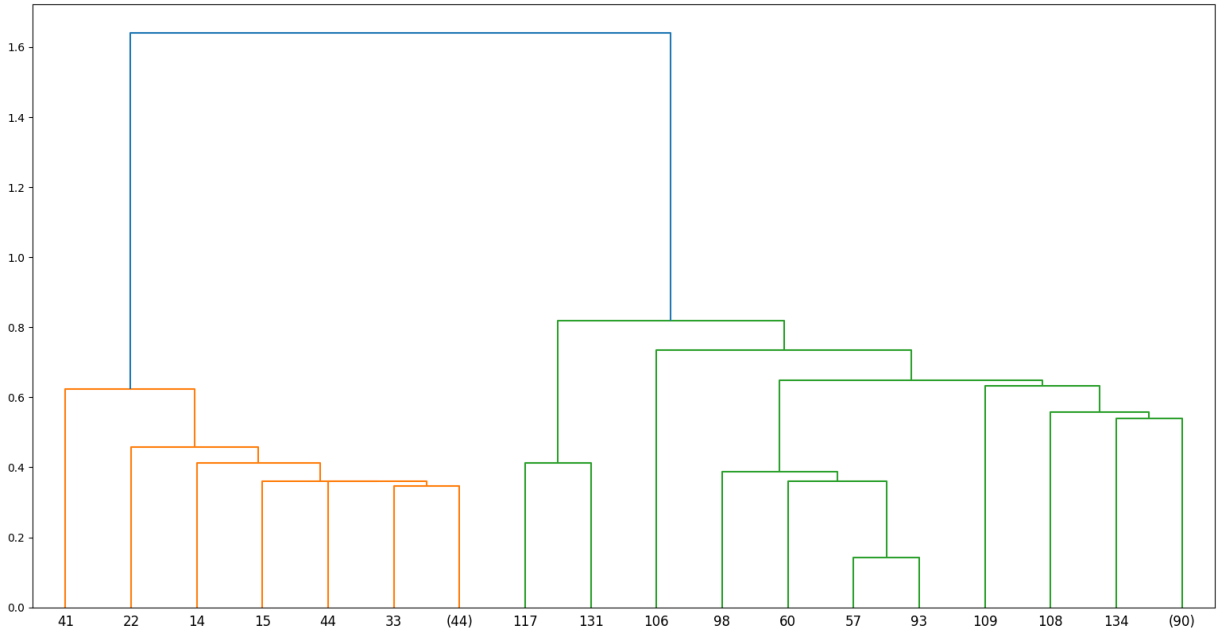
4 кластера:

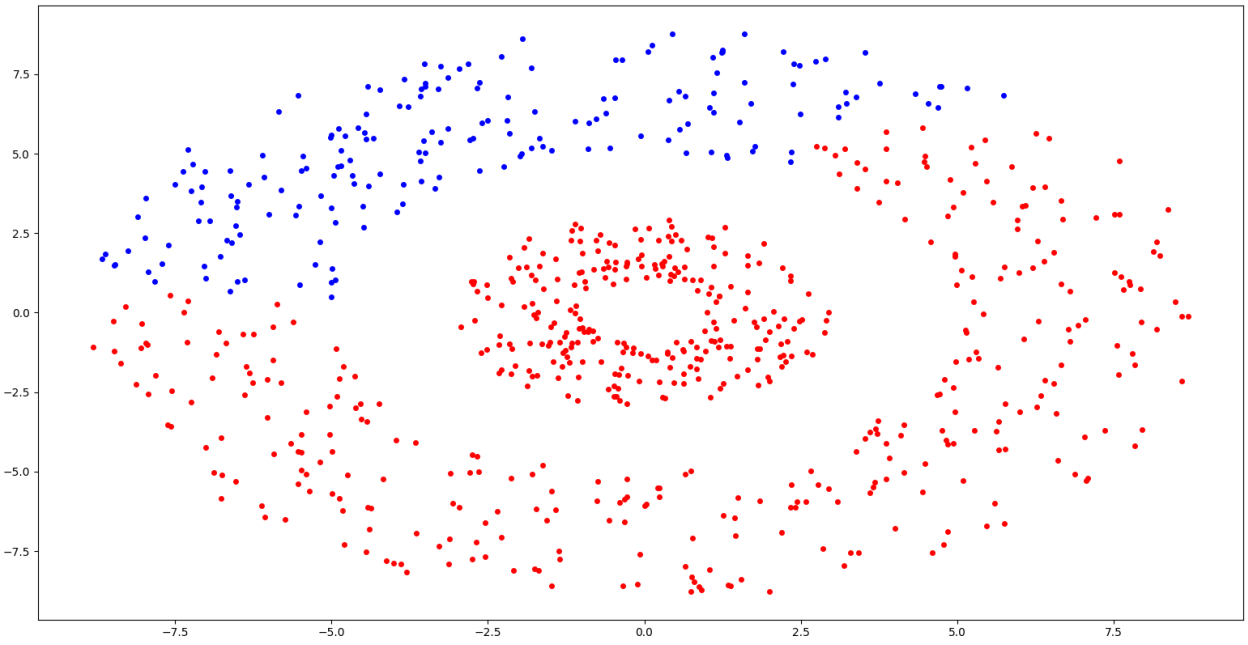
5 кластеров:



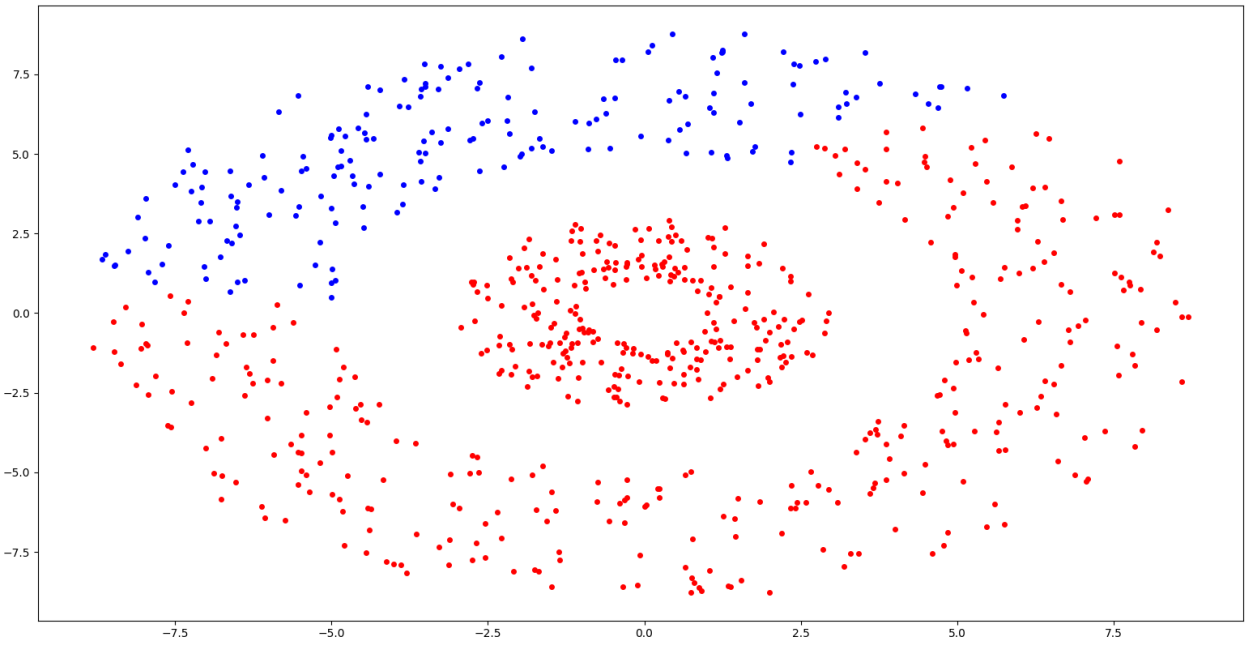
3.3. Построена дендрограмма



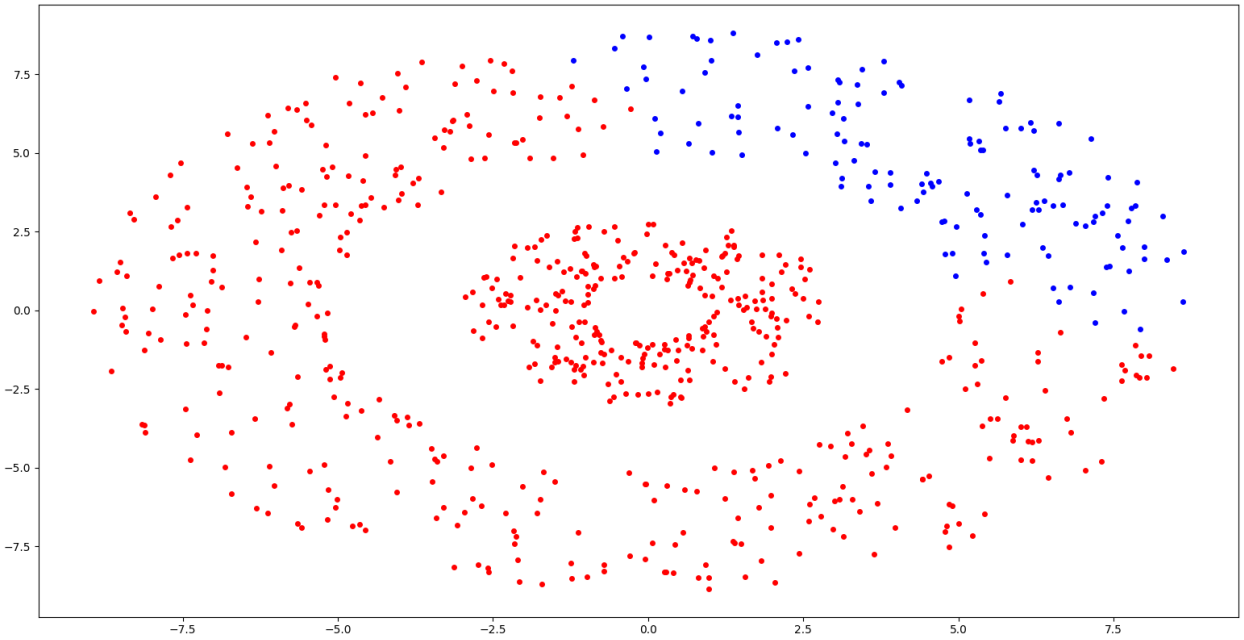
3.4. Сгенерированы случайные данные в виде двух колец. Проведена иерархическая кластеризация.



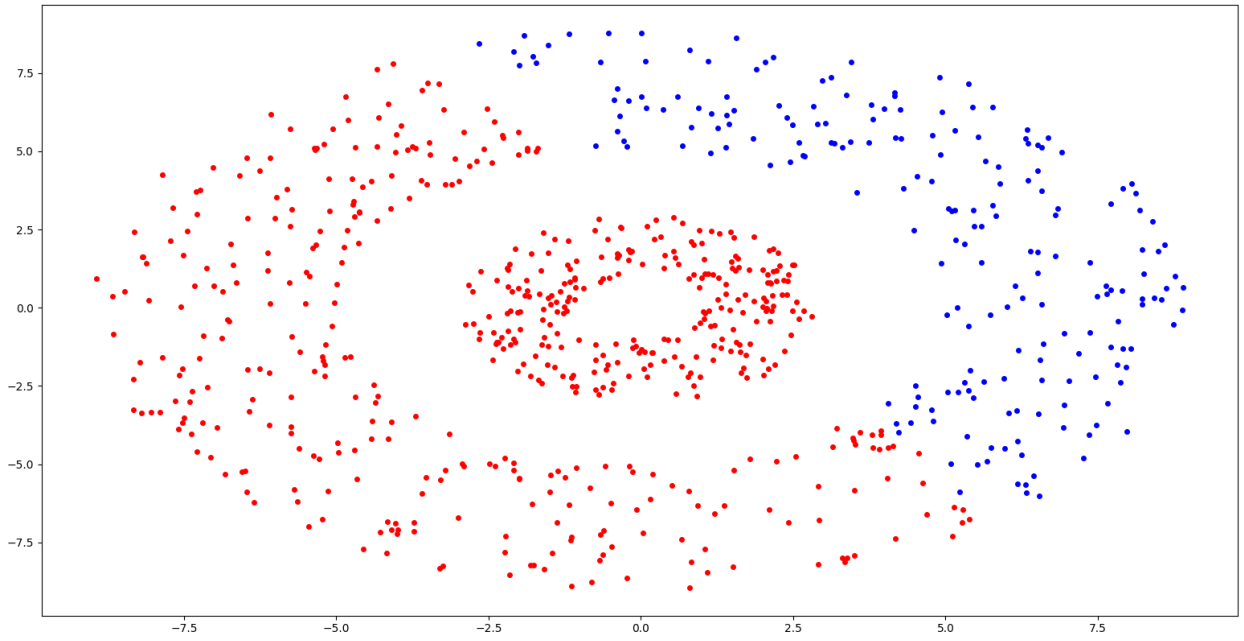
3.5. Проведены исследования для различных параметров linkage ward:



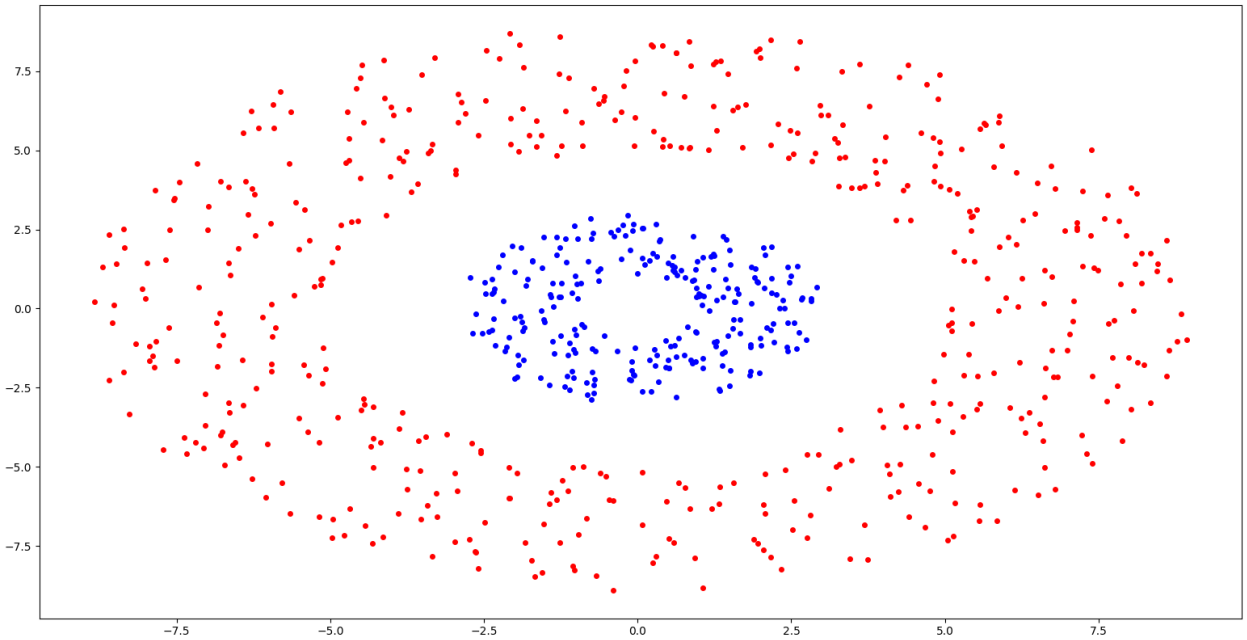
average:



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()
```