МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В. И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по лабораторной работе №5
по дисциплине «Машинное обучение»
Тема: Кластеризация (k-средних, иерархическая)

Студент гр. 1310 _____ Комаров Д. Е.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2025

Постановка задачи

Цель работы: ознакомление с методами кластеризации модуля Sklearn

Выполнение лабораторной работы

Загрузка данных

Для выполнения лабораторной работы используем набор данных «Iris». Данный набор данных содержит 150 записей о трех классах цветов. Загрузим данные в датафрейм, удалив из него метки классов. Фрагмент результата представлен на рисунке 1.

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

Рисунок 1 – Данные для кластеризации

K-means

Проведем кластеризацию данных методом k-средних, получим центры кластеров и определим, какая запись попала в какой из кластеров. Отобразим результат на графиках пар признаков. Данные графики представлены на рисунке 2.

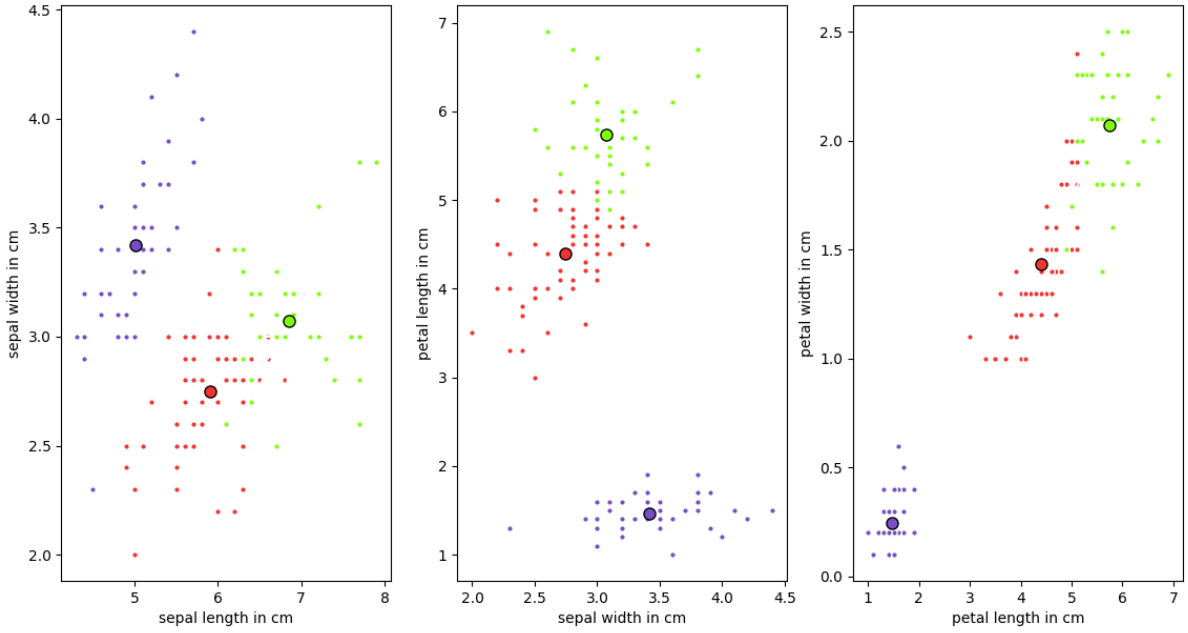

Рисунок 2 – Данные после кластеризации методом k-средних

Из графика, представленного на рисунке 2 видно, что по признакам *petal width* и *petal length* произошло наилучшее разделение данных. Значение параметра *n_init* определяет, сколько раз алгоритм будет выполнен для различных случайных начальных центров кластеров.

Уменьшим размерность данных до 2 методом главных компонент и проведем кластеризацию снова. Результат представлен на рисунке 3. Цветами обозначены области, занимаемые кластерами.

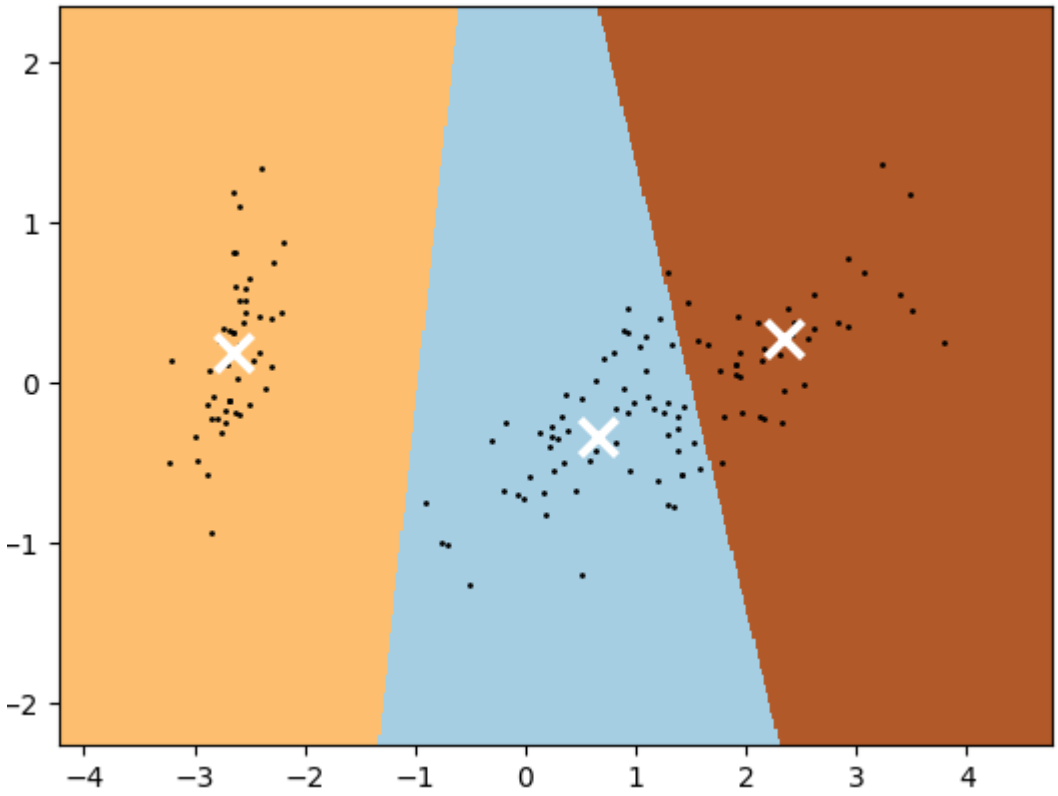

Рисунок 3 – Данные пониженной размерности после кластеризации методом k-средних

Протестируем алгоритм k-средних при различных параметрах *init*. Предыдущие результаты были получены при значении *k-means++* данного параметра. Изменим его значение на *random*. Результат представлен на рисунке 4.

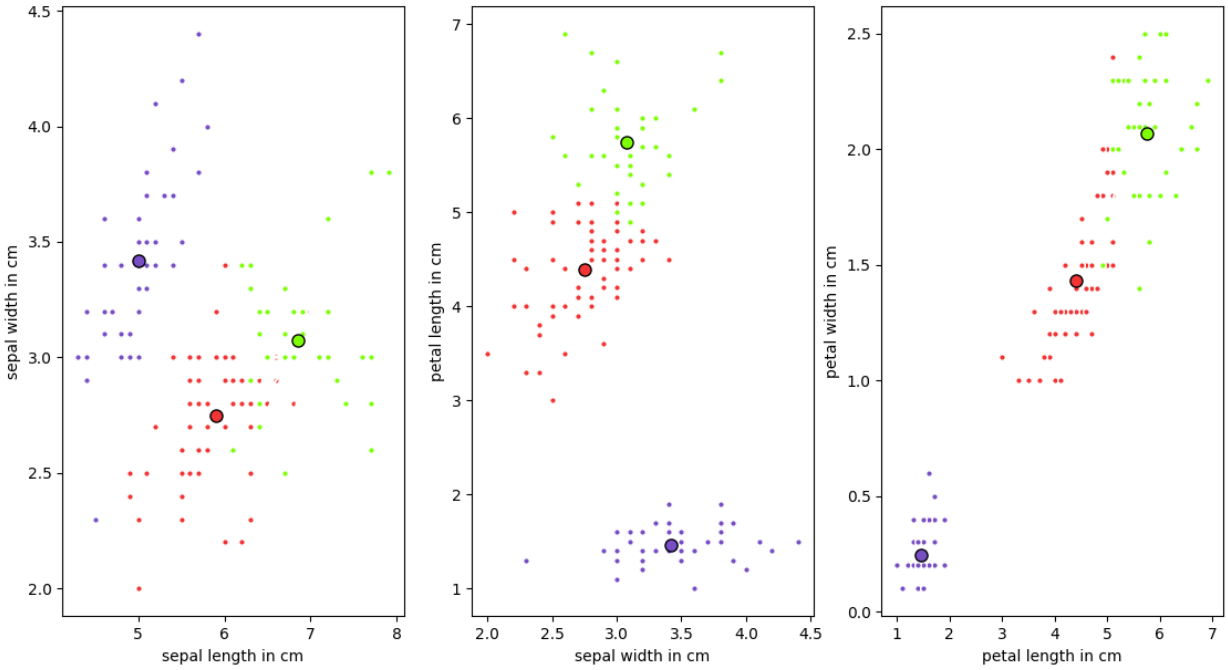

Рисунок 4 – Данные после кластеризации методом k-средних с параметром *init='random'*

Зададим начальные центры кластеров вручную. Координаты центров выберем исходя из полученных ранее центров кластеров. Результат представлен на рисунке 5.

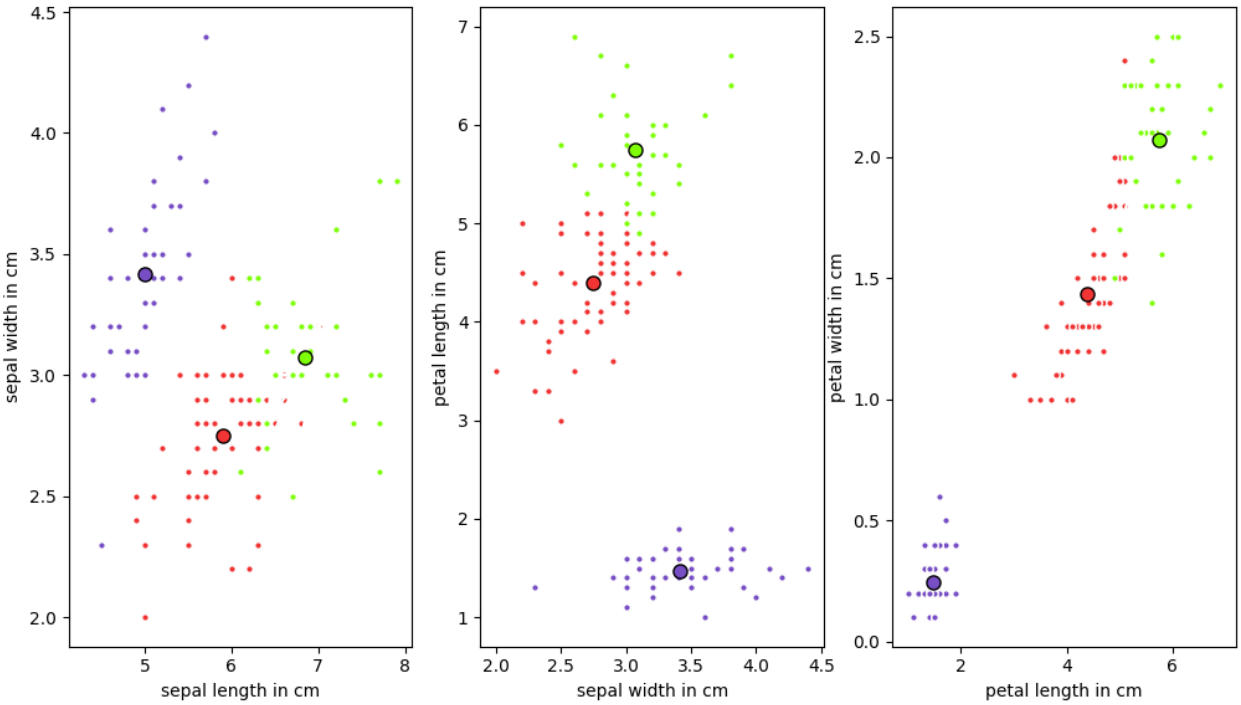

Рисунок 5 – Данные после кластеризации методом k-средних с вручную заданными начальными центрами кластеров

Определим наилучшее количество кластеров методом локтя. Данный метод заключается в проведении кластеризации для различного количества кластеров и поиске точки локтя (такой точки, после достижения которой суммарная дисперсия перестает резко падать). График, построенный для использования данного метода представлен на рисунке 6.

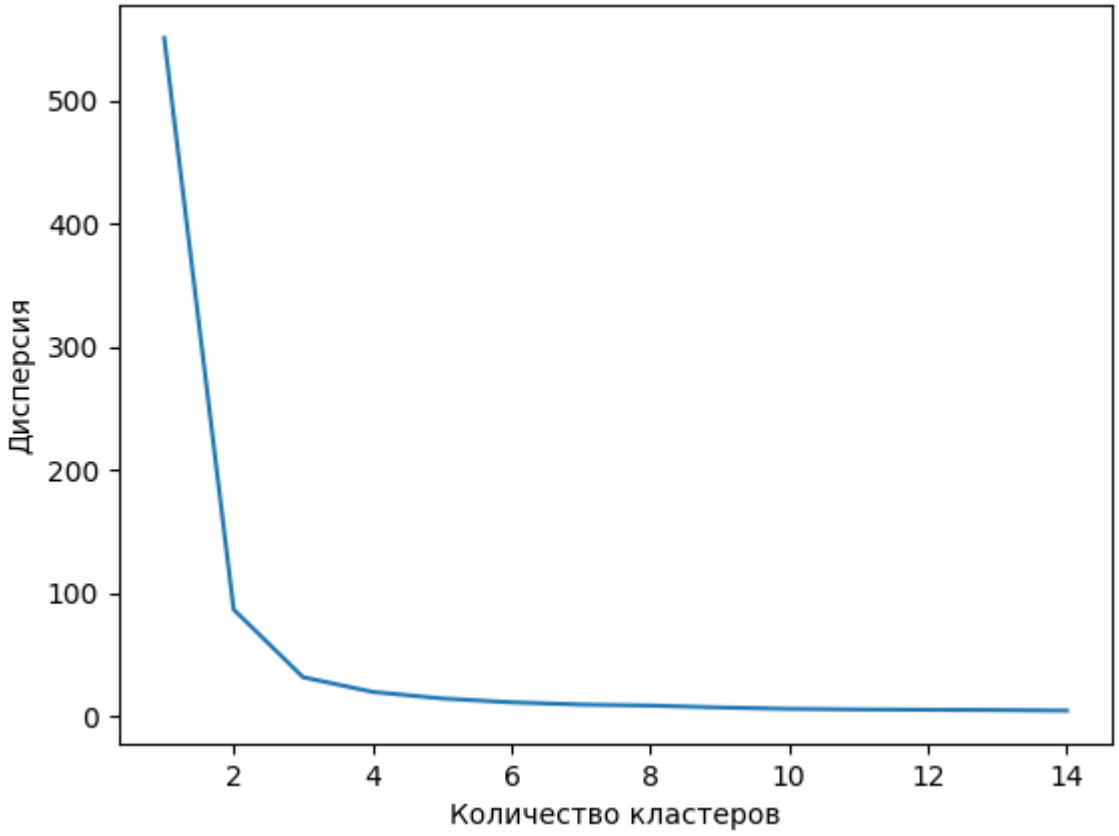

Рисунок 6 – График для поиска оптимального количества кластеров методом локтя

Как можно увидеть из рисунка 6, на данном графике локтевая точка расположена в районе 3, а следовательно, 3 – оптимальное количество кластеров.

Проведем кластеризацию, используя пакетный метод k-средних. Данный метод отличается от обычного тем, что кластеризация происходит не для всех данных сразу, а для небольших пакетов данных, которые добавляются постепенно. Такая кластеризация эффективнее работает на больших объемах данных и позволяет не проводить кластеризацию заново при добавлении

новых данных, а только обновлять предыдущий результат. Результат кластеризации пакетным методом k-средних представлен на рисунке 7.

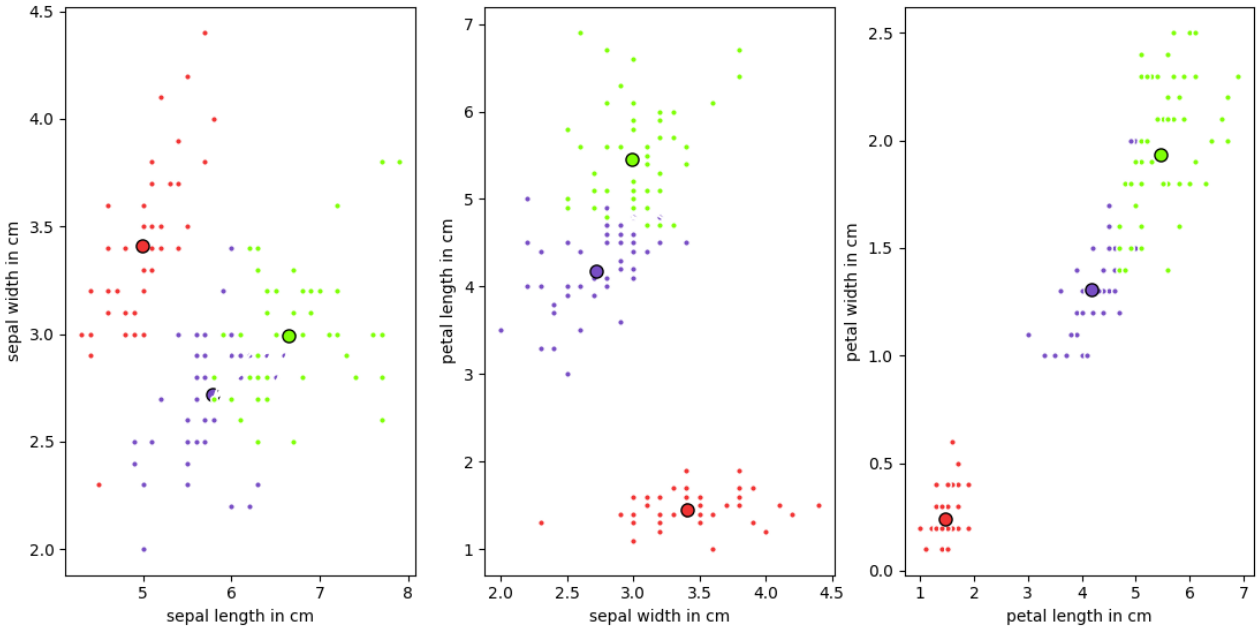

Рисунок 7 – Данные после кластеризации пакетным методом k-средних

Построим диаграмму рассеяния, на которой будут выделены точки, которые для обычного и пакетного метода k-средних попали в разные кластеры. Данная диаграмма представлена на рисунке 8.

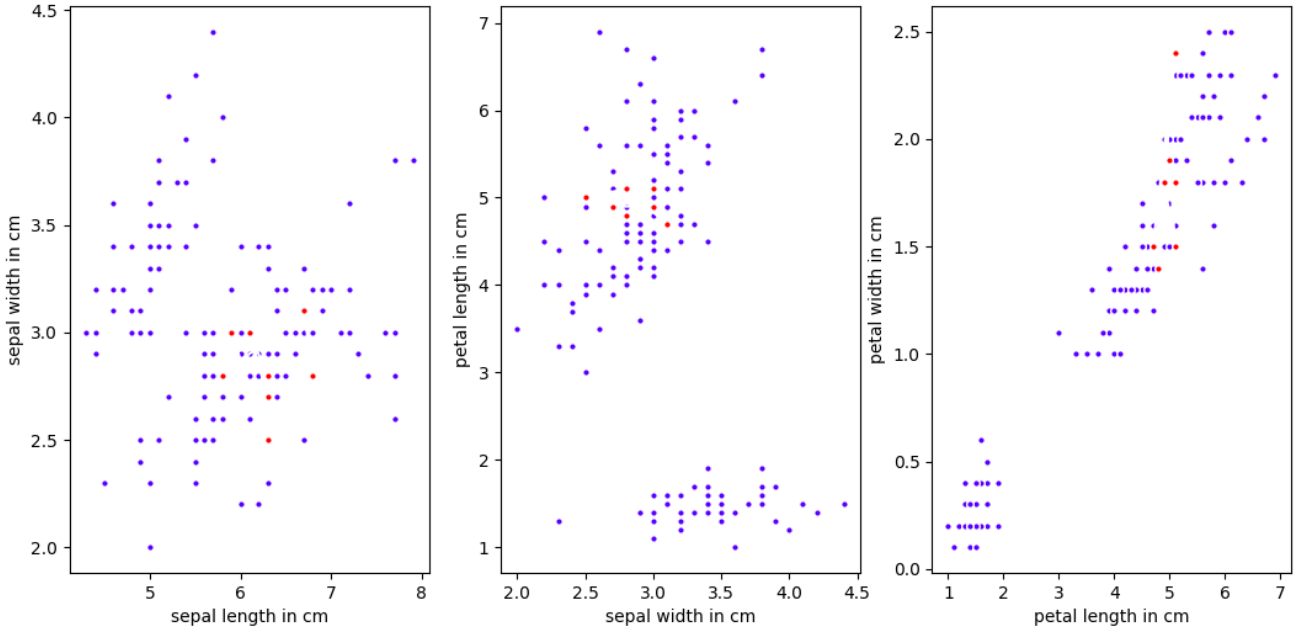

Рисунок 8 – Сравнения результатов пакетного и обычного k-means

Синим отмечены точки, попавшие в один и тот же кластер при использовании обоих методов, красным – попавшие в разные кластеры. Как

можно увидеть из рисунка 8, большинство точек при использовании обоих методов попали в одинаковые кластеры. Разница наблюдается лишь для точек, лежащих на границах кластеров.

Иерархическая кластеризация

Проведем иерархическую кластеризацию на тех же данных. Суть данного метода заключается в том, что в нем каждое наблюдение изначально заносится в отдельный кластер. Затем схожие между собой кластеры объединяются до тех пор, пока количество кластеров не станет равным заданному. Результат представлен на рисунке 9.

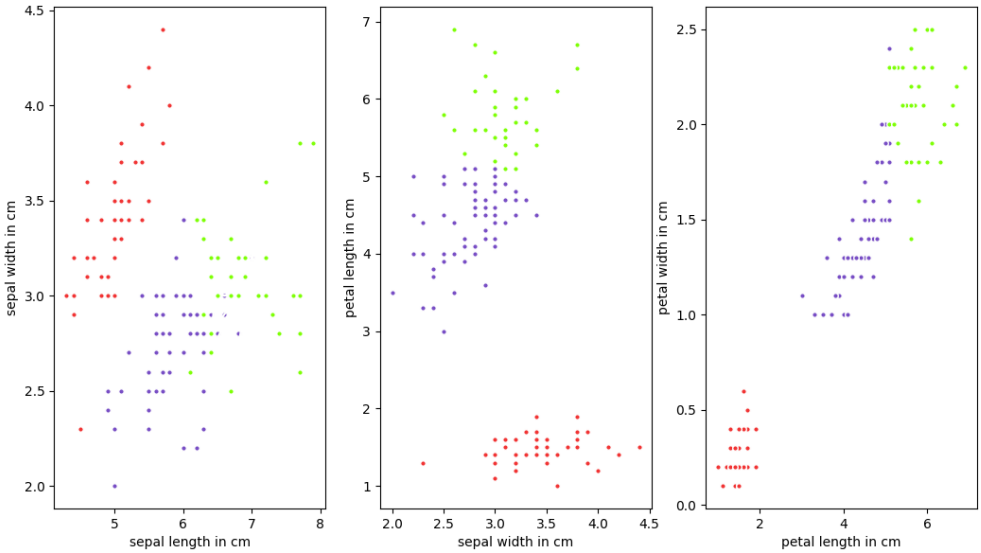

Рисунок 9 – Данные после иерархической кластеризации для 3 кластеров

Изменим количество кластеров до 2. Результат представлен на рисунке 10.

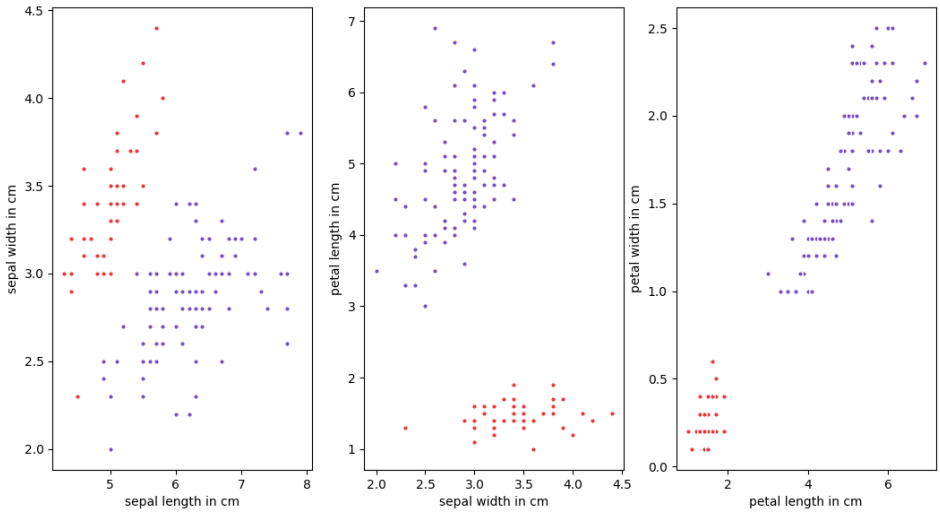

Рисунок 10 – Данные после иерархической кластеризации для 2 кластеров

Изменим количество кластеров до 5. Результат представлен на рисунке 11.

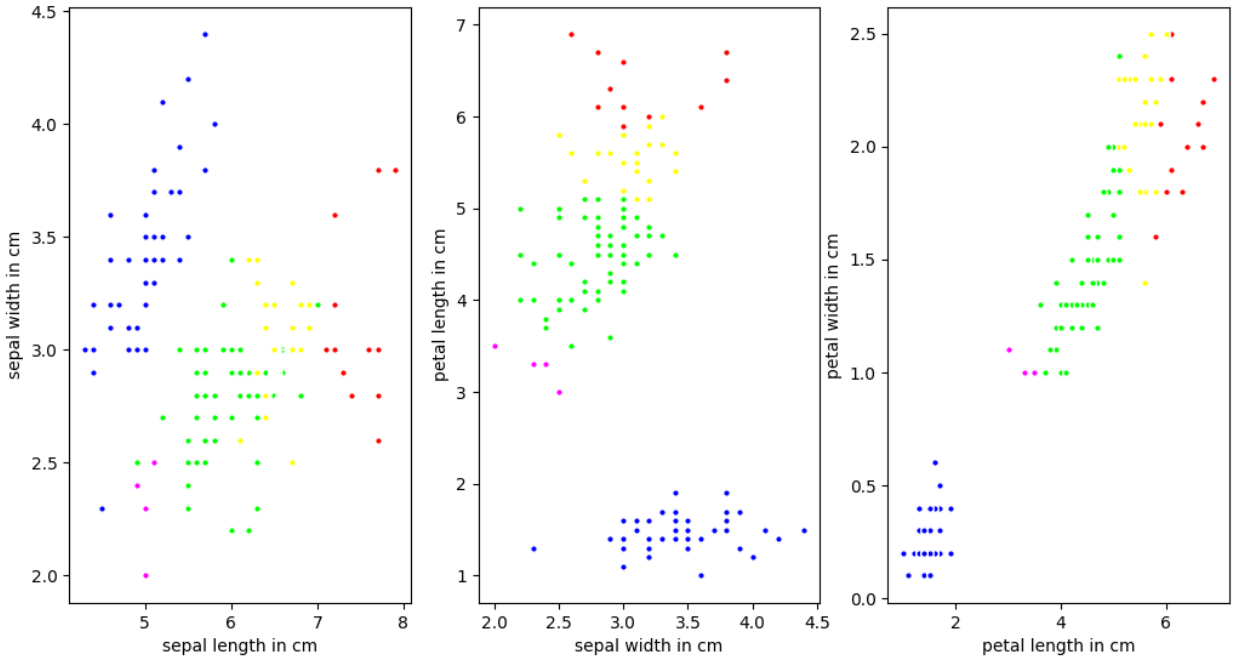

Рисунок 11 – Данные после иерархической кластеризации для 5 кластеров

Отобразим результат иерархической кластеризации на дендограмме. Данная дендограмма представлена на рисунке 12. Дендограмма построена до 6 уровня.

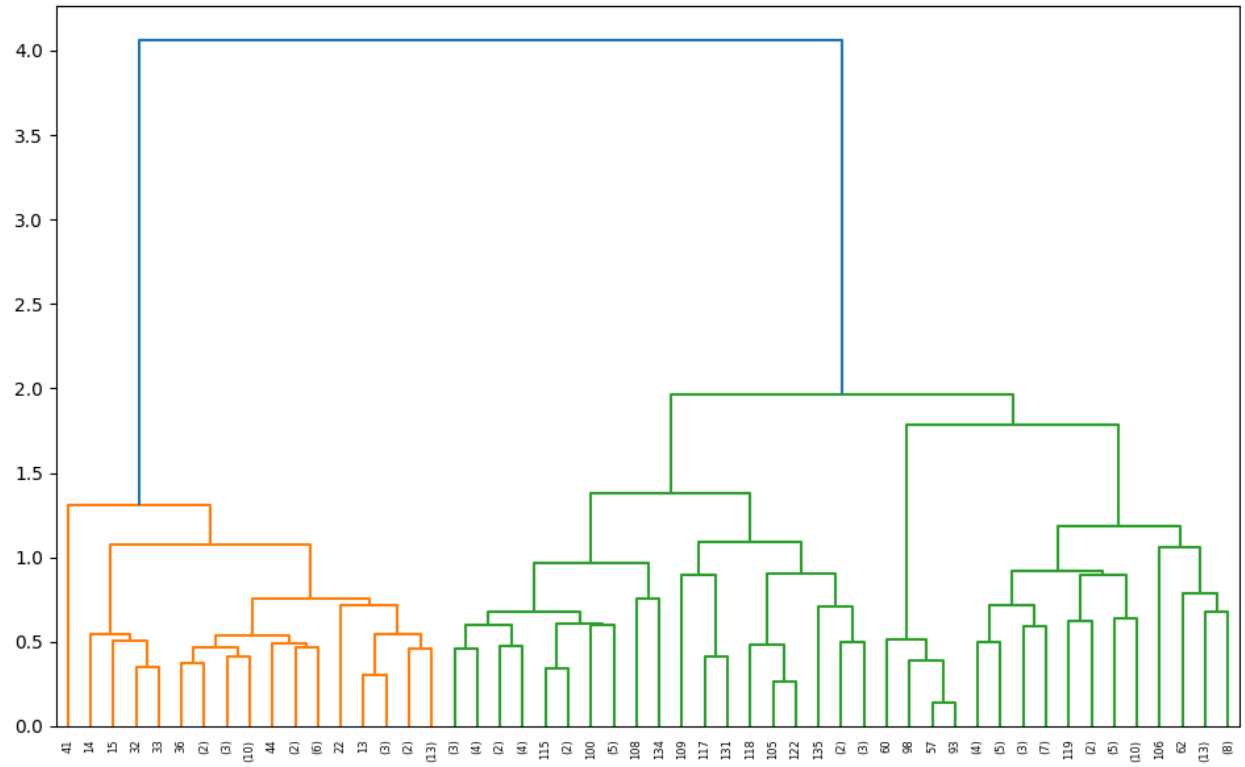

Рисунок 12 – Дентрограмма иерархической кластеризации

Листьями данной дендограммы являются кластеры. Дендограмма показывает, как кластеры объединяются друг с другом, пока все не объединятся в один кластер.

Сгенерируем случайные данные в виде двух колец и проведем их иерархическую кластеризацию для двух кластеров. Результат представлен на рисунке 13.

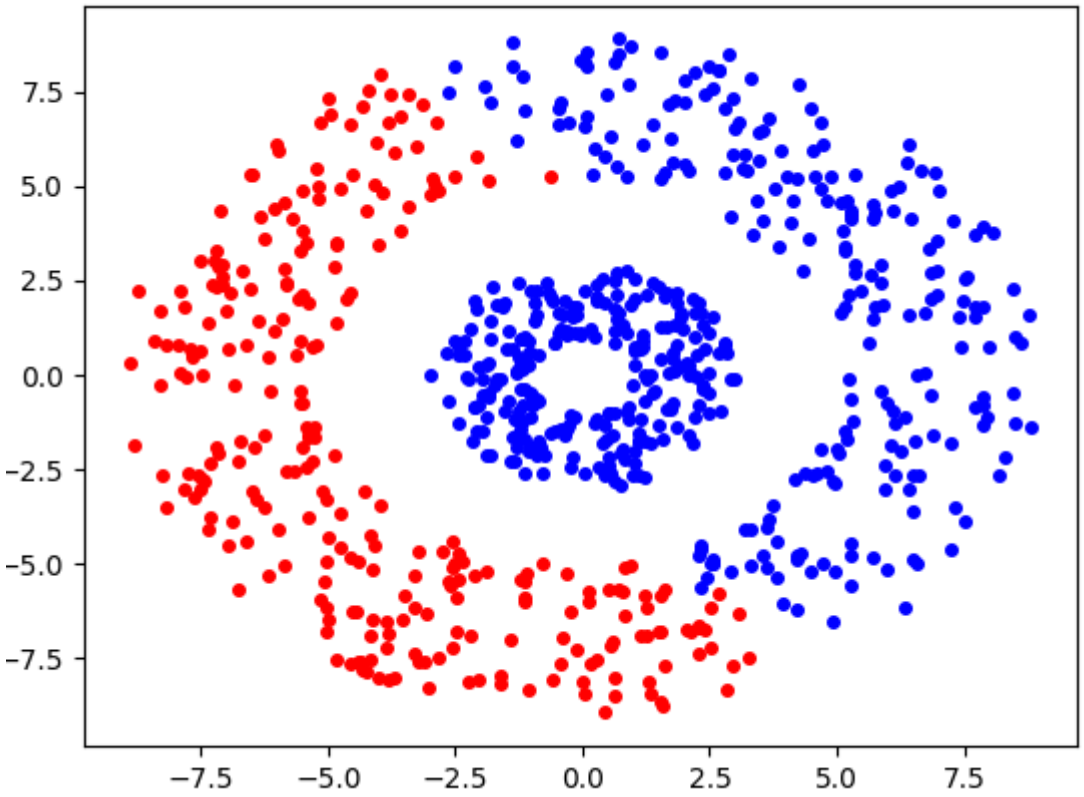

Рисунок 13 – Результат иерархической кластеризации двух колец данных при значении *linkage='ward'*

Как можно увидеть из рисунка 13, желаемый результат, при котором 2 кольца были бы отнесены к двум различным кластерам достигнут не был. Протестируем иерархическую кластеризацию при различных параметрах *linkage*. По умолчанию используется параметр *ward*. При нем 2 кластера объединяются так, чтобы прирост дисперсии внутри кластера был минимален. Изменим данный параметр на *average*. При нем 2 объединяемых кластера имеют наименьшее среднее расстояние между их точками. Результат представлен на рисунке 14.

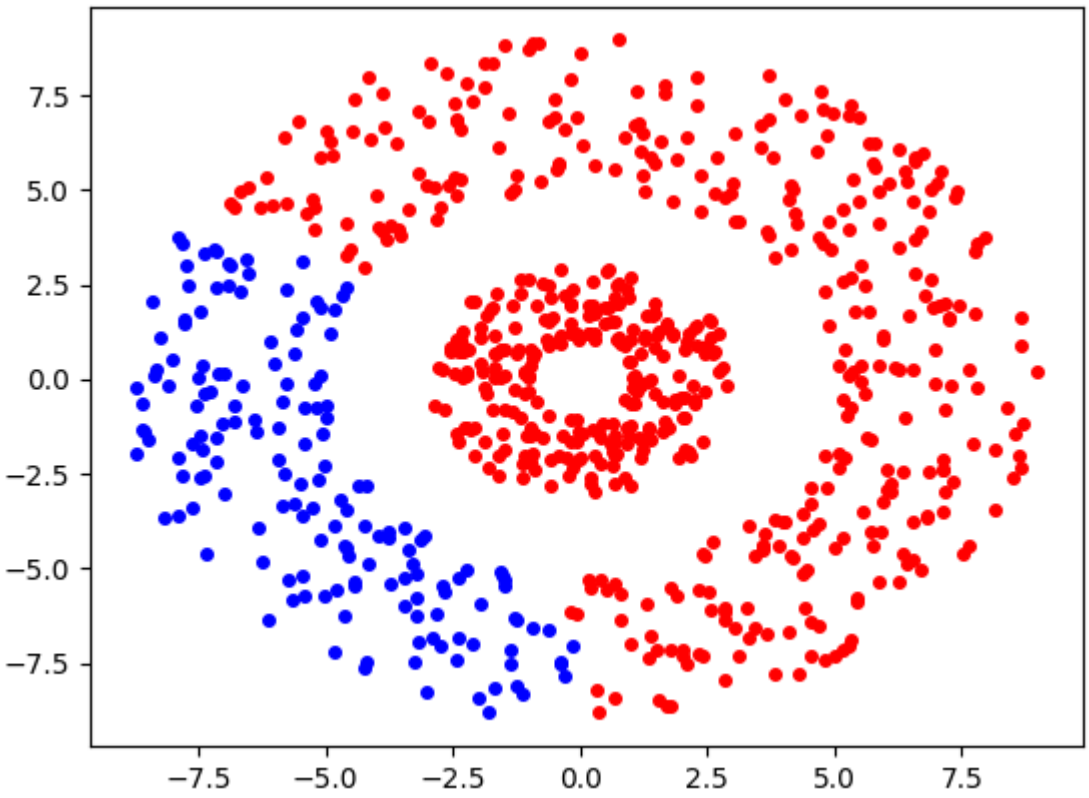

Рисунок 14 – Результат иерархической кластеризации двух колец данных при значении *linkage='average'*

Изменим значение *linkage* на *complete*. При нем объединяются 2 кластера, имеющие наименьшее расстояние между самыми удаленными их точками. Результат представлен на рисунке 15.

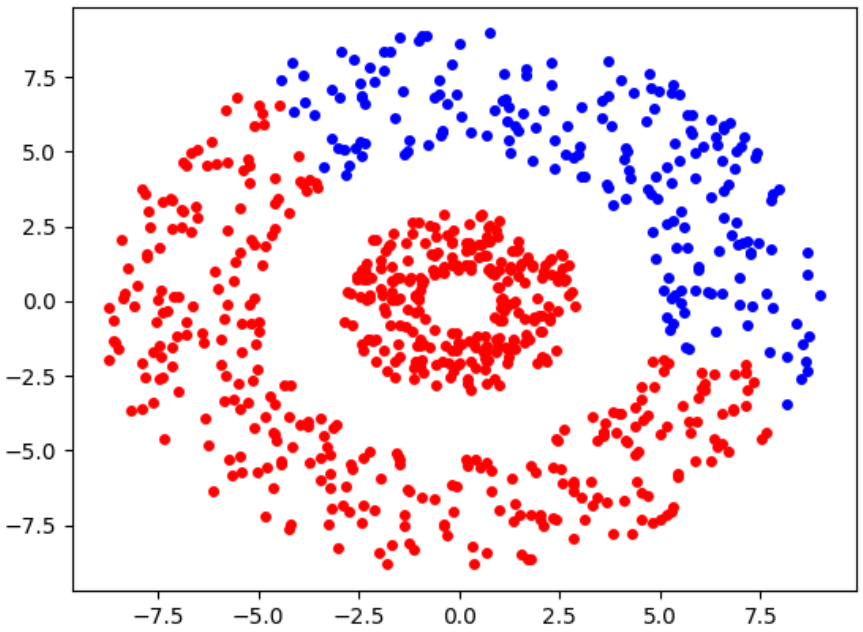

Рисунок 15 – Результат иерархической кластеризации двух колец данных при значении *linkage='complete'*

Изменим значение *linkage* на *single*. При нем объединяются 2 кластера, имеющие минимальное расстояние между их наблюдениями. Результат представлен на рисунке 16.

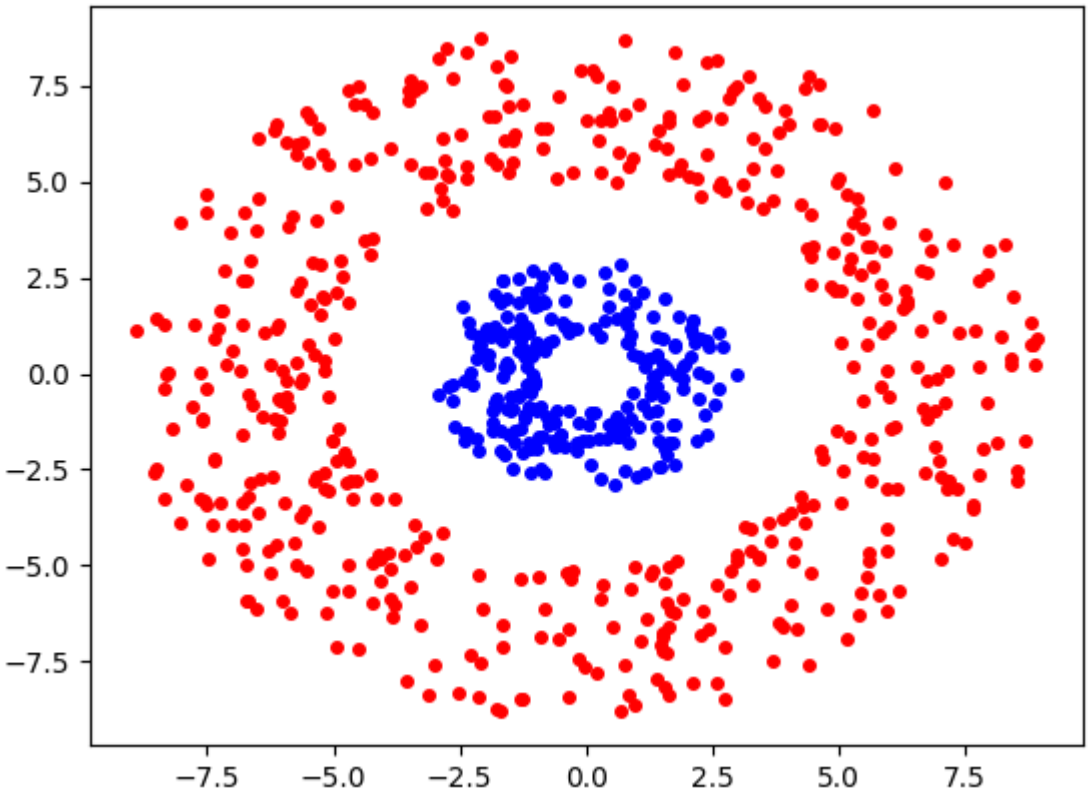

Рисунок 16 – Результат иерархической кластеризации двух колец данных при значении *linkage='single'*

Как можно заметить, при *linkage* равному *single* было достигнуто желаемое разделение данных.

Выводы

В ходе выполнения лабораторной работы. Было проведено ознакомление с методами кластеризации модуля Sklearn.

При помощи алгоритма k-средних была проведена кластеризация исходных данных на 3 кластера. Алгоритм был протестирован для различных начальных способов задания начальных центров кластеров (случайно, вручную и методом *k-means++*). Также данным методом была проведена кластеризация данных после понижения размерности методом главных компонент. Было определено оптимальное количество кластеров для данного набора данных методом локтя, которое оказалось равно 3. Также была

проведена кластеризация пакетным методом k-средних. Данная модификация отличается от обычного k-средних что кластеризация происходит не для всех данных сразу, а для небольших пакетов данных, которые добавляются постепенно. Было проведено сравнение результатов обычного и пакетного k-средних. Почти все наблюдения (за исключением находящихся на границах кластеров) обоими алгоритмами были разделены на кластеры одинаково.

Также была проведена иерархическая кластеризация тех же данных. Иерархическая кластеризация, в отличие от k-средних, изначально определяет каждое наблюдение в отдельный кластер, а затем объединяет их, пока количество кластеров не станет равно заданному. Иерархическая кластеризация была протестирована на различном количестве кластеров. Также была построена дендограмма, показывающая, как именно кластеры объединяются друг с другом.

Был и сгенерированы случайные данные в виде 2-х колец. Была проведена иерархическая кластеризация этих данных при различных параметрах *linkage*. Было выяснено, что для данного случая оптимальным является параметр *linkage*, равный *single*.

Код программы, написанной для выполнения лабораторной работы представлен в приложении А.

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММЫ

```
import pandas as pd
import numpy as np
from sklearn import cluster
from sklearn import metrics
from sklearn import preprocessing
from sklearn import decomposition
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram
import random
import math

def
plotClusters(no_labeled_data,names,k_means_labels,k_means_cluster_centers):
    f, ax = plt.subplots(1, 3)
    colors = ["#784EC5", "#F13434", "#7FFF07"]
    for i in range(len(k_means_cluster_centers)):
        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_memb
ers][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)
            ax[j].set_xlabel(names[j])
            ax[j].set_ylabel(names[j+1])
    plt.show()

def plotHierClusters(no_labled_data,labels):
    f, ax = plt.subplots(1, 3)
    colors = ["#0000FF", "#FF0000", "#00FF00","#FF00FF","#FFFF00"]
    for i in range(5):
        my_members = labels == i
        for j in range(3):
             ax[j].plot(no_labeled_data[my_members][j],no_labeled_data[my_memb
ers][j+1], 'w',markerfacecolor=colors[i], marker='o', markersize=4)
            ax[j].set_xlabel(names[j])
            ax[j].set_ylabel(names[j+1])
    plt.show()

def plotPCAClusters(pca_data,k_means_cluster_centers):
    h = 0.02
    x_min, x_max = pca_data[:, 0].min() - 1, pca_data[:, 0].max() + 1
    y_min, y_max = pca_data[:, 1].min() - 1, pca_data[:, 1].max() + 1
     xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max,
h))
    Z = k_means.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(pca_data[:, 0], pca_data[:, 1], "k.", markersize=2)
    centroids = k_means_cluster_centers
```

```
    plt.scatter(
        centroids[:, 0],
        centroids[:, 1],
        marker="x",
        s=169,
        linewidths=3,
        color="w",
        zorder=10,
    )
    plt.show()

def plot_dendrogram(model, **kwargs):
    counts = np.zeros(model.children_.shape[0])
    n_samples = len(model.labels_)
    for i, merge in enumerate(model.children_):
        current_count = 0
        for child_idx in merge:
            if child_idx < n_samples:
                current_count += 1
            else:
                current_count += counts[child_idx - n_samples]
        counts[i] = current_count

    linkage_matrix = np.column_stack(
        [model.children_, model.distances_, counts]
    ).astype(float)
    dendrogram(linkage_matrix, **kwargs)

data = pd.read_csv('iris.data',header=None)
names=['sepal length in cm', 'sepal width in cm', 'petal length in cm','petal
width in cm']
no_labeled_data=data.drop(4,axis=1)
print(no_labeled_data)

k_means = cluster.KMeans(init='k-means++', n_clusters=3, n_init=15)
k_means.fit(no_labeled_data)
k_means_cluster_centers = k_means.cluster_centers_
print(k_means_cluster_centers)
k_means_labels                                                          =
metrics.pairwise.pairwise_distances_argmin(no_labeled_data,k_means_cluster_ce
nters)
print(k_means_labels)
plotClusters(no_labeled_data,names,k_means_labels,k_means_cluster_centers)

pca = decomposition.PCA(n_components = 2)
pca_data = pca.fit_transform(no_labeled_data)
k_means = cluster.KMeans(init='k-means++', n_clusters=3, n_init=15)
k_means.fit(pca_data)
k_means_labels                                                          =
metrics.pairwise.pairwise_distances_argmin(pca_data,k_means.cluster_centers_)
plotPCAClusters(pca_data,k_means.cluster_centers_)

k_means = cluster.KMeans(init='random', n_clusters=3)
k_means.fit(no_labeled_data)
k_means_cluster_centers = k_means.cluster_centers_
k_means_labels                                                          =
metrics.pairwise.pairwise_distances_argmin(no_labeled_data,k_means_cluster_ce
nters)
plotClusters(no_labeled_data,names,k_means_labels,k_means_cluster_centers)

init_centr = np.array([
    [5,3.5,1.5,0.25],
    [6,2.5,4.5,1.5],
```

```
    [7,3,6,2]
])
k_means = cluster.KMeans(init=init_centr, n_clusters=3)
k_means.fit(no_labeled_data)
k_means_cluster_centers = k_means.cluster_centers_
k_means_labels                                                          =
metrics.pairwise.pairwise_distances_argmin(no_labeled_data,k_means_cluster_ce
nters)
plotClusters(no_labeled_data,names,k_means_labels,k_means_cluster_centers)

X= no_labeled_data.iloc[:,2:].values
wcss=[]
for i in range(1,15):
    kmean = cluster.KMeans(n_clusters=i,init="k-means++")
    kmean.fit_predict(X)
    wcss.append(kmean.inertia_)
plt.plot(range(1,15),wcss)
plt.xlabel("Количество кластеров")
plt.ylabel("Дисперсия")
plt.show()

mb_k_means=cluster.MiniBatchKMeans(n_clusters=3)
mb_k_means.fit(no_labeled_data)
mb_k_means_cluster_centers = mb_k_means.cluster_centers_
mb_k_means_cluster_centers=sorted(mb_k_means_cluster_centers.tolist(),key=lam
bda x:x[0])
mb_k_means_labels                                                       =
metrics.pairwise.pairwise_distances_argmin(no_labeled_data,mb_k_means_cluster
_centers)
plotClusters(no_labeled_data,names,mb_k_means_labels,mb_k_means_cluster_cente
rs)

k_means = cluster.KMeans(n_clusters=3)
k_means.fit(no_labeled_data)
k_means_cluster_centers = k_means.cluster_centers_
k_means_cluster_centers=sorted(k_means_cluster_centers.tolist(),key=lambda
x:x[0])
k_means_labels                                                          =
metrics.pairwise.pairwise_distances_argmin(no_labeled_data,k_means_cluster_ce
nters)
f, ax = plt.subplots(1, 3)
for j in range(3):
       my_members   =   [k_means_labels[i]==mb_k_means_labels[i]   for   i   in
range(len(k_means_labels))]
     ax[j].plot(no_labeled_data[my_members][j],no_labeled_data[my_members][j+1
], 'w',markerfacecolor="#5900FFFF", marker='o', markersize=4)
      my_members  =  [not  (k_means_labels[i]==mb_k_means_labels[i])  for  i  in
range(len(k_means_labels))]
     ax[j].plot(no_labeled_data[my_members][j],no_labeled_data[my_members][j+1
], 'w',markerfacecolor="#FF0000FF", marker='o', markersize=4)
    ax[j].set_xlabel(names[j])
    ax[j].set_ylabel(names[j+1])
plt.show()

hier = cluster.AgglomerativeClustering(n_clusters=3, linkage='average')
hier = hier.fit(no_labeled_data)
hier_labels = hier.labels_
plotHierClusters(no_labeled_data,hier_labels)

hier = cluster.AgglomerativeClustering(n_clusters=2, linkage='average')
hier = hier.fit(no_labeled_data)
hier_labels = hier.labels_
plotHierClusters(no_labeled_data,hier_labels)
```

```
hier = cluster.AgglomerativeClustering(n_clusters=5, linkage='average')
hier = hier.fit(no_labeled_data)
hier_labels = hier.labels_
plotHierClusters(no_labeled_data,hier_labels)

hier                                                                    =
cluster.AgglomerativeClustering(linkage='average',compute_distances=True)
hier = hier.fit(no_labeled_data)
plot_dendrogram(hier, 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 = cluster.AgglomerativeClustering(n_clusters=2, linkage='ward')
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()

hier = cluster.AgglomerativeClustering(n_clusters=2, linkage='average')
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()

hier = cluster.AgglomerativeClustering(n_clusters=2, linkage='complete')
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()

hier = cluster.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()
```