



Специальные технологии баз данных и информационных систем

НИЯУ МИФИ, КАФЕДРА ФИНАНСОВОГО МОНИТОРИНГА

КУРС ЛЕКЦИЙ

В.Ю. РАДЫГИН. ЛЕКЦИЯ 3. СЕМЕСТР 2

Библиотеки

В данной лекции будут рассмотрены примеры с использованием следующих библиотек:

- NumPy – <https://numpy.org/>
- Pandas – <https://pandas.pydata.org/>
- scikit-learn – <https://scikit-learn.org>
- Matplotlib – <https://matplotlib.org/>

Часть 1

МЕТОДЫ КЛАСТЕРИЗАЦИИ ДАННЫХ

Кластеризация

Задача кластеризации – это задача разделения множества наблюдений на непересекающиеся подмножества (кластеры). Причём внутри одного кластера наблюдения должны обладать схожими признаками, а наблюдения разных кластеров должны быть существенно отличимы. В задачах кластеризации число кластеров может быть неизвестно заранее и даже может не быть конечным. Сам факт выявления числа кластеров – это уже отдельная востребованная задача.

Постановка задачи кластеризации

Пусть множество всех возможных наблюдений X . Множество идентификаторов кластеров Y . На множестве X задана функция расстояния между объектами (наблюдениями) $\rho(x_1, x_2)$, где $x_1, x_2 \in X$. Дана выборка $X^m = \{x_1, x_2, \dots, x_m\} \subset X$.

Задача:

Построить множество $Z^m = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$, где $\{x_1, x_2, \dots, x_m\} = X^m$, а $y_1, y_2, \dots, y_m \in Y$. Причём для некоторых i и $j \in 1, 2, \dots, m$ возможно, что $y_i = y_j$. Причём для всех x_i, x_j , таких, что $y_i = y_j$ должно выполняться условие их близости относительно метрики ρ , а для объектов из разных кластеров данная метрика должна быть различима.

Множество идентификаторов кластеров Y может быть не задано, тогда его нахождение также является частью задачи кластеризации.

Цели кластеризации

Понимание данных. Разделение наблюдений на схожие по характеристикам группы может дать понимание каких-либо закономерностей явлений, приведших к образованию данного множества информации.

Сжатие данных. Выбор в каждом кластере только одного (или небольшого по сравнению с общим размером кластера числа наблюдений) может позволить перейти к более компактному хранению информации.

Обнаружение новизны — выявление нетипичных объектов (не попавших ни в один из кластеров) может дать новые знания об исследуемом явлении.

Примеры задач кластеризации

Примером задач кластеризации может быть задача выявления по результатам маркетинговых исследований характерных группы потребителей, дифференцированных по сфере их интересов.

Другим примером кластеризации могут служить задачи разделения множества новостей на группы по содержанию.

Распознавание произвольных графических образов также может быть примером задачи кластеризации.

Задача 1

Решим классическую задачу кластеризации. Кластеризацию ирисов Фишера [1]. Ирисы Фишера – это набор данных, собранных американским ботаником Эдгаром Андерсоном. Каждая запись данного набора состоит из длины наружной доли околоцветника или чашелистника (англ. sepal length), ширины наружной доли околоцветника или чашелистника (англ. sepal width), длины внутренней доли околоцветника или лепестка (англ. petal length), ширина внутренней доли околоцветника или лепестка (англ. petal width) и указания вида ириса (класса). Всего рассмотрено три вида ирисов: setosa, versicolor, и virginica. Первый из них линейно отделим от других.

На данной задаче часто проверяют качество методов кластеризации, сравнивая полученные результаты с реальным делением на классы (по видам ирисов).

Ирисы Фишера на Википедии

Ирисы Фишера

Длина чашелистика ↕	Ширина чашелистика ↕	Длина лепестка ↕	Ширина лепестка ↕	Вид ириса ↕
5.1	3.5	1.4	0.2	<i>setosa</i>
4.9	3.0	1.4	0.2	<i>setosa</i>
4.7	3.2	1.3	0.2	<i>setosa</i>
4.6	3.1	1.5	0.2	<i>setosa</i>
5.0	3.6	1.4	0.2	<i>setosa</i>
5.4	3.9	1.7	0.4	<i>setosa</i>
4.6	3.4	1.4	0.3	<i>setosa</i>
5.0	3.4	1.5	0.2	<i>setosa</i>
4.4	2.9	1.4	0.2	<i>setosa</i>
4.9	3.1	1.5	0.1	<i>setosa</i>
5.4	3.7	1.5	0.2	<i>setosa</i>
4.8	3.4	1.6	0.2	<i>setosa</i>
4.8	3.0	1.4	0.1	<i>setosa</i>



Iris setosa



Iris virginica



Импорт данных

Скопируем данную таблицу в текстовый файл (для однозначности назовём его `irises.txt`). Затем импортируем его и визуально изучим.

Импорт и подготовка данных

```
example2-3runner.py - E:\Works\Victor\Students\STDB\Term2\Lecture3\example2-3runner.py (3.7.2)
File Edit Format Run Options Window Help
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
plt.get_current_fig_manager().window.wm_geometry('1400x750+50+50')

pd.set_option('display.max_columns', 2000)
pd.set_option('display.width', 2000)

table0 = pd.read_excel("../Lecture6/irises.xlsx")
table = table0.copy()

from sklearn import preprocessing
scaler_std = preprocessing.StandardScaler()
x = scaler_std.fit_transform(table[['sepal_length', 'sepal_width',
                                     'petal_length', 'petal_width']])
table[['sepal_length', 'sepal_width',
        'petal_length', 'petal_width']] = x

print(table)
```

Ln: 15 Col: 48

Импорт и подготовка данных (текстом)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.get_current_fig_manager().window.wm_geometry('1400x750+50+50')
pd.set_option('display.max_columns', 2000)
pd.set_option('display.width', 2000)
table0 = pd.read_excel("../Lec6/irises.xlsx")
table = table0.copy()
from sklearn import preprocessing
scaler_std = preprocessing.StandardScaler()
x = scaler_std.fit_transform(table[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']])
table[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']] = x
print(table)
```

Результат

```
*Python 3.7.2 Shell*
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: E:\Works\Victor\Students\STDB\Term2\Lec3\example2-3.py ====
      sepal_length  sepal_width  petal_length  petal_width  class_label
0      -0.900681      1.019004      -1.340227      -1.315444      setosa
1      -1.143017      -0.131979      -1.340227      -1.315444      setosa
2      -1.385353      0.328414      -1.397064      -1.315444      setosa
3      -1.506521      0.098217      -1.283389      -1.315444      setosa
4      -1.021849      1.249201      -1.340227      -1.315444      setosa
5      -0.537178      1.939791      -1.169714      -1.052180      setosa
6      -1.506521      0.788808      -1.340227      -1.183812      setosa
7      -1.021849      0.788808      -1.283389      -1.315444      setosa
8      -1.748856      -0.362176      -1.340227      -1.315444      setosa
Ln: 9 Col: 0
```

Визуализируем ирисы

```
example2-3runner.py - E:\Works\Victor\Students\STDB\Term2\Lecture3\example2-3runner.py (3.7.2)
File Edit Format Run Options Window Help

colors_map = {'virginica': 'red',
              'setosa': 'green', 'versicolor': 'blue'}
table['color'] = table['class_label']
table['color'] = table['color'].apply(
    lambda x: colors_map[x])

titles = ['sepal length', 'sepal width',
          'petal length', 'petal width']

for i in range(0, 4):
    y = table0.iloc[:, i]
    for j in range(0, 4):
        x = table0.iloc[:, j]
        ax = plt.subplot(4, 4, i * 4 + j + 1)
        if i == 0:
            ax.set_title(titles[j])
        if j == 0:
            ax.set_ylabel(titles[i])
        if i == j:
            plt.hist(y, rwidth = 0.5)
        else:
            plt.scatter(x, y, c = table['color'], s = 2)
plt.show()
```

Ln: 32 Col: 0

Визуализируем ирисы (текстом)

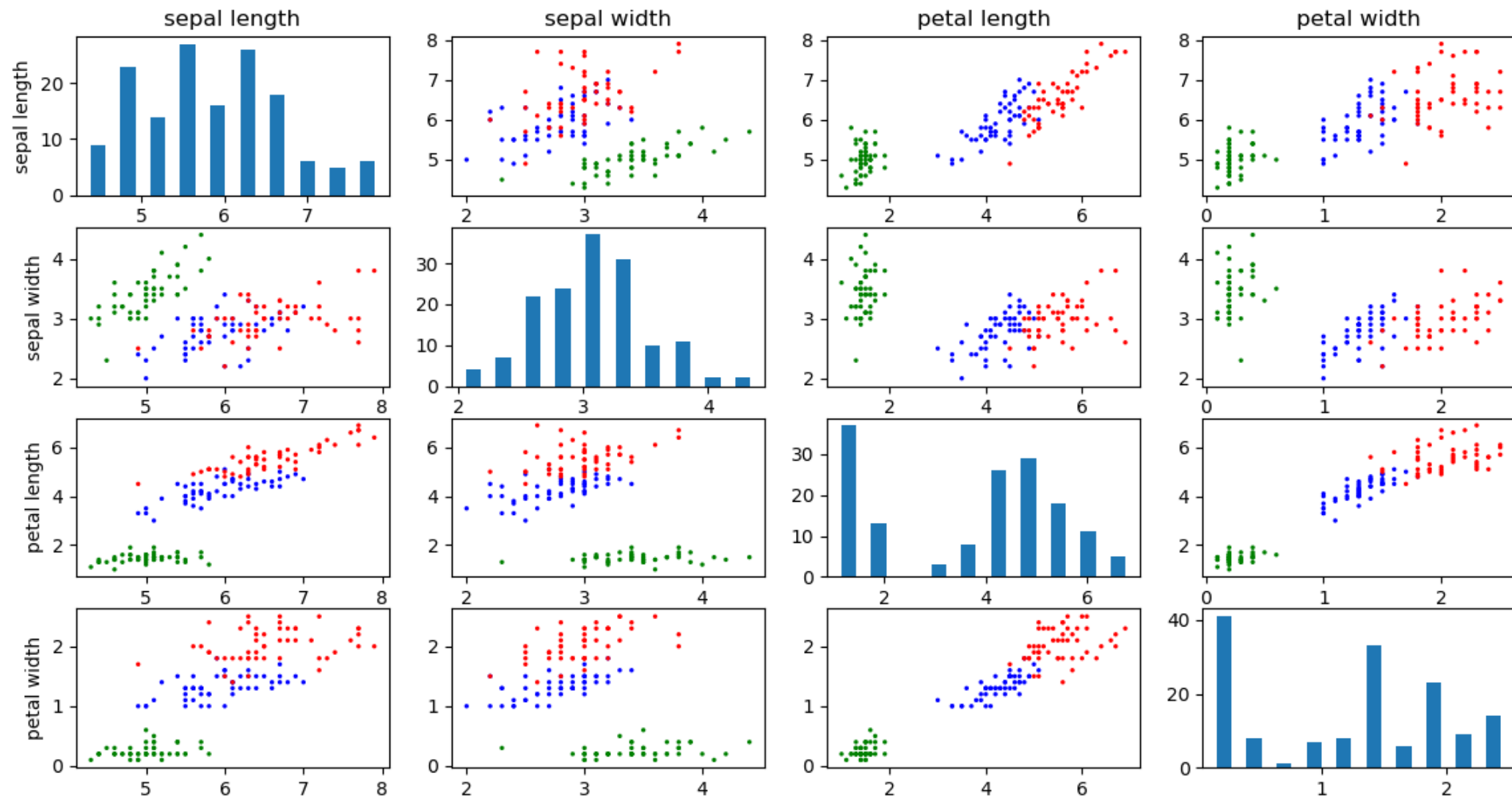
```
colors_map = {'virginica': 'red',
              'setosa': 'green', 'versicolor': 'blue'}
table['color'] = table['class_label']
table['color'] = table['color'].apply(
    lambda x: colors_map[x])

titles = ['sepal length', 'sepal width',
          'petal length', 'petal width']

for i in range(0, 4):
    y = table0.iloc[:, i]
    for j in range(0, 4):
```

```
        x = table0.iloc[:, j]
        ax = plt.subplot(4, 4, i * 4 + j + 1)
        if i == 0:
            ax.set_title(titles[j])
        if j == 0:
            ax.set_ylabel(titles[i])
        if i == j:
            plt.hist(y, rwidth = 0.5)
        else:
            plt.scatter(x, y, c = table['color'], s = 2)
plt.show()
```

Figure 1



Метод к-средних

Идея метода очень проста. В её основе лежит понятие центроида – точки, расстояние ρ от которой до всех объектов кластера минимально.

1. Выбираем наугад центроиды кластеров.
2. Распределяем все наблюдения по кластерам на основе принципа наикратчайшего расстояния до центроида.
3. Внутри каждого кластера вычисляем новые центроиды по принципу нахождения центра тяжести.
4. Если новые центроиды отличаются от центроидов предыдущего шагов, то возвращаемся к шагу 2.
5. Если центроиды не изменились, то текущее распределение по кластерам и есть результат кластеризации.

Некоторые замечания

В обычном методе к-средних в качестве функции ρ используют Евклидово расстояние ($\rho(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \dots + (x_m - y_m)^2}$), но оно может быть заменено на любые другие метрики, например, Расстояние Чебышева или расстояние Левенштейна (для строк и предложений текста).

Для нахождения нового центроида обычно используют принцип центра тяжести:

$$x'_i = \frac{1}{|X^t|} \sum_{x_{i,j} \in X^t} x_{i,j},$$

где x'_i – i -я координата центроида, $|X^t|$ – мощность кластера X^t , $x_{i,j}$ – i -я координата j -го элемента кластера. Но данный подход может быть заменён на любой другой эквивалентный ему.

Подход с остановкой только в случае, если координаты центроида перестали меняться обычно не очень эффективен. Чаще всего его заменяют условием сходимости центроидов.

Подготовка данных

Для эффективного решения задачи кластеризации необходимо убрать доминирование одних переменных над другими за счёт разницы абсолютных значений. Обычно для этого выполняют процедуру стандартизации данных. Данная задача в Python решается при помощи класса `StandardScaler` модуля `preprocessing` библиотеки `Scikit-Learn`.

Стандартизацию мы уже сделали сразу после импорта данных.

Стандартизация данных

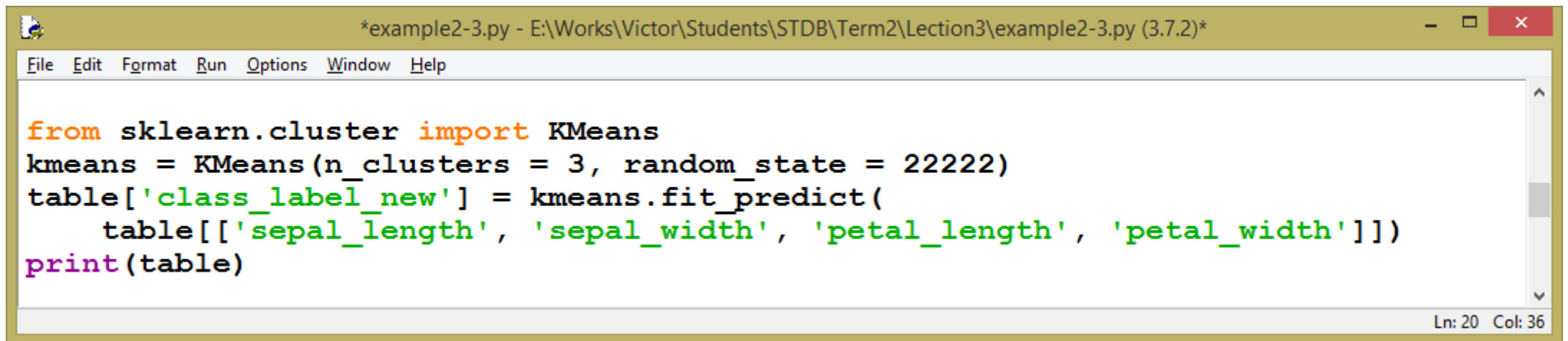
Стандартизация данных – это такое биективное отображение данных из пространства действительных чисел в пространство действительных чисел, при котором данные оказываются распределёнными вокруг 0 со стандартным отклонением 1:

$$x' = \frac{x - M_x}{\sigma_x},$$

где M_x – математическое ожидание (среднее арифметическое) величины x , а σ_x – стандартное отклонение величины x .

Кластерный анализ в Python

Для применения метода k-средних в Python используется класс KMeans модуля cluster библиотеки Scikit-Learn [2].

A screenshot of a Python IDE window titled '*example2-3.py - E:\Works\Victor\Students\STDB\Term2\Lec3\example2-3.py (3.7.2)*'. The window has a menu bar with 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The code editor contains the following Python code:

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters = 3, random_state = 22222)
table['class_label_new'] = kmeans.fit_predict(
    table[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']])
print(table)
```

The status bar at the bottom right shows 'Ln: 20 Col: 36'.

Кластерный анализ в Python (текстом)

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters = 3, random_state = 22222)
table['class_label_new'] = kmeans.fit_predict(
    table[['sepal_length',
            'sepal_width', 'petal_length', 'petal_width']])
print(table)
```

Результат

```
*Python 3.7.2 Shell*
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: E:\Works\Victor\Students\STDB\Term2\Lec3\example2-3.py ====
    sepal_length  sepal_width  petal_length  petal_width  class_label  color  class_label_new
0      -0.900681      1.019004      -1.340227      -1.315444      setosa  green           1
1      -1.143017     -0.131979      -1.340227      -1.315444      setosa  green           1
2      -1.385353      0.328414      -1.397064      -1.315444      setosa  green           1
3      -1.506521      0.098217      -1.283389      -1.315444      setosa  green           1
4      -1.021849      1.249201      -1.340227      -1.315444      setosa  green           1
5      -0.537178      1.939791      -1.169714      -1.052180      setosa  green           1
6      -1.506521      0.788808      -1.340227      -1.183812      setosa  green           1
7      -1.021849      0.788808      -1.283389      -1.315444      setosa  green           1
8      -1.748856     -0.362176      -1.340227      -1.315444      setosa  green           1
9      -1.143017      0.098217      -1.283389      -1.447076      setosa  green           1
```

Визуализируем кластеры

```
example2-3.py - E:\Works\Victor\Students\STDB\Term2\Lecture3\example2-3.py (3.7.2)
File Edit Format Run Options Window Help
colors_map_new = {0: 'red', 1: 'green', 2: 'blue'}
table['color_new'] = table['class_label_new']
table['color_new'] = table['color_new'].apply(lambda x: colors_map_new[x])

for i in range(0, 4):
    y = table.iloc[:, i]
    for j in range(0, 4):
        x = table.iloc[:, j]
        ax = plt.subplot(4, 4, i * 4 + j + 1)
        if i == 0:
            ax.set_title(titles[j])
        if j == 0:
            ax.set_ylabel(titles[i])
        if i == j:
            plt.hist(y, rwidth = 0.5)
        else:
            plt.scatter(x, y, c = table['color_new'], s = 2)
plt.get_current_fig_manager().window.wm_geometry('1400x750+50+50')
plt.show()
```

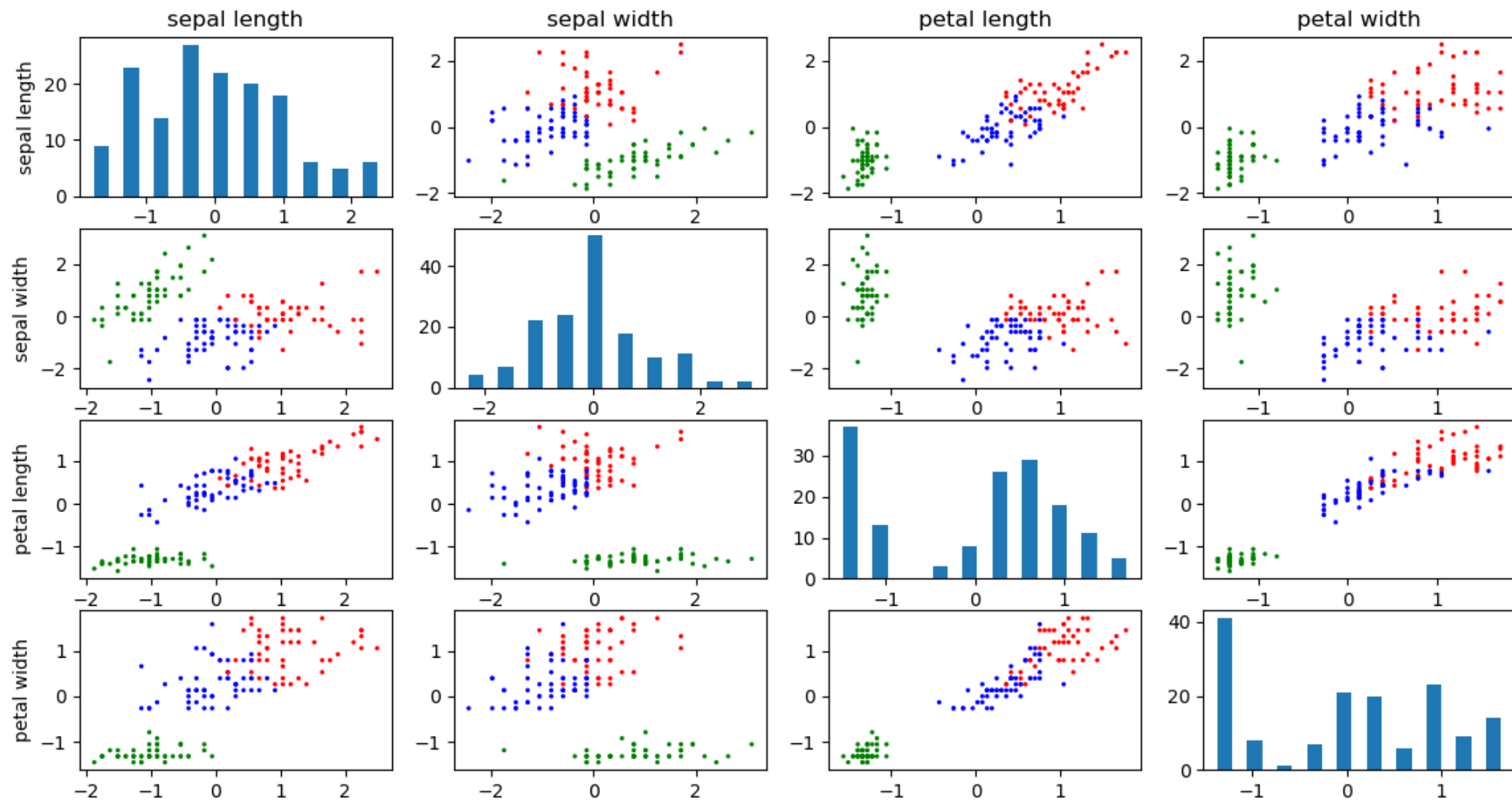
Ln: 79 Col: 66

Визуализируем кластеры (текстом)

```
colors_map_new = {0: 'red', 1: 'green', 2: 'blue'}
table['color_new'] = table['class_label_new']
table['color_new'] = table['color_new'].apply(
    lambda x: colors_map_new[x])
for i in range(0, 4):
    y = table.iloc[:, i]
    for j in range(0, 4):
        x = table.iloc[:, j]
        ax = plt.subplot(4, 4, i * 4 + j + 1)
```

```
if i == 0:
    ax.set_title(titles[j])
if j == 0:
    ax.set_ylabel(titles[i])
if i == j:
    plt.hist(y, rwidth = 0.5)
else:
    plt.scatter(x, y, c = table['color_new'], s = 2)
plt.get_current_fig_manager(
    ).window.wm_geometry('1400x750+50+50')
plt.show()
```

Figure 1

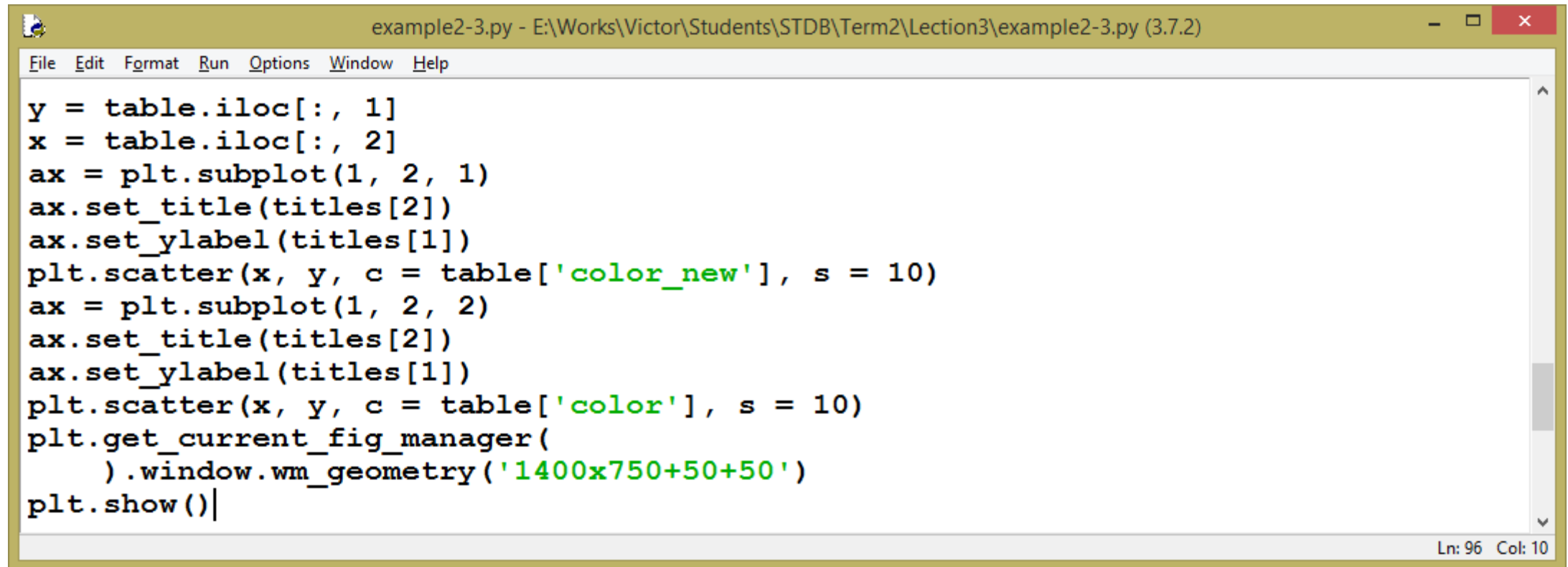


Цвета совпали?

Если сравнить изображения на слайдах 24 и 15, то мы увидим, что цвета большей части точек совпали. Исключение, ряд точек на границах кластеров. Но надо не забывать, что нумерация кластеров случайна и зависит от генератора случайных чисел. При другой инициализации генератора формы кластеров останутся те же, а вот нумерация может быть другой!

Попробуем посмотреть поближе!

Сравнение проекций

A screenshot of a Python IDE window titled 'example2-3.py - E:\Works\Victor\Students\STDB\Term2\Lection3\example2-3.py (3.7.2)'. The window has a menu bar with 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The main text area contains Python code for creating two subplots. The code uses 'plt.subplot' to create a 1x2 grid of plots. The first plot is titled 'titles[2]' and has a y-axis labeled 'titles[1]'. It uses 'plt.scatter' to plot 'x' and 'y' with 'color_new' as the color and 's = 10' as the size. The second plot is also titled 'titles[2]' and has a y-axis labeled 'titles[1]'. It uses 'plt.scatter' to plot 'x' and 'y' with 'color' as the color and 's = 10' as the size. The code also includes 'plt.get_current_fig_manager().window.wm_geometry('1400x750+50+50')' and 'plt.show()'. The status bar at the bottom right shows 'Ln: 96 Col: 10'.

```
example2-3.py - E:\Works\Victor\Students\STDB\Term2\Lection3\example2-3.py (3.7.2)
File Edit Format Run Options Window Help

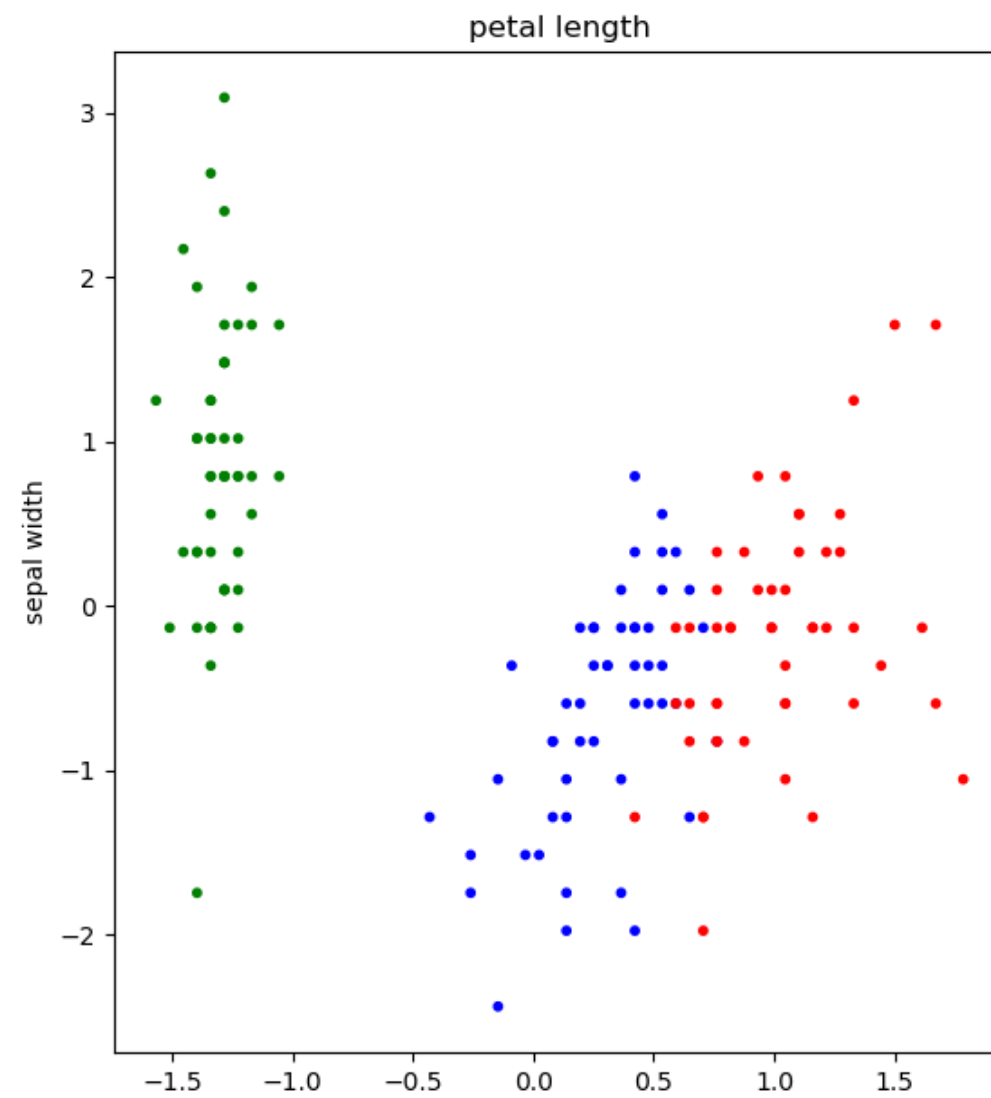
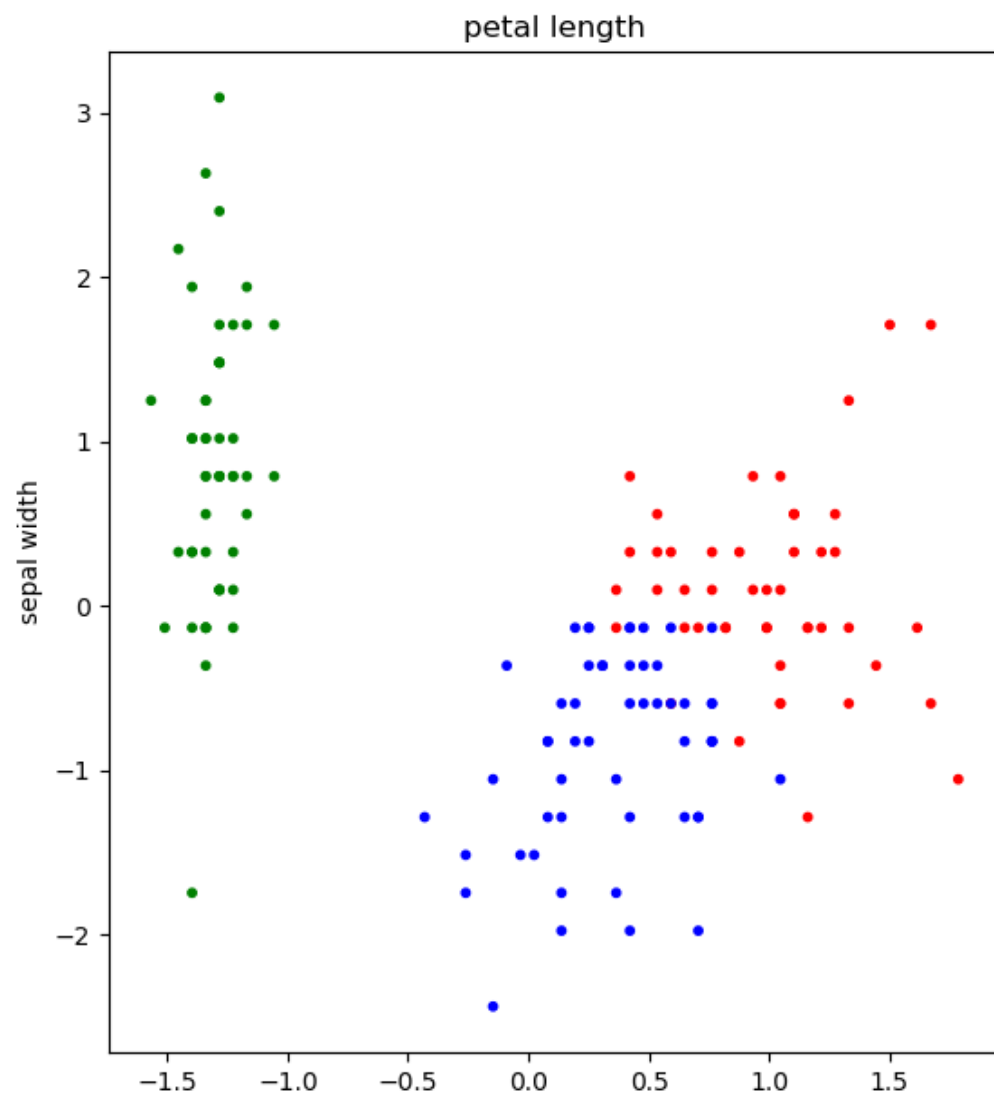
y = table.iloc[:, 1]
x = table.iloc[:, 2]
ax = plt.subplot(1, 2, 1)
ax.set_title(titles[2])
ax.set_ylabel(titles[1])
plt.scatter(x, y, c = table['color_new'], s = 10)
ax = plt.subplot(1, 2, 2)
ax.set_title(titles[2])
ax.set_ylabel(titles[1])
plt.scatter(x, y, c = table['color'], s = 10)
plt.get_current_fig_manager(
    ).window.wm_geometry('1400x750+50+50')
plt.show()

Ln: 96 Col: 10
```

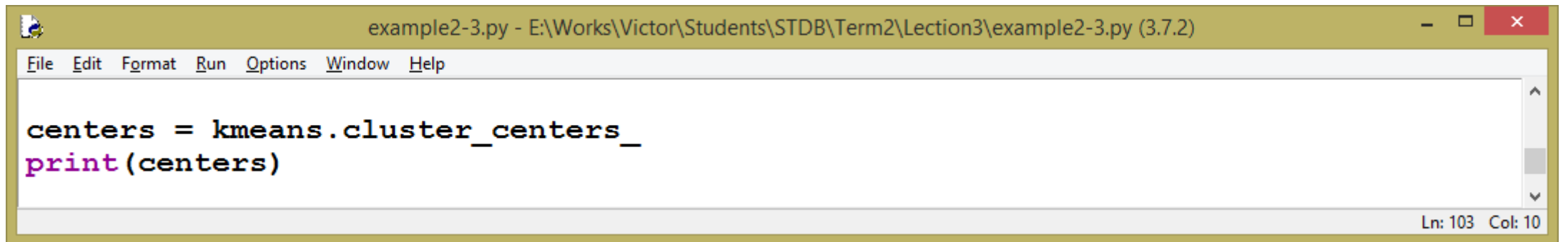
Сравнение проекций (текстом)

```
y = table.iloc[:, 1]
x = table.iloc[:, 2]
ax = plt.subplot(1, 2, 1)
ax.set_title(titles[2])
ax.set_ylabel(titles[1])
plt.scatter(x, y, c = table['color_new'], s = 10)
ax = plt.subplot(1, 2, 2)
ax.set_title(titles[2])
ax.set_ylabel(titles[1])
plt.scatter(x, y, c = table['color'], s = 10)
plt.get_current_fig_manager().window.wm_geometry('1400x750+50+50')
plt.show()
```

Figure 1



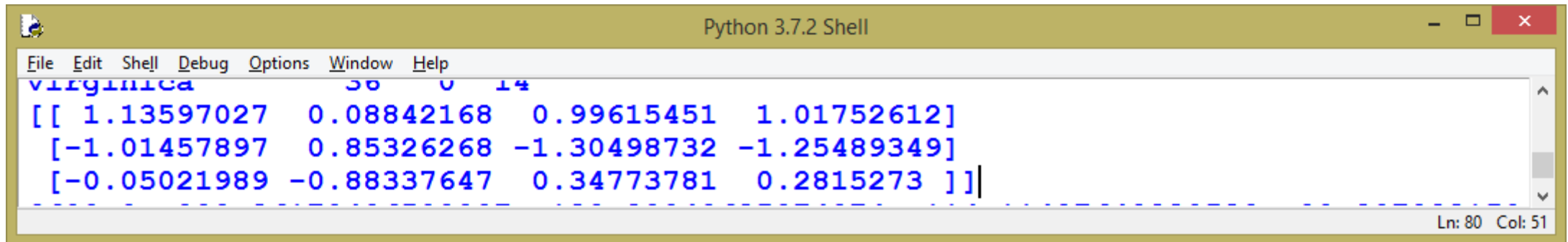
Как узнать координаты центроидов?

A screenshot of a Python IDE window titled "example2-3.py - E:\Works\Victor\Students\STDB\Term2\Lection3\example2-3.py (3.7.2)". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Window", and "Help". The code editor contains two lines of Python code: `centers = kmeans.cluster_centers_` and `print(centers)`. The status bar at the bottom right shows "Ln: 103 Col: 10".

```
example2-3.py - E:\Works\Victor\Students\STDB\Term2\Lection3\example2-3.py (3.7.2)
File Edit Format Run Options Window Help
centers = kmeans.cluster_centers_
print(centers)
Ln: 103 Col: 10
```

```
centers = kmeans.cluster_centers_  
print(centers)
```

Как узнать координаты центроидов?



```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
virginica 50 0 14
[[ 1.13597027  0.08842168  0.99615451  1.01752612]
 [-1.01457897  0.85326268 -1.30498732 -1.25489349]
 [-0.05021989 -0.88337647  0.34773781  0.2815273  ]]|
Ln: 80 Col: 51
```


Отметим центроиды на проекции

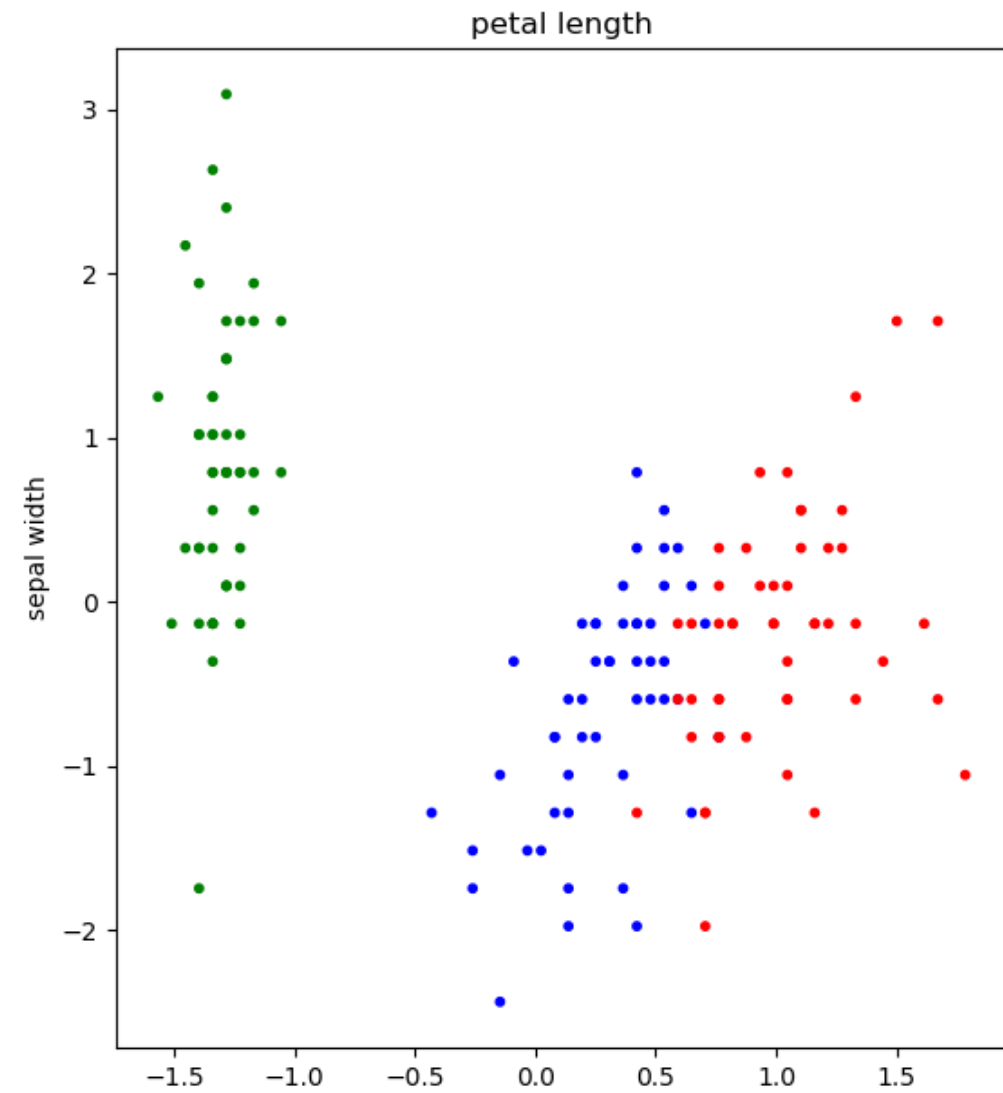
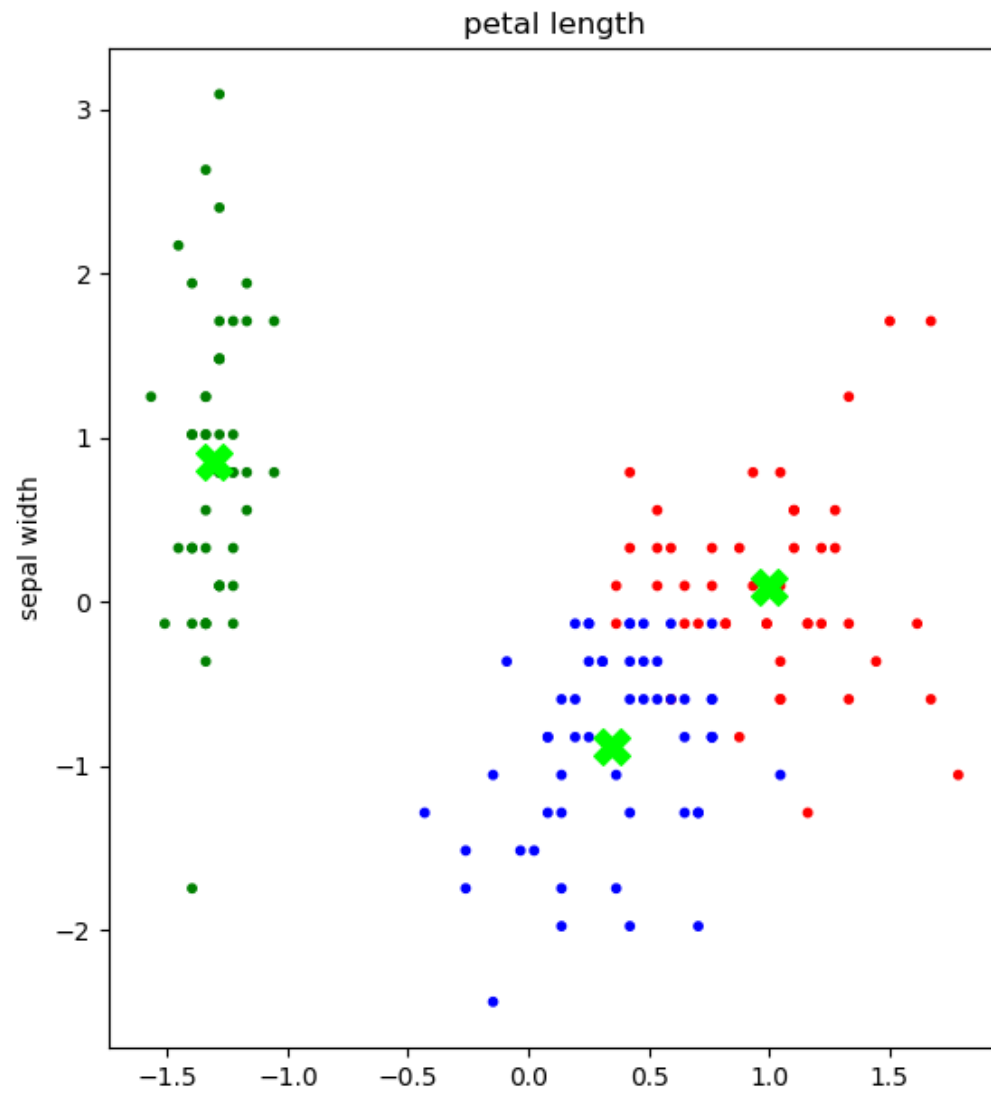
```
*example2-3.py - E:\Works\Victor\Students\STDB\Term2\Lecture3\example2-3.py (3.7.2)*
File Edit Format Run Options Window Help
centers = kmeans.cluster_centers_
y = table.iloc[:, 1]
x = table.iloc[:, 2]
ax = plt.subplot(1, 2, 1)
ax.set_title(titles[2])
ax.set_ylabel(titles[1])
plt.scatter(x, y, c = table['color_new'], s = 10)
plt.scatter(centers[0][2], centers[0][1], c = ['lime'], s = 200, marker = 'X')
plt.scatter(centers[1][2], centers[1][1], c = ['lime'], s = 200, marker = 'X')
plt.scatter(centers[2][2], centers[2][1], c = ['lime'], s = 200, marker = 'X')
ax = plt.subplot(1, 2, 2)
ax.set_title(titles[2])
ax.set_ylabel(titles[1])
plt.scatter(x, y, c = table['color'], s = 10)
|
plt.get_current_fig_manager().window.wm_geometry('1400x750+50+50')
plt.show()
```

Ln: 122 Col: 0

Отметим центроиды на проекции (текстом)

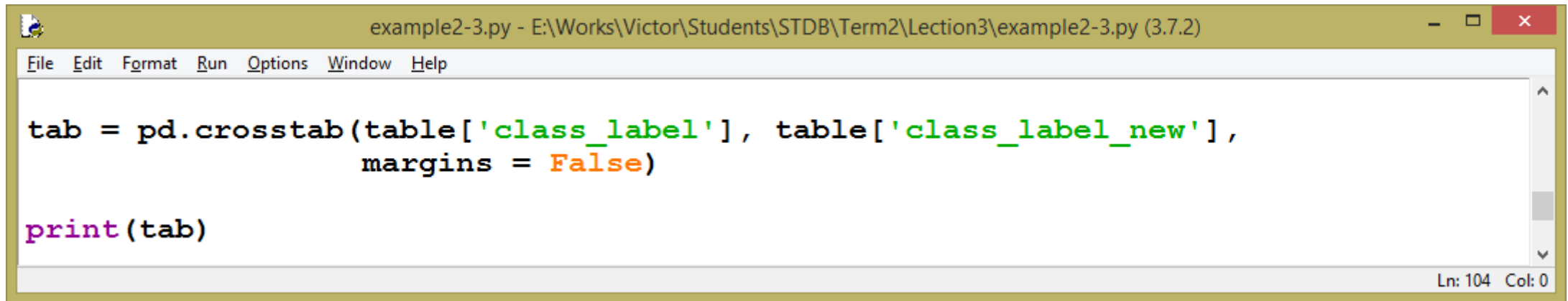
```
centers = kmeans.cluster_centers_  
  
y = table.iloc[:, 1]  
x = table.iloc[:, 2]  
  
ax = plt.subplot(1, 2, 1)  
ax.set_title(titles[2])  
ax.set_ylabel(titles[1])  
  
plt.scatter(x, y, c = table['color_new'], s = 10)  
  
plt.scatter(centers[0][2], centers[0][1], c = ['lime'], s = 200, marker = 'X')  
  
plt.scatter(centers[1][2], centers[1][1], c = ['lime'], s = 200, marker = 'X')  
  
plt.scatter(centers[2][2], centers[2][1], c = ['lime'], s = 200, marker = 'X')  
  
ax = plt.subplot(1, 2, 2)  
ax.set_title(titles[2])  
ax.set_ylabel(titles[1])  
  
plt.scatter(x, y, c = table['color'], s = 10)  
  
plt.get_current_fig_manager().window.wm_geometry('1400x750+50+50')  
  
plt.show()
```

Figure 1



x=0.864644 y=-0.96791

Оценим результат. Кросс-таблица

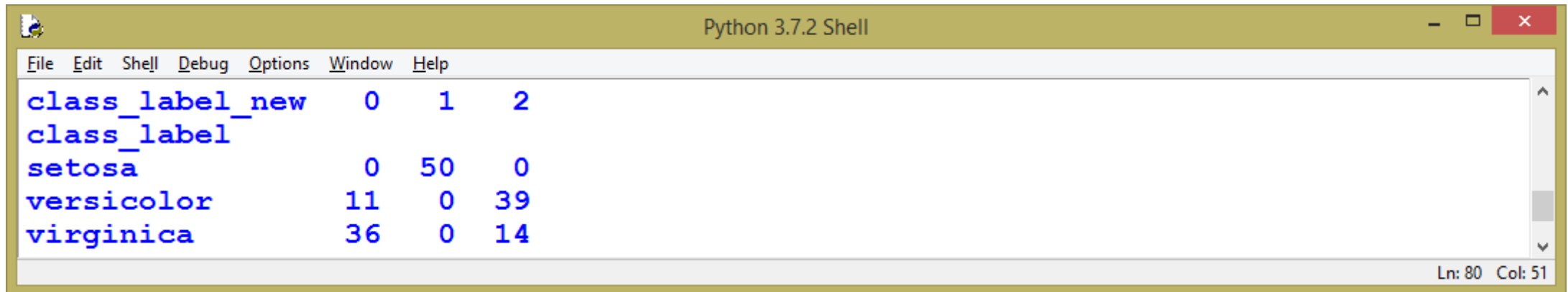
A screenshot of a Python IDE window titled "example2-3.py - E:\Works\Victor\Students\STDB\Term2\Lec3\example2-3.py (3.7.2)". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Window", and "Help". The code editor contains the following Python code:

```
tab = pd.crosstab(table['class_label'], table['class_label_new'],  
                  margins = False)  
  
print(tab)
```

The status bar at the bottom right shows "Ln: 104 Col: 0".

```
tab = pd.crosstab(table['class_label'], table['class_label_new'],  
                  margins = False)  
  
print(tab)
```

Оценим результат. Кросс-таблица



The image shows a screenshot of a Python 3.7.2 Shell window. The window has a title bar that says "Python 3.7.2 Shell" and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with options: File, Edit, Shell, Debug, Options, Window, and Help. The main area of the window displays a cross-tabulation table with the following data:

class_label_new	0	1	2
class_label			
setosa	0	50	0
versicolor	11	0	39
virginica	36	0	14

At the bottom right of the window, the status bar indicates "Ln: 80 Col: 51".

Метрики алгоритма. Инерция и искажение

Искажение – это сумма квадратов расстояний образцов до их ближайшего центра кластера или, говоря другим языком, сумма квадратов минимальных расстояний от образцов до центров кластеров (если перебирать все центры).

$$\text{distortion} = \sum_{i=0}^N \min_{\mu_j \in C} (\|x_i - \mu_j\|^2),$$

где N – число наблюдений, C – множество центров кластеров, x_i – вектор i -го наблюдения, μ_j – вектор j -го центра кластера.

Инерция – это среднее квадратов расстояний от центров кластеров соответствующих кластеров.

$$\text{inertia} = \frac{1}{N} \sum_{i=0}^N \|x_i - \mu_{j_i}\|^2,$$

где j_i – это номер класса, сопоставленного наблюдению x_i .

Как выбрать число кластеров?

В данном примере мы знали число кластеров. Но что делать, если нам оно не известно?

Для определения оптимального числа кластеров вводят следующий критерий: сумма квадратов расстояний от объектов до центроидов кластеров, к которым они относятся должна быть минимальна.

Проблем в том, что минимум данного критерия достигается, когда число кластеров равно числу объектов. Поэтому обычно используют эвристическое понятие «число кластеров, с которого сумма квадратов падает уже не так быстро». Данный критерий можно описать и более формальным выражением, но мы этого делать не будем.

Данный подход принято называть метод «локтя» (elbow method) [3].

Расчёт параметров инерции и искажения

```
example2-3runner.py - E:\Works\Victor\Students\STDB\Term2\Lecture3\example2-3runner.py (3.7.2)
File Edit Format Run Options Window Help
from sklearn import metrics
from scipy.spatial.distance import cdist

rg = range(2, 11)

for k in rg:
    kmeans = KMeans(n_clusters = k, random_state = 22222).fit(
        table[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']])
    inertia.append(kmeans.inertia_)
    distortion.append(sum(np.min(cdist(
        table[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']],
        kmeans.cluster_centers_,
        'euclidean'), axis = 1))/table.shape[0])

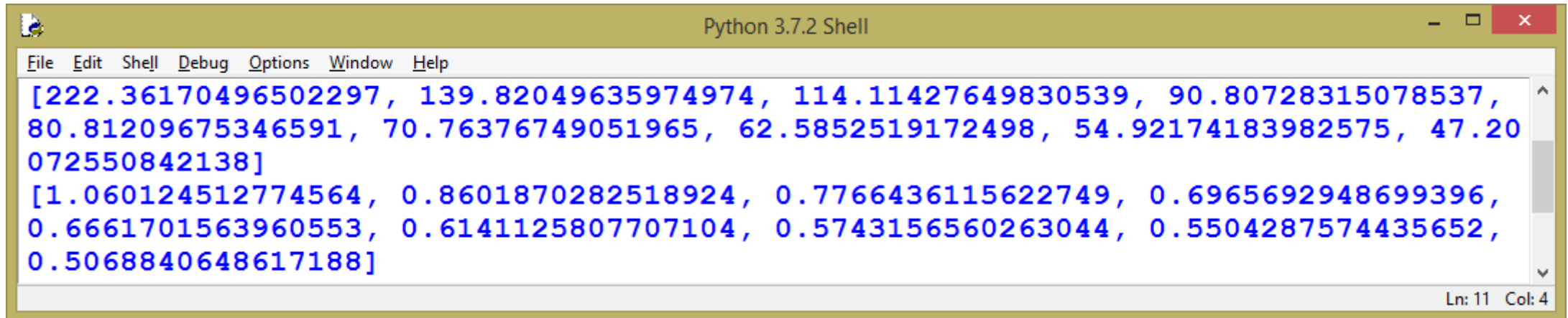
print(inertia)
print(distortion)
```

Ln: 47 Col: 5

Расчёт параметров инерции и искажения (текстом)

```
from sklearn import metrics
from scipy.spatial.distance import cdist
rg = range(2, 11)
for k in rg:
    kmeans = KMeans(n_clusters = k, random_state = 22222).fit(
        table[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']])
    inertia.append(kmeans.inertia_)
    distortion.append(sum(np.min(cdist(
        table[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']],
        kmeans.cluster_centers_,
        'euclidean'),axis = 1))/table.shape[0])
print(inertia)
print(distortion)
```

Результат



A screenshot of a Python 3.7.2 Shell window. The window has a title bar "Python 3.7.2 Shell" and a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main area displays two lines of blue text, each enclosed in square brackets and containing a list of floating-point numbers. The first line contains 8 numbers, and the second line contains 8 numbers. The status bar at the bottom right shows "Ln: 11 Col: 4".

```
[222.36170496502297, 139.82049635974974, 114.11427649830539, 90.80728315078537,  
80.81209675346591, 70.76376749051965, 62.5852519172498, 54.92174183982575, 47.20  
072550842138]  
[1.060124512774564, 0.8601870282518924, 0.7766436115622749, 0.6965692948699396,  
0.6661701563960553, 0.6141125807707104, 0.5743156560263044, 0.5504287574435652,  
0.5068840648617188]
```

Графики

```
example2-3runner.py - E:\Works\Victor\Students\STDB\Term2\Lecture3\example2-3runner.py (3.7.2)
File Edit Format Run Options Window Help

plt.title("Метод локтя для инерции\n",fontsize=16)
plt.scatter(x=[i for i in rg],y=inertia,s=150,edgecolor='k')
plt.plot(rg, inertia);
plt.grid(True)
plt.xlabel("Число кластеров",fontsize=14)
plt.ylabel("Инерция",fontsize=15)
plt.xticks([i for i in rg],fontsize=14)
plt.yticks(fontsize=15)
plt.get_current_fig_manager(
    ).window.wm_geometry('1200x800+50+50')
plt.show()

plt.title("Метод локтя для искажения\n",fontsize=16)
plt.scatter(x=[i for i in rg],y=distortion,s=150,edgecolor='k')
plt.plot(rg, distortion);
plt.grid(True)
plt.xlabel("Число кластеров",fontsize=14)
plt.ylabel("Искажение",fontsize=15)
plt.xticks([i for i in rg],fontsize=14)
plt.yticks(fontsize=15)
plt.get_current_fig_manager(
    ).window.wm_geometry('1200x800+50+50')
plt.show()
```

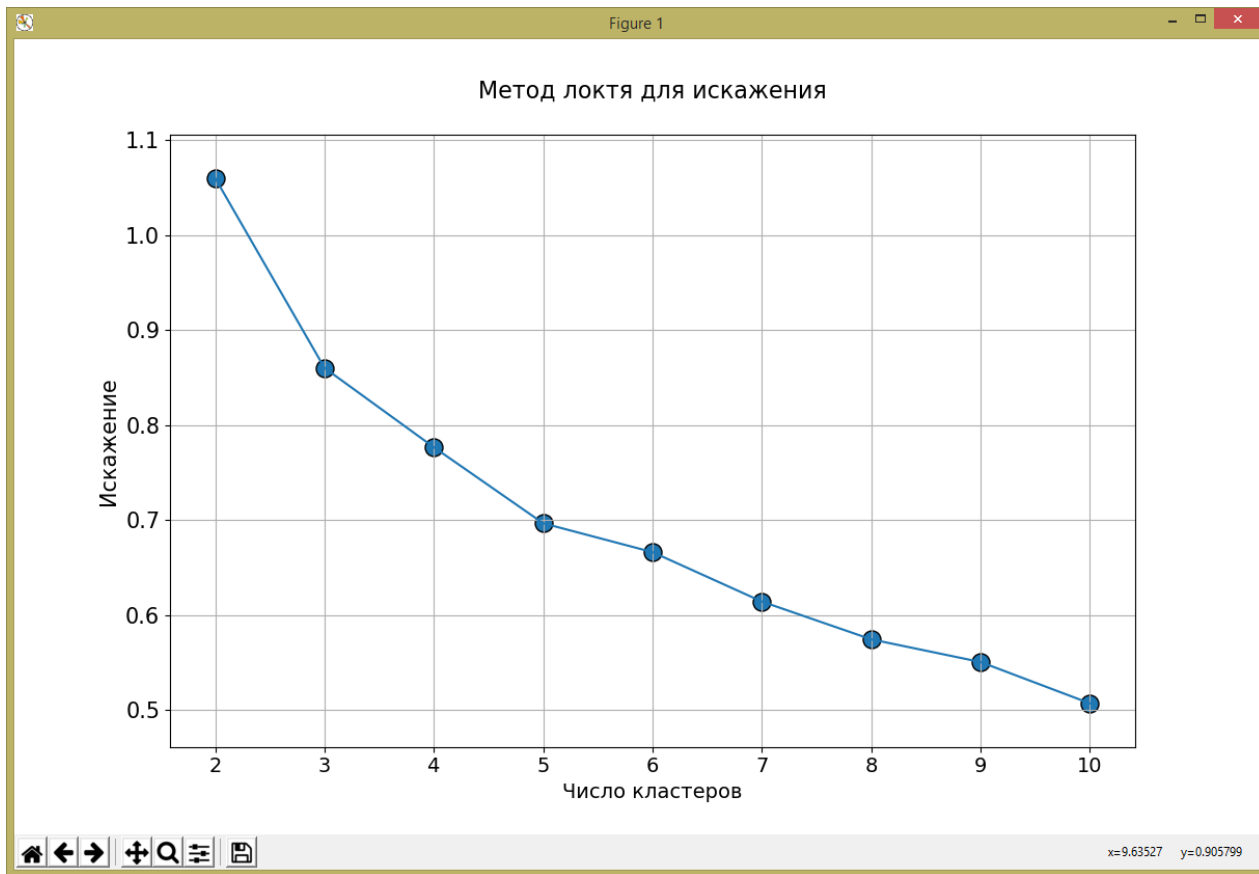
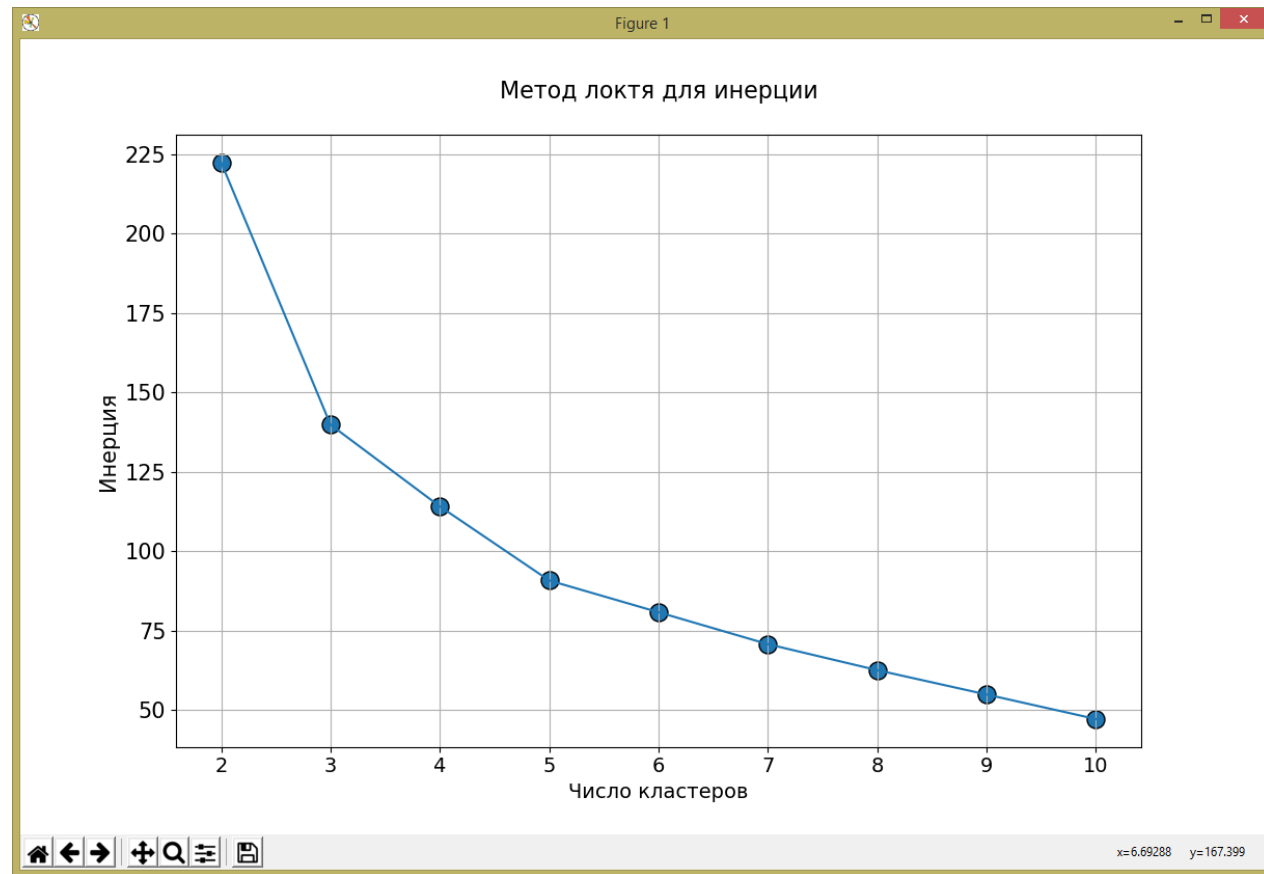
Ln: 38 Col: 35

Графики (текстом)

```
plt.title("Метод локтя для инерции\n",fontsize=16)
plt.scatter(x=[i for i in rg],y=inertia,s=150,edgecolor='k')
plt.plot(rg, inertia);
plt.grid(True)
plt.xlabel("Число кластеров",fontsize=14)
plt.ylabel("Инерция",fontsize=15)
plt.xticks([i for i in rg],fontsize=14)
plt.yticks(fontsize=15)
plt.get_current_fig_manager(
    ).window.wm_geometry('1200x800+50+50')
plt.show()
```

```
plt.title("Метод локтя для искажения\n",fontsize=16)
plt.scatter(x=[i for i in rg],
            y=distortion,s=150,edgecolor='k')
plt.plot(rg, distortion);
plt.grid(True)
plt.xlabel("Число кластеров",fontsize=14)
plt.ylabel("Искажение",fontsize=15)
plt.xticks([i for i in rg],fontsize=14)
plt.yticks(fontsize=15)
plt.get_current_fig_manager(
    ).window.wm_geometry('1200x800+50+50')
plt.show()
```

Результат



Выводы

На обоих графиках видно, что после точки 3 график приближается к прямой. Это означает, что 3 – оптимальное число кластеров. Можно сказать, что метод локтя дал правильный ответ. Причём метод локтя на основе искажения даёт более очевидный ответ, так как на нём точки 4 и 5 больше вписываются в прямую, чем на графике на основе инерции.

Другие метрики. Силуэт

Альтернативой методу локтя может быть использование метрики силуэта [4]. Для расчёта силуэта кластера используют следующую формулу:

$$\text{silhouette} = \frac{b - a}{\max(a, b)},$$

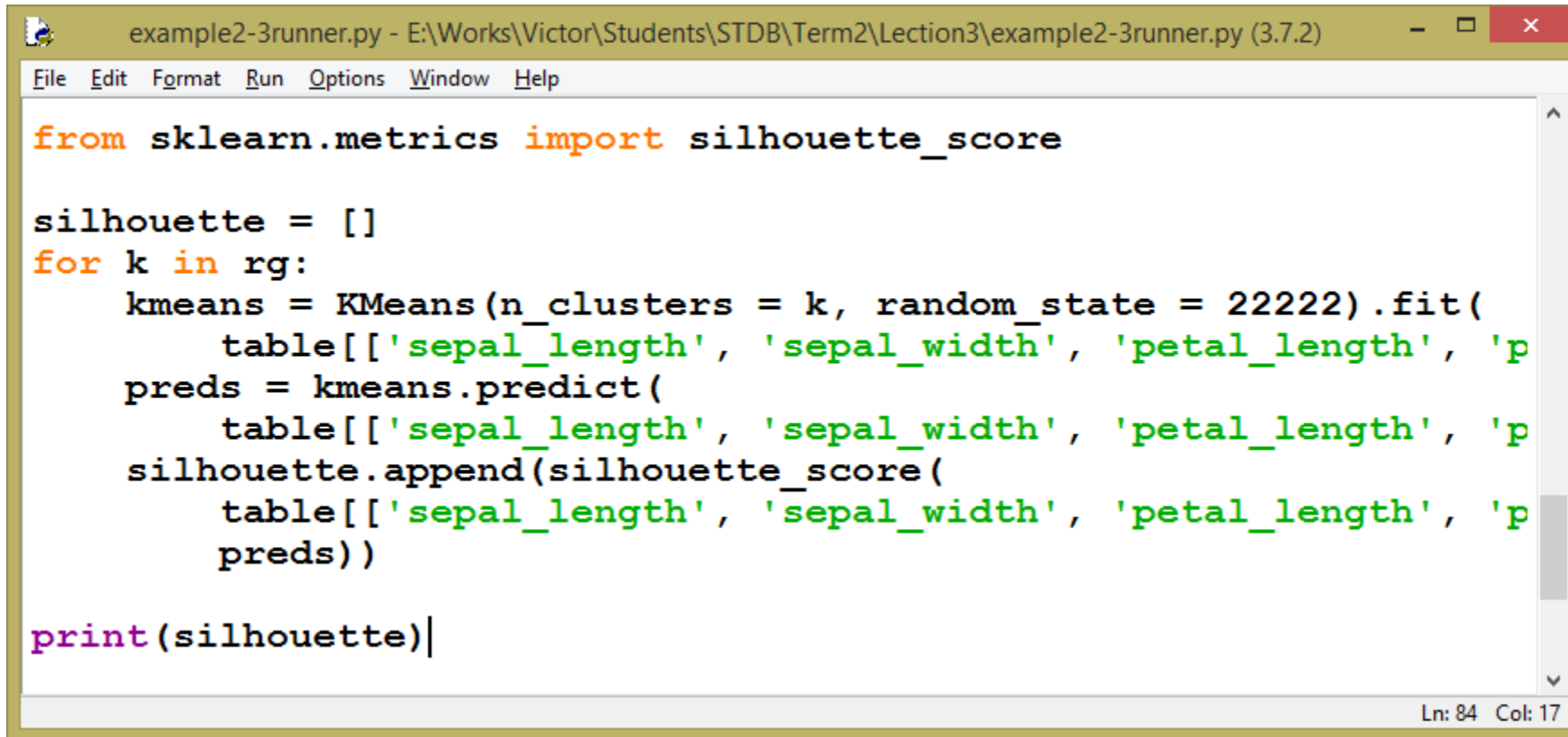
где a – это среднее внутрикластерное расстояние между всеми входящими в кластер парами наблюдений, b – это среднее расстояние между наблюдениями текущего кластера и наблюдениями ближайшего из других кластеров.

Итоговой метрикой является среднее по всем кластерами значение силуэта.

Чем ближе значение силуэта к 1 тем лучше отделены друг от друга кластеры. Значение в районе 0 означает перекрытие кластеров. Отрицательное значение показывает неправильное разделение наблюдений по кластерам.

Тем не менее, метрика по каждому кластеру и по каждому наблюдению также может быть использована для оценки качества кластеризации.

Расчёт силуэта



```
example2-3runner.py - E:\Works\Victor\Students\STDB\Term2\Lecture3\example2-3runner.py (3.7.2)
File Edit Format Run Options Window Help

from sklearn.metrics import silhouette_score

silhouette = []
for k in rg:
    kmeans = KMeans(n_clusters = k, random_state = 22222).fit(
        table[['sepal_length', 'sepal_width', 'petal_length', 'p
    preds = kmeans.predict(
        table[['sepal_length', 'sepal_width', 'petal_length', 'p
    silhouette.append(silhouette_score(
        table[['sepal_length', 'sepal_width', 'petal_length', 'p
        preds))

print(silhouette)|

Ln: 84 Col: 17
```

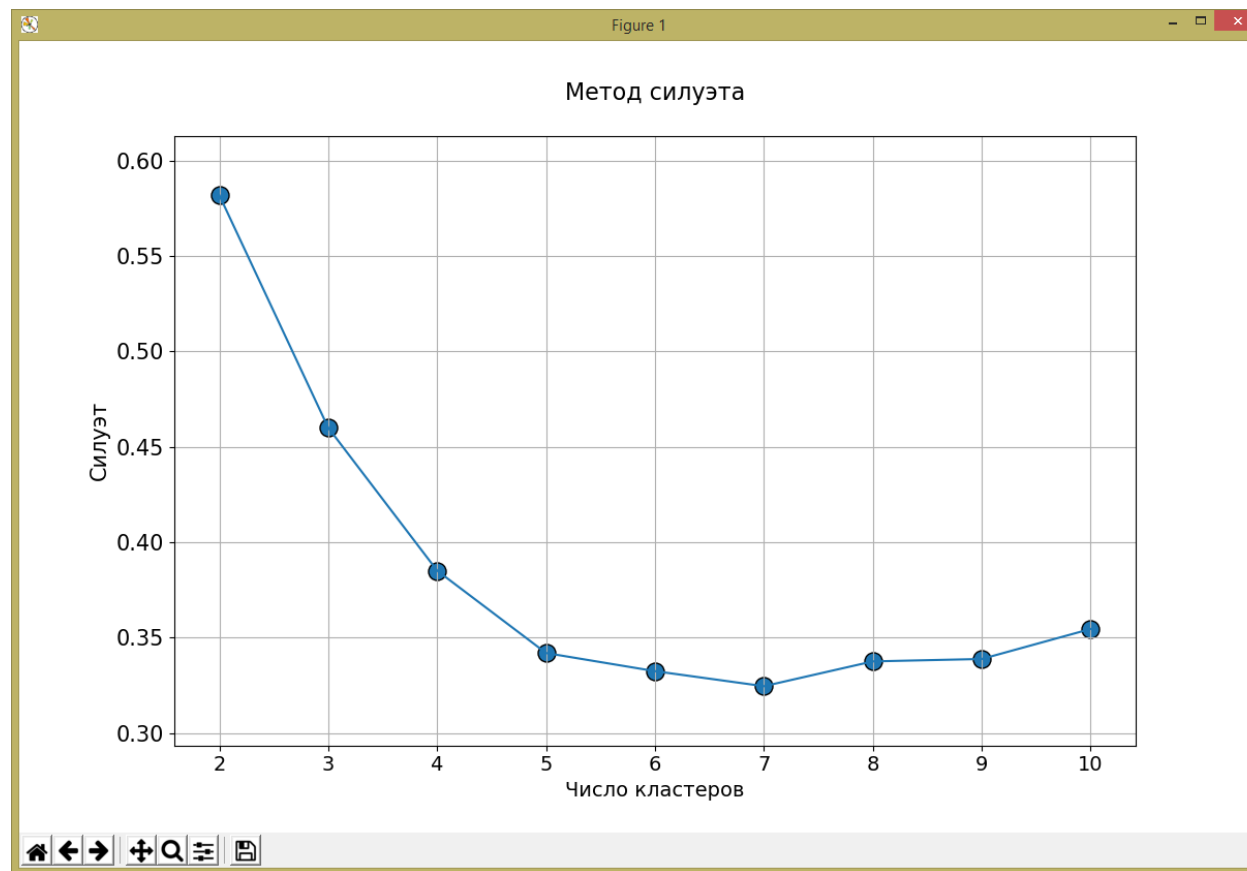

Расчёт силуэта

```
from sklearn.metrics import silhouette_score

silhouette = []
for k in rg:
    kmeans = KMeans(n_clusters = k, random_state = 22222).fit(
        table[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']])
    preds = kmeans.predict(
        table[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']])
    silhouette.append(silhouette_score(
        table[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']],
        preds))

print(silhouette)
```

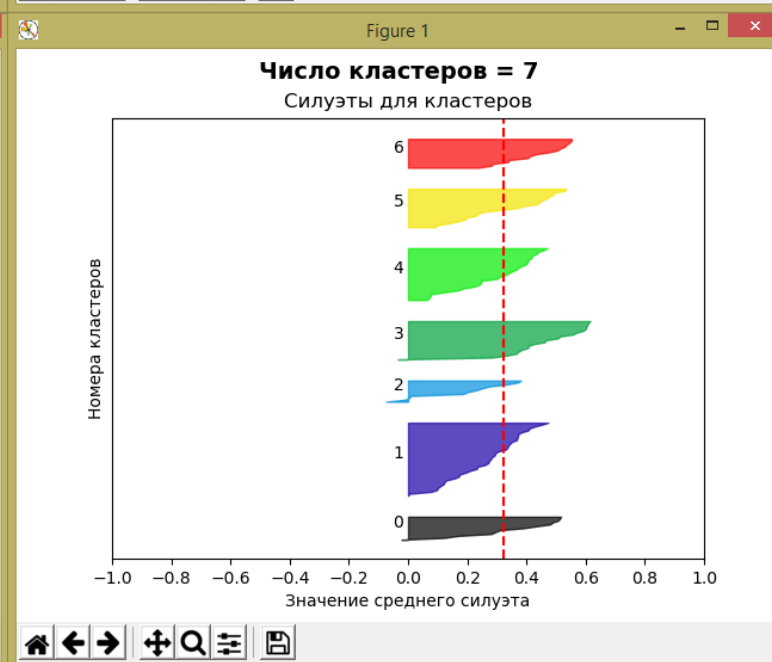
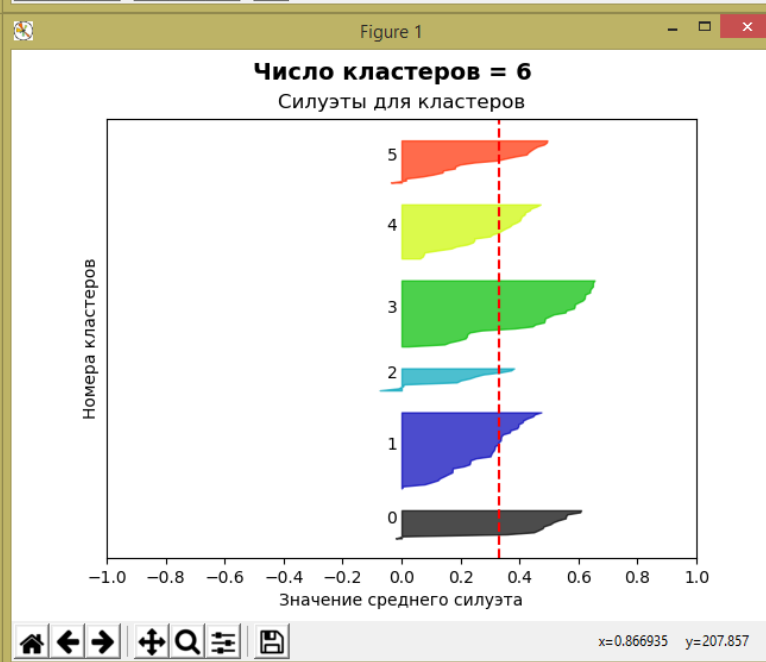
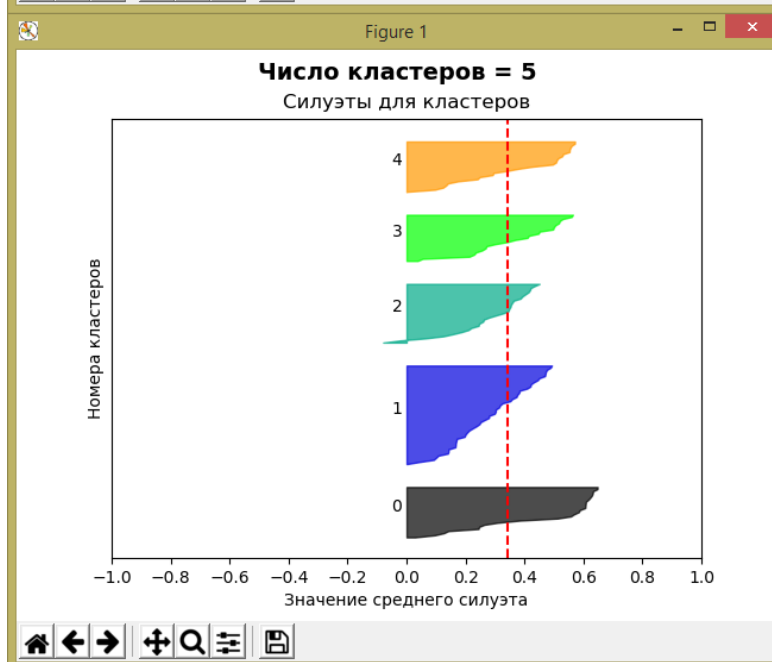
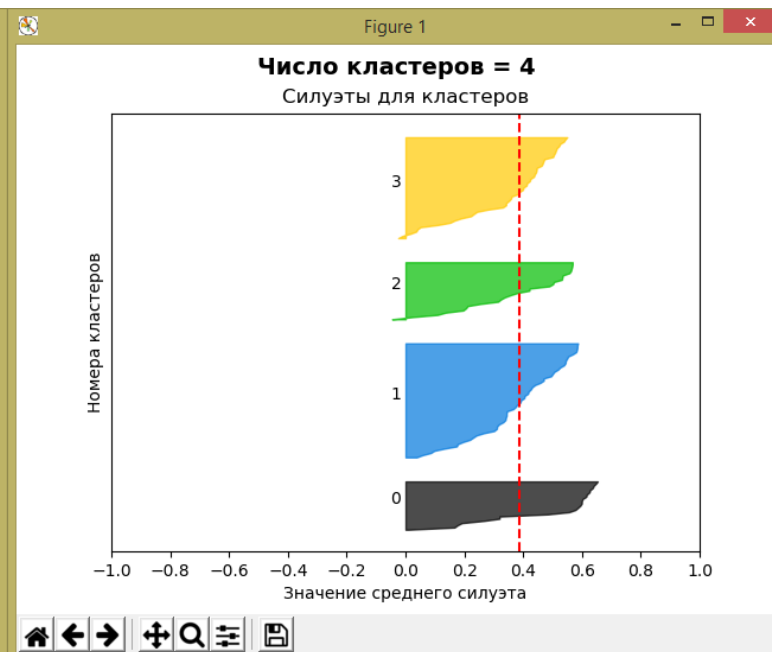
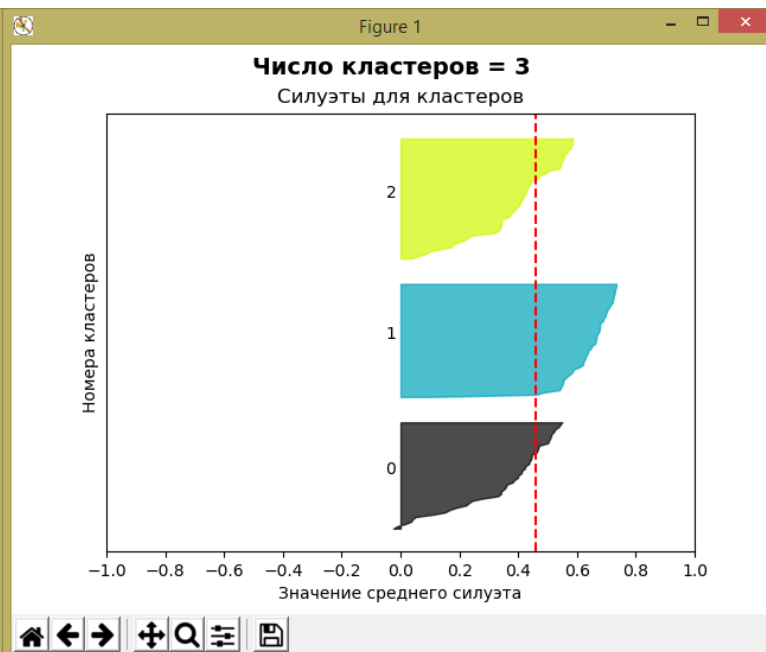
Результат



Выводы

Если судить только по данному графику, то оптимальное число кластеров – 2 (наибольшее значение). Это связано с тем, что в задаче Ирисов два кластера перемешаны и метод силуэта из-за этого считает, что их разделять не надо.

С другой стороны, в задачах с непеременными кластерами метод локтя обычно проигрывает по ясности методу силуэта.



* как построить подобные графики хорошо описано в [4]

Выводы

На данных графиках стоит обратить внимание на следующие факторы:

- Равномерность по горизонтали пятен кластеров. Каждый пик на пятне кластера – это силуэт одной точки внутри него. Чем более равномерно пятно (чем больше оно похоже на плато), тем более одинаково лежат точки кластера.
- Чем более равномерны размеры пятен по вертикали – тем ровнее распределение числа наблюдений по кластерам.
- Чем ближе пятно или его всплески к 1 по горизонтали, тем дальше от границы кластера его точки.
- Чем ближе пятно или его всплески к 0 по горизонтали, тем ближе к границе кластера его точки.
- Если есть часть пятна в отрицательной зоне по горизонтали, то в кластере есть точки, которые, скорее всего, должны быть не в нем.
- Близость всех кластеров к среднему силуэту (красная линия – как раз, значение в одной точке на обычном графике силуэта) – тем равномернее качество отделения кластеров друг от друга.

Другие метрики

Помимо рассмотренных метрик, есть много других метрик [5]. К примеру, довольно интересны метрики **Индекс Дэвиса-Болдина** [6], **гомогенности** [7], **полноты** [8] и **V-мера** [9].

Часть 2

АГЛОМЕРАТИВНЫЕ МЕТОДЫ

Агломеративные методы

Идея агломеративных или иерархических методов кластеризации заключается в соединении маленьких кластеров в большие. Подобные алгоритмы начинают работу с того, что каждому наблюдению сопоставляют свой собственный кластер. Затем два ближайших кластера объединяются в один и так далее, пока не будет образован один общий кластер.

Наиболее известны методы single-link (наименьшее расстояние между двумя любыми представителями кластеров), complete-link (наименьшее расстояние между двумя наиболее удалёнными представителями кластеров), Уорда (Ward), average-link (среднее расстояние между представителями кластеров).

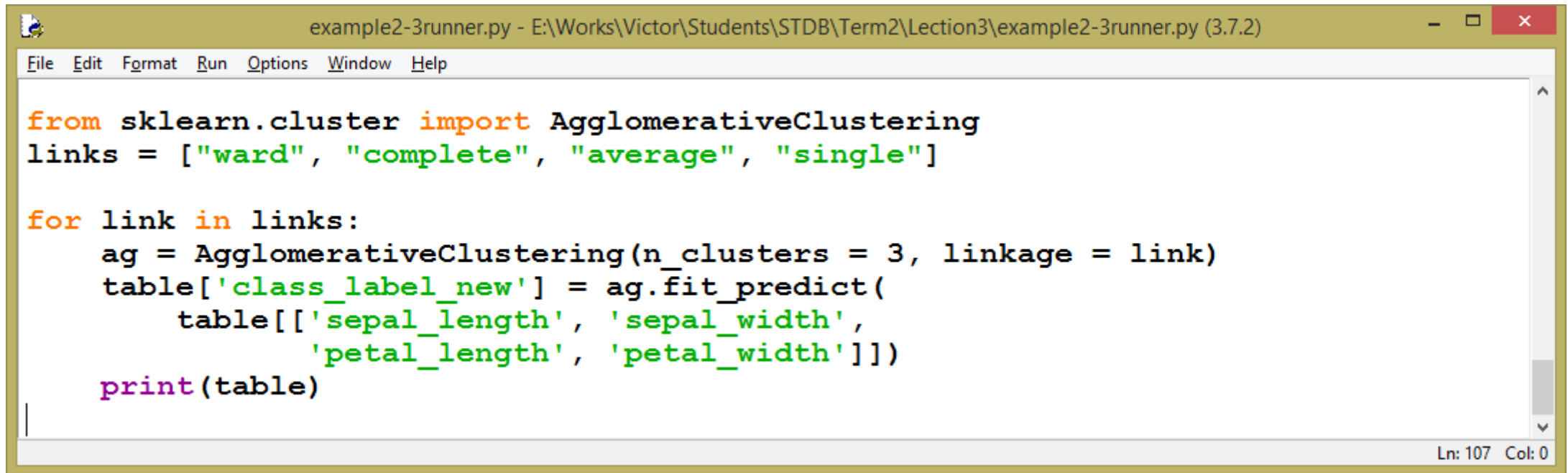
Агломеративный кластерный анализ в Python

Для применения агломеративных методов в Python используется класс **AgglomerativeClustering** модуля cluster библиотеки Scikit-Learn [10].

Для выбора конкретного метода при создании объекта кластера нужно указать параметр `linkage = метод`, где метод может быть:

- "ward" – метод Уорда;
- "complete" – метод полной связи;
- "average" – метод средней связи;
- "single" – метод одиночной связи.

Агломеративный кластерный анализ



```
example2-3runner.py - E:\Works\Victor\Students\STDB\Term2\Lecture3\example2-3runner.py (3.7.2)
File Edit Format Run Options Window Help

from sklearn.cluster import AgglomerativeClustering
links = ["ward", "complete", "average", "single"]

for link in links:
    ag = AgglomerativeClustering(n_clusters = 3, linkage = link)
    table['class_label_new'] = ag.fit_predict(
        table[['sepal_length', 'sepal_width',
                'petal_length', 'petal_width']])
    print(table)
```

Ln: 107 Col: 0

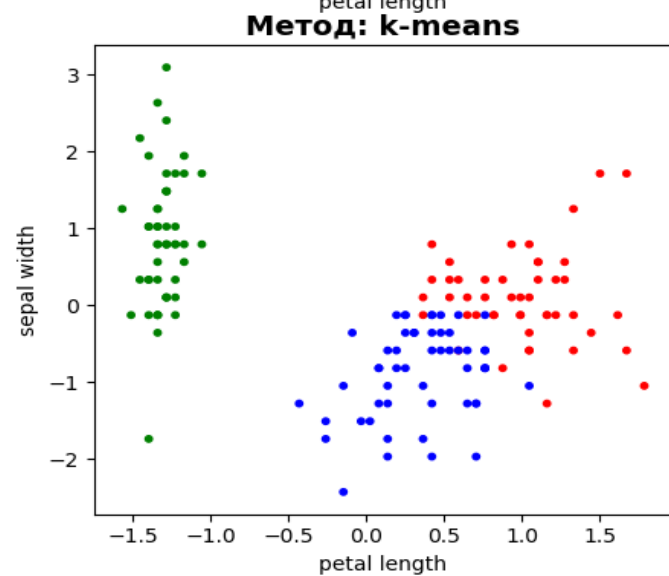
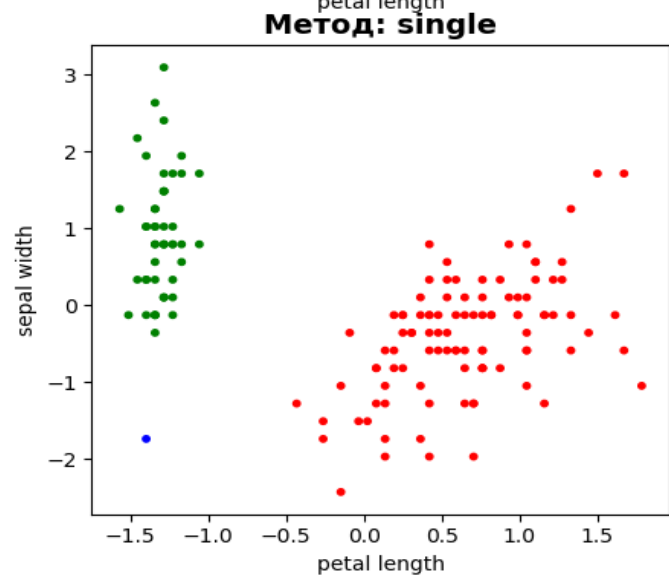
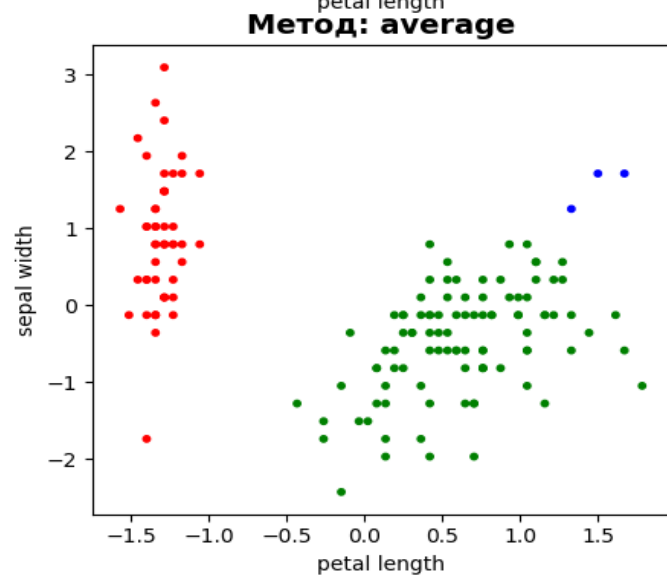
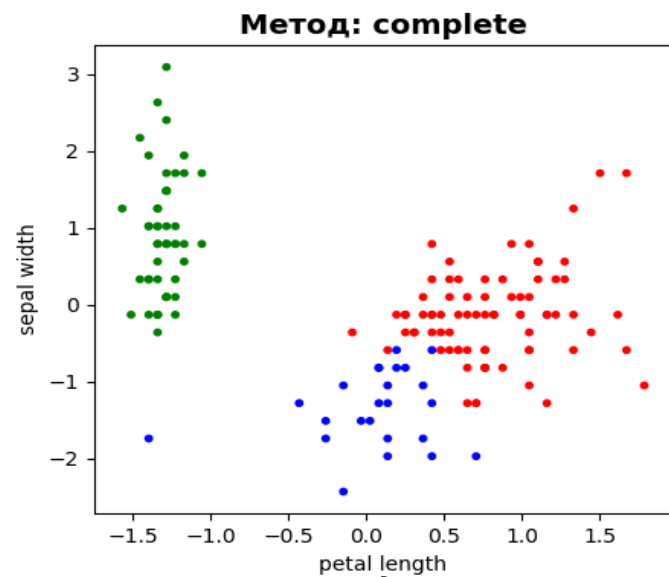
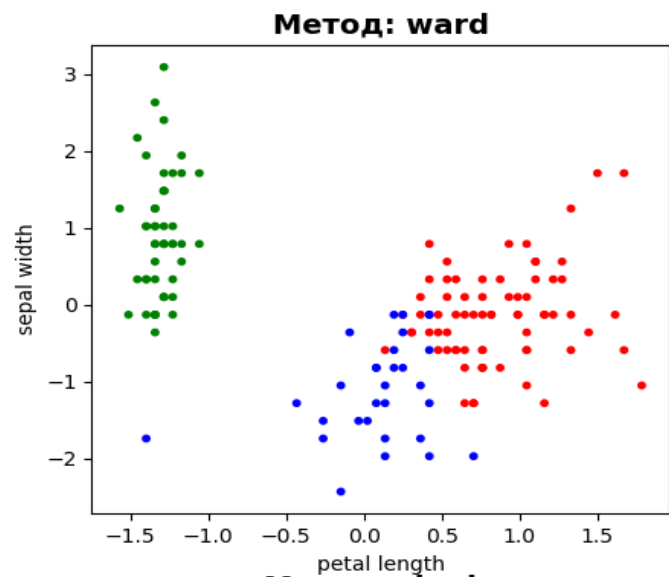
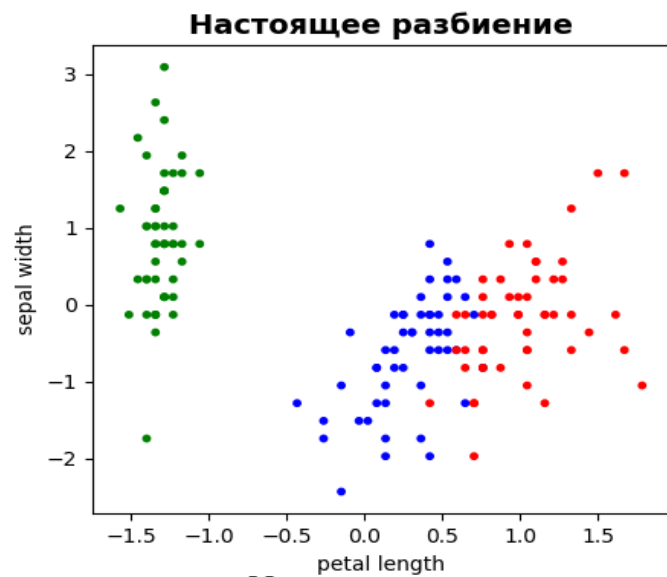
Агломеративный кластерный анализ

```
from sklearn.cluster import AgglomerativeClustering

links = ["ward", "complete", "average", "single"]

for link in links:

    ag = AgglomerativeClustering(n_clusters = 3, linkage = link)
    table['class_label_new'] = ag.fit_predict(
        table[['sepal_length', 'sepal_width',
                'petal_length', 'petal_width']])
    print(table)
```



Дендрограмма

Для анализа иерархической кластеризации принято использовать способ визуализации в виде дендрограмм. К сожалению, в Scikit-Learn нет готового метода, зато есть страница где дан код, как его написать [11].

Мы не будем разбирать данный код, а просто его скопируем и запустим.

Метод построения

```
*example2-3agl.py - E:\Works\Victor\Students\STDB\Term2\Lection3\example2-3agl.py (3.7.2)*
File Edit Format Run Options Window Help

import numpy as np
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram
from sklearn.cluster import AgglomerativeClustering

def plot_dendrogram(model, **kwargs):
    # Create linkage matrix and then plot the dendrogram

    # create the counts of samples under each node
    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 # leaf node
            else:
                current_count += counts[child_idx - n_samples]
        counts[i] = current_count

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

    # Plot the corresponding dendrogram
    dendrogram(linkage_matrix, **kwargs)
```

Ln: 48 Col: 0

Метод построения (текстом)

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram
from sklearn.cluster import AgglomerativeClustering

def plot_dendrogram(model, **kwargs):
    # Create linkage matrix and then plot the dendrogram

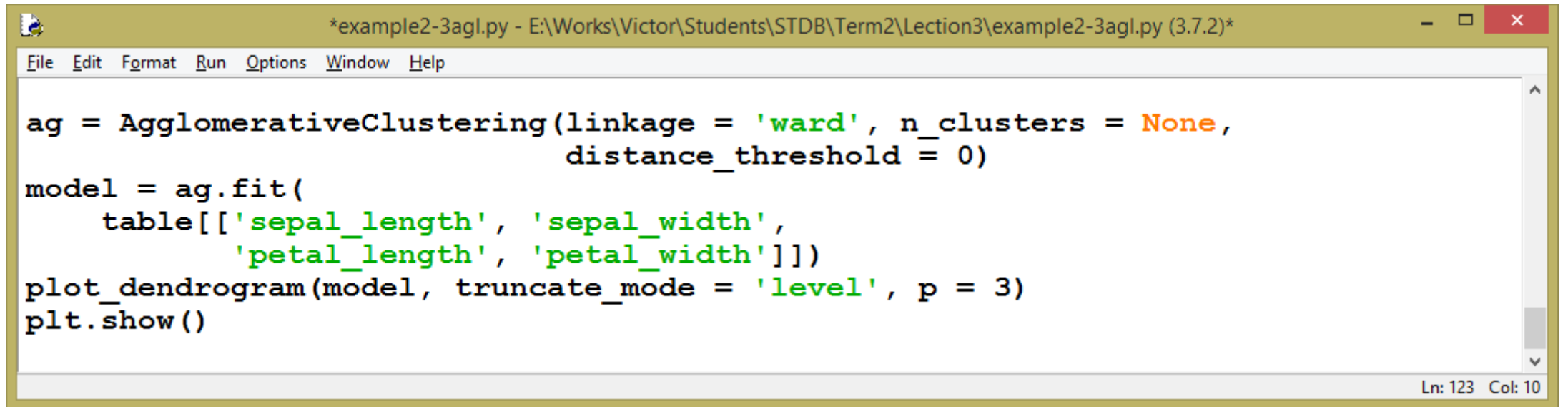
    # create the counts of samples under each node
    counts = np.zeros(model.children_.shape[0])
    n_samples = len(model.labels_)
```

```
    for i, merge in enumerate(model.children_):
        current_count = 0
        for child_idx in merge:
            if child_idx < n_samples:
                current_count += 1 # leaf node
            else:
                current_count += counts[child_idx - n_samples]
        counts[i] = current_count

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

    # Plot the corresponding dendrogram
    dendrogram(linkage_matrix, **kwargs)
```

Используем метод



The image shows a screenshot of a Python IDE window. The title bar reads '*example2-3agl.py - E:\Works\Victor\Students\STDB\Term2\Lection3\example2-3agl.py (3.7.2)*'. The menu bar includes 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The code editor contains the following Python code:

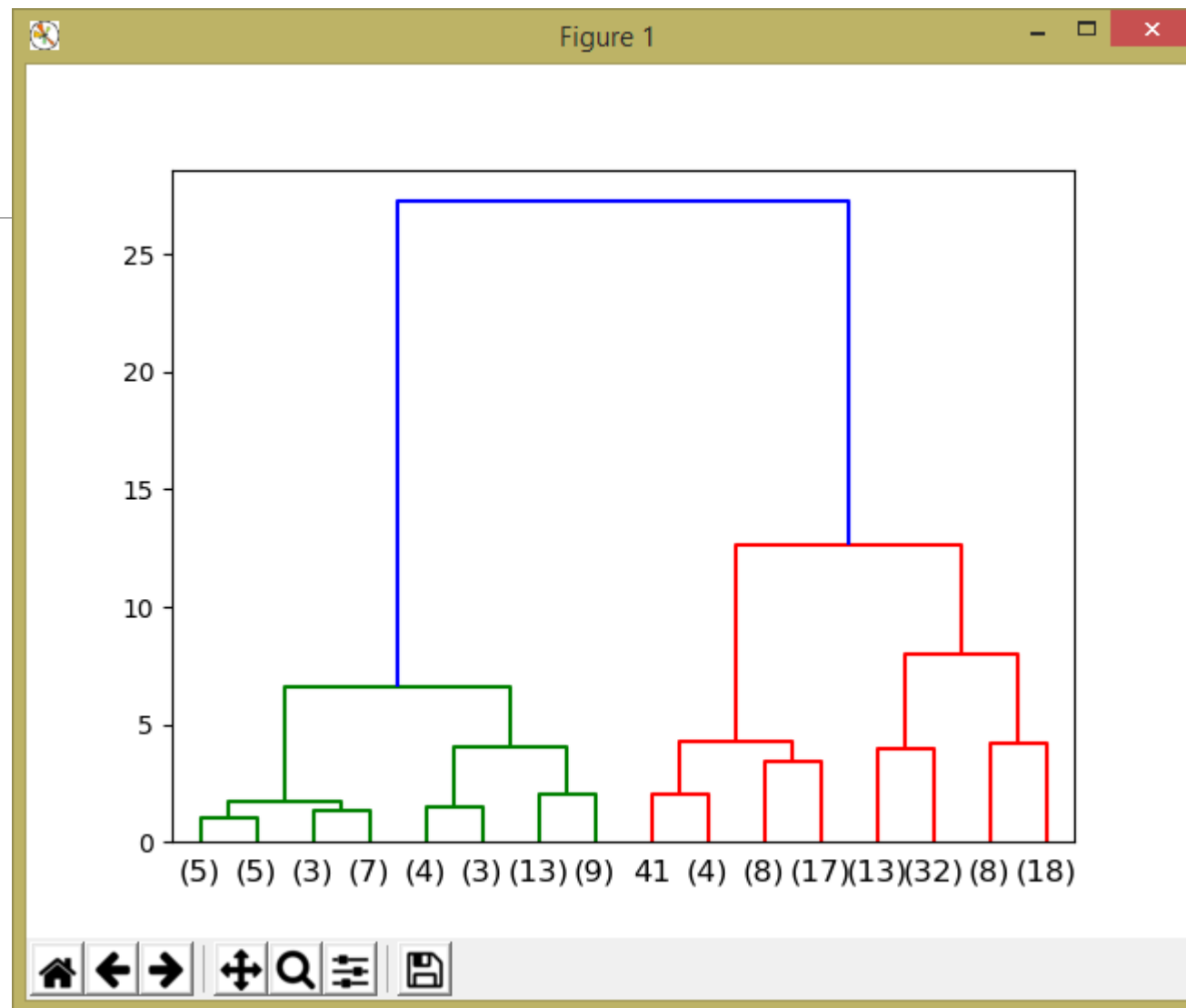
```
ag = AgglomerativeClustering(linkage = 'ward', n_clusters = None,  
                             distance_threshold = 0)  
model = ag.fit(  
    table[['sepal_length', 'sepal_width',  
          'petal_length', 'petal_width']])  
plot_dendrogram(model, truncate_mode = 'level', p = 3)  
plt.show()
```

The status bar at the bottom right indicates 'Ln: 123 Col: 10'.

Используем метод (текстом)

```
ag = AgglomerativeClustering(linkage = 'ward', n_clusters = None,  
                             distance_threshold = 0)  
  
model = ag.fit(  
    table[['sepal_length', 'sepal_width',  
          'petal_length', 'petal_width']])  
  
plot_dendrogram(model, truncate_mode = 'level', p = 3)  
  
plt.show()
```

Результат



Интерпретация

Определение числа кластеров по дендрограмме осуществляется выбором максимальной высоты, на которую мы можем спуститься сверху. Например, если мы можем спуститься только до глубины 15 снизу, то кластеров надо взять два. Если же мы можем спуститься до глубины 10, то кластеров нужно 3.

Интернет ресурсы и литература

1. https://ru.wikipedia.org/wiki/%D0%98%D1%80%D0%B8%D1%81%D1%8B_%D0%A4%D0%B8%D1%88%D0%B5%D1%80%D0%B0 – википедия об ирисах Фишера.
2. <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>
3. <http://espressocode.top/elbow-method-for-optimal-value-of-k-in-kmeans/>
4. https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html#sphx-glr-auto-examples-cluster-plot-kmeans-silhouette-analysis-py
5. https://github.com/tirthajyoti/Machine-Learning-with-Python/blob/master/Clustering-Dimensionality-Reduction/Clustering_metrics.ipynb
6. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.davies_bouldin_score.html#sklearn.metrics.davies_bouldin_score
7. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.homogeneity_score.html#sklearn.metrics.homogeneity_score
8. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.completeness_score.html#sklearn.metrics.completeness_score
9. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.v_measure_score.html#sklearn.metrics.v_measure_score
10. <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html#sklearn.cluster.AgglomerativeClustering>
11. https://scikit-learn.org/stable/auto_examples/cluster/plot_agglomerative_dendrogram.html#sphx-glr-auto-examples-cluster-plot-agglomerative-dendrogram-py