Рубежный контроль №2

Акушко А.С. РТ5-61Б

вариант 2

## Задача

Кластеризуйте данные с помощью двух алгоритмов кластеризации. Алгоритм: MeanShift и иерархическая кластеризация. Сравните качество кластеризации с помощью следующих метрик качества кластеризации (если это возможно для Вашего набора данных):

1. Adjusted Rand index
2. Adjusted Mutual Information
3. Homogeneity, completeness, V-measure
4. Коэффициент силуэта

Сделате выводы о том, какой алгоритм осуществляет более качественную кластеризацию на Вашем наборе данных. Набор данных №3: [https://scikit-](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html#sklearn.datasets.load_iris) [learn.org/stable/modules/generated/sklearn.datasets.load\_iris.html#sklearn.datasets.load\_iris](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html#sklearn.datasets.load_iris)

## Загрузка данных

In [1]:

**import numpy as np import pandas as pd**

**from typing import** Dict, Tuple

**from scipy import** stats

**from IPython.display import** Image

**from sklearn import** cluster, datasets, mixture **from sklearn.neighbors import** kneighbors\_graph **from sklearn.preprocessing import** StandardScaler **from sklearn.metrics import** adjusted\_rand\_score

**from sklearn.metrics import** adjusted\_mutual\_info\_score

**from sklearn.metrics import** homogeneity\_completeness\_v\_measure

**from sklearn.metrics import** silhouette\_score

**from sklearn.cluster import** MeanShift, AgglomerativeClustering

**from itertools import** cycle, islice

**import seaborn as sns**

**import matplotlib.pyplot as plt**

%**matplotlib** inline sns.set(style="ticks")

**from sklearn.datasets import** load\_iris

In [2]:

iris = load\_iris()

**for** x **in** iris: print(x)

data target

target\_names DESCR

feature\_names

filename

In [3]:

*# Признаки*

print(iris.feature\_names)

['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal wi dth (cm)']

In [4]:

*# Метки*

print(iris.target)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

2 2]

In [5]:

*# Имена меток*

print(iris.target\_names)

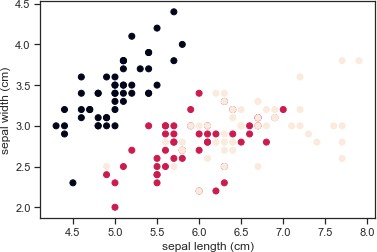
['setosa' 'versicolor' 'virginica']

In [6]:

*# Разделение набора данных* x\_axis = iris.data[:, 0] y\_axis = iris.data[:, 1]

In [7]:

*# Построение* plt.xlabel(iris.feature\_names[0]) plt.ylabel(iris.feature\_names[1]) plt.scatter(x\_axis, y\_axis, c=iris.target) plt.show()



In [8]:

data = pd.DataFrame(data= np.c\_[iris.data[:, 0], iris.data[:, 1]],

columns= ['total\_phenols', 'color\_intensity'])

In [9]:

data.head()

Out[9]:

**total\_phenols color\_intensity**

|  |  |  |
| --- | --- | --- |
| **0** | 5.1 | 3.5 |
| **1** | 4.9 | 3.0 |
| **2** | 4.7 | 3.2 |
| **3** | 4.6 | 3.1 |
| **4** | 5.0 | 3.6 |

In [10]:

data.head()

Out[10]:

**total\_phenols color\_intensity**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **0** | 5.1 | 3.5 |
| **1** | 4.9 | 3.0 |
| **2** | 4.7 | 3.2 |
| **3** | 4.6 | 3.1 |
| **4** | 5.0 | 3.6 |
| In [11]:  Out[11]: | data.shape (150, 2) |  |  |

In [12]:

**def** do\_clustering(cluster\_dataset, method):

*"""*

*Выполнение кластеризации для данных примера*

*"""*

temp\_cluster = method.fit\_predict(cluster\_dataset)

**return** temp\_cluster

In [13]:

**import warnings**

warnings.simplefilter(action='ignore', category=**FutureWarning**)

**def** claster\_metrics(method, data, true\_y):

*"""*

*Вычисление метрик кластеризации*

*"""*

result\_Method = do\_clustering(data, method)

list = []

list.append(adjusted\_rand\_score(true\_y, result\_Method)) list.append(adjusted\_mutual\_info\_score(true\_y, result\_Method))

h, c, v = homogeneity\_completeness\_v\_measure(true\_y, result\_Method) list.append(h)

list.append(c) list.append(v)

list.append(silhouette\_score(data, result\_Method))

names = ['ARI', 'AMI', 'Homogeneity', 'Completeness', 'V-measure', ' Silhouette']

**for** i **in** range(0,6):

print('**{}**: **{}**;'.format(names[i], list[i]))

# MeanShift

In [14]:

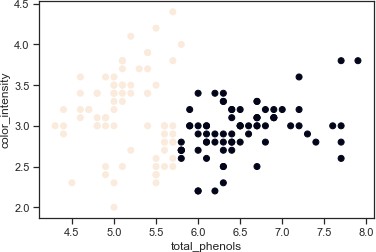
result\_MeanShift = do\_clustering(data, MeanShift())

In [15]:

plt.xlabel('total\_phenols') plt.ylabel('color\_intensity')

plt.scatter(data['total\_phenols'], data['color\_intensity'], c=result\_Mea nShift)

plt.show()



# Иерархическая кластеризация

In [16]:

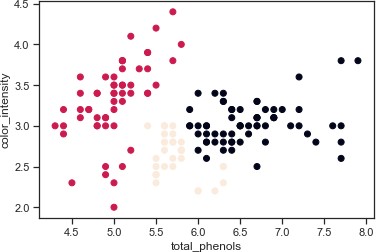
result\_AgglomerativeClustering = do\_clustering(data, AgglomerativeCluste ring(n\_clusters=3))

In [17]:

plt.xlabel('total\_phenols') plt.ylabel('color\_intensity')

plt.scatter(data['total\_phenols'], data['color\_intensity'], c=result\_Agg lomerativeClustering)

plt.show()



# Сравнение качества кластеризации

In [18]:

claster\_metrics(MeanShift(), data, iris.target)

ARI: 0.3944401908806803;

AMI: 0.43177435829008837;

Homogeneity: 0.355574438925241;

Completeness: 0.5636444355672562;

V-measure: 0.43606057162569084;

Silhouette: 0.4644681851183547;

In [19]:

claster\_metrics(AgglomerativeClustering(n\_clusters=3), data, iris.target

)

ARI: 0.5112126489117526;

AMI: 0.5240179186847518;

Homogeneity: 0.5190720845536648;

Completeness: 0.5414839345877656;

V-measure: 0.5300412040588491;

Silhouette: 0.3653346819163389;

Иерархическая кластеризация оказалась более качественной по сравнению с MeanShift.