Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

Дисциплина «Технологии машинного обучения»

Отчёт

по рубежному контролю №2

Тема: «Технологии использования и оценки моделей машинного обучения.»

Вариант 3

Студент:

Белкина Е.В.

Группа ИУ5-61Б

Преподаватель:

Гапанюк Ю.Е.

Москва, 2020 г.

Задание

Задача 2. ***Кластеризация данных*** (по вариантам)

Кластеризуйте данные с помощью двух алгоритмов кластеризации (варианты по группам приведены в таблице).

Группа	Алгоритм №1	Алгоритм №2
ИУ5-61Б, ИУ5Ц-81Б	K-Means	DBSCAN

Сравните качество кластеризации с помощью следующих метрик качества кластеризации (если это возможно для Вашего набора данных):

Adjusted Rand index

Adjusted Mutual Information

Homogeneity, completeness, V-measure

Коэффициент силуэта

Сделайте выводы о том, какой алгоритм осуществляет более качественную кластеризацию на Вашем наборе данных.

Набор данных:

https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_wine.html#sklearn.datasets.load_wine

Выполнение задания

1. Импортируем необходимые библиотеки с помощью команды import.

```
[ ]  import numpy as np
     import pandas as pd
     from typing import Dict, Tuple
     from scipy import stats
     from IPython.display import Image
     from sklearn.datasets import load_wine
     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 itertools import cycle, islice
     import seaborn as sns
     import matplotlib.pyplot as plt
     %matplotlib inline
     sns.set(style="ticks")
```

2. Импортируем датасет load_wine из sklearn в соответствии с заданием варианта

   Преобразуем датасет Scikit-learn в Pandas Dataframe

```
[ ]  #Создадим DataFrame для датасета wine
     wine = load_wine()
     data = pd.DataFrame(data=wine['data'], columns=wine['feature_names'])
     data.head()
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315_of_diluted_wines	proline
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065.0
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050.0
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185.0
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480.0
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735.0

3. Проверим наличие пропусков данных

```
[ ]  data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 13 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   alcohol                       178 non-null    float64
 1   malic_acid                    178 non-null    float64
 2   ash                           178 non-null    float64
 3   alcalinity_of_ash             178 non-null    float64
 4   magnesium                     178 non-null    float64
 5   total_phenols                 178 non-null    float64
 6   flavanoids                    178 non-null    float64
 7   nonflavanoid_phenols          178 non-null    float64
 8   proanthocyanins               178 non-null    float64
 9   color_intensity               178 non-null    float64
 10  hue                           178 non-null    float64
 11  od280/od315_of_diluted_wines  178 non-null    float64
 12  proline                       178 non-null    float64
dtypes: float64(13)
memory usage: 18.2 KB
```

```
[ ]  data.isnull().sum()
```

```
alcohol                         0
malic_acid                      0
ash                             0
alcalinity_of_ash               0
magnesium                       0
total_phenols                   0
flavanoids                      0
nonflavanoid_phenols            0
proanthocyanins                 0
color_intensity                 0
hue                             0
od280/od315_of_diluted_wines    0
proline                         0
dtype: int64
```

Можем видеть, что пропуски данных в датасете отсутствуют.

4. Подбор гиперпараметра количества кластеров

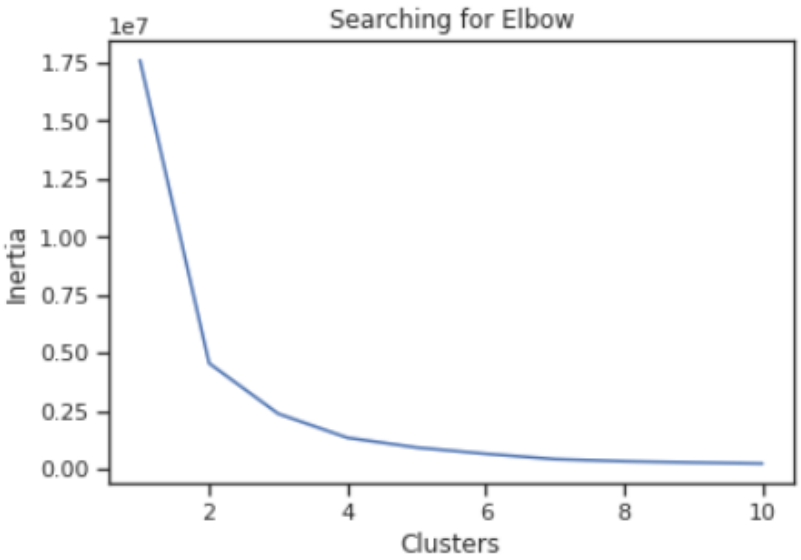
```
[ ]  X = data
```

```
clusters = []

for i in range(1, 11):
    km = KMeans(n_clusters=i).fit(X)
    clusters.append(km.inertia_)

fig, ax = plt.subplots(figsize=(6, 4))
sns.lineplot(x=list(range(1, 11)), y=clusters, ax=ax)
ax.set_title('Searching for Elbow')
ax.set_xlabel('Clusters')
ax.set_ylabel('Inertia')

plt.show()
```



Используем "правило локтя".

Видим, что после 4 кластеров уменьшение инерции резко замедляется.

5. Кластеризация K-Means

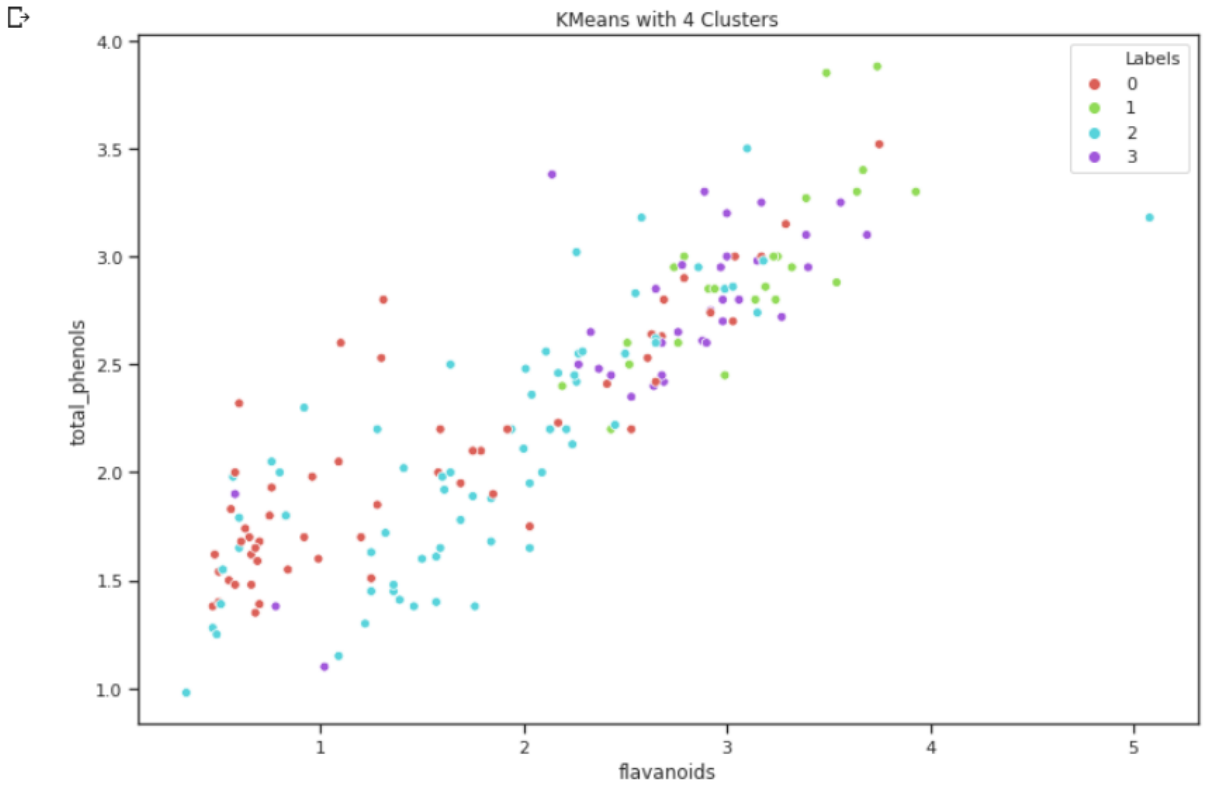
The KMeans algorithm clusters data by trying to separate samples in n groups of equal variance, minimizing a criterion known as the inertia or within-cluster sum-of-squares (see below). This algorithm requires the number of clusters to be specified. It scales well to large number of samples and has been used across a large range of application areas in many different fields.

n_clusters*int, default=8*

The number of clusters to form as well as the number of centroids to generate.

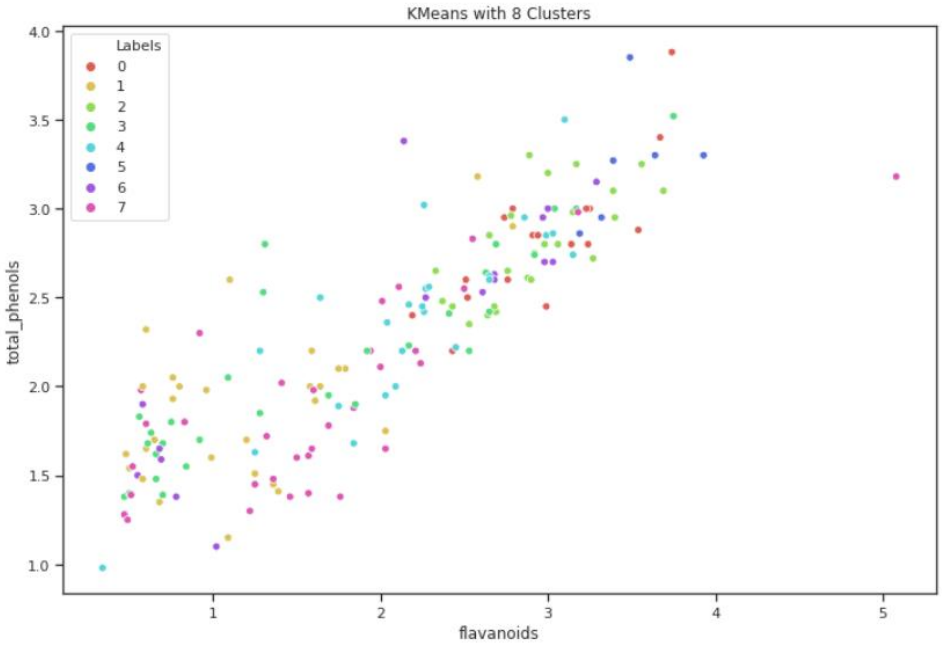
```
[ ]  km4 = KMeans(n_clusters=4).fit(X)

     X['Labels'] = km4.labels_
     plt.figure(figsize=(12, 8))
     sns.scatterplot(X['flavanoids'], X['total_phenols'], hue=X['Labels'],
                     palette=sns.color_palette('hls', 4))
     plt.title('KMeans with 4 Clusters')
     plt.show()
```



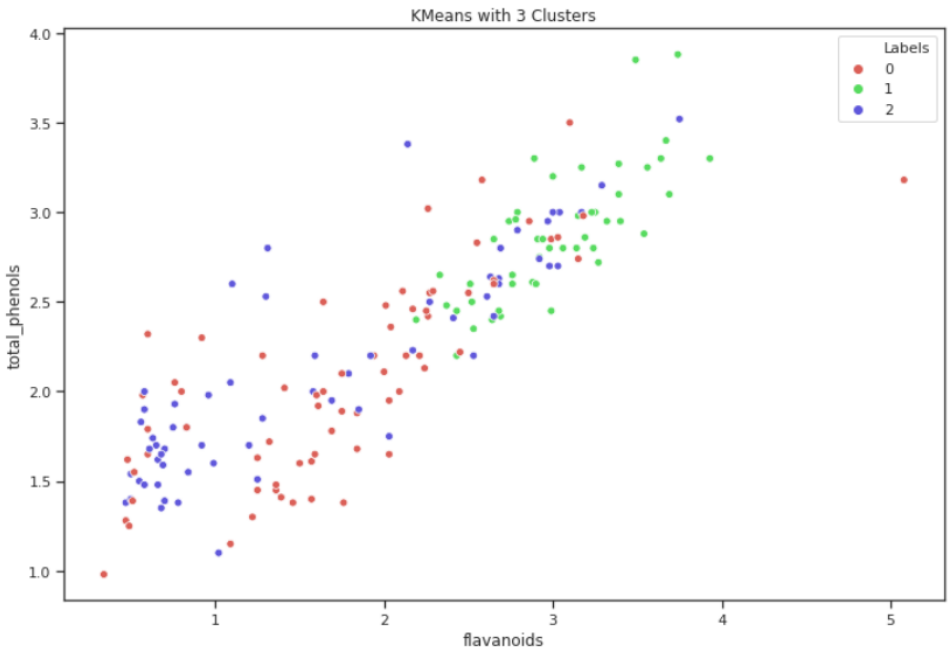
```
km8 = KMeans(n_clusters=8).fit(X)

X['Labels'] = km8.labels_
plt.figure(figsize=(12, 8))
sns.scatterplot(X['flavanoids'], X['total_phenols'], hue=X['Labels'],
                palette=sns.color_palette('hls', 8))
plt.title('KMeans with 8 Clusters')
plt.show()
```




```
km3 = KMeans(n_clusters=3).fit(X)

X['Labels'] = km3.labels_
plt.figure(figsize=(12, 8))
sns.scatterplot(X['flavanoids'], X['total_phenols'], hue=X['Labels'],
                palette=sns.color_palette('hls', 3))
plt.title('KMeans with 3 Clusters')
plt.show()
```



6. Кластеризация DBSCAN

Perform DBSCAN clustering from vector array or distance matrix.

DBSCAN - Density-Based Spatial Clustering of Applications with Noise. Finds core samples of high density and expands clusters from them. Good for data which contains clusters of similar density.

psfloat, default=0.5

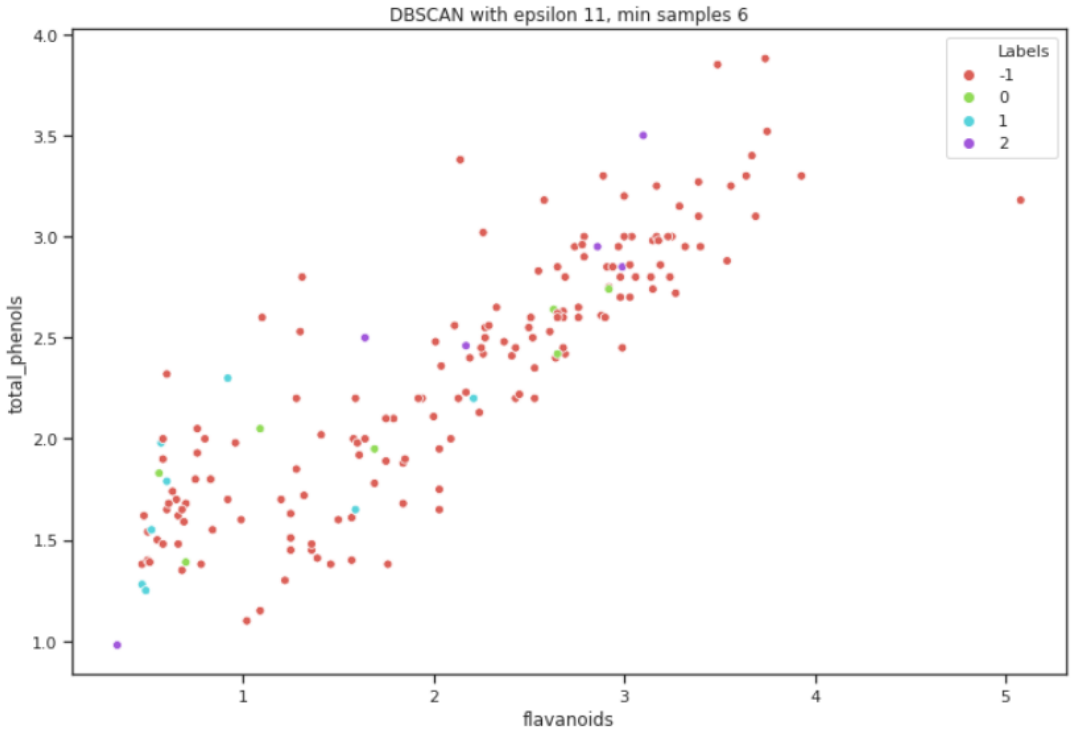
> The maximum distance between two samples for one to be considered as in the neighborhood of the other. This is not a maximum bound on the distances of points within a cluster. This is the most important DBSCAN parameter to choose appropriately for your data set and distance function.

min_samplesint, default=5

> The number of samples (or total weight) in a neighborhood for a point to be considered as a core point. This includes the point itself.

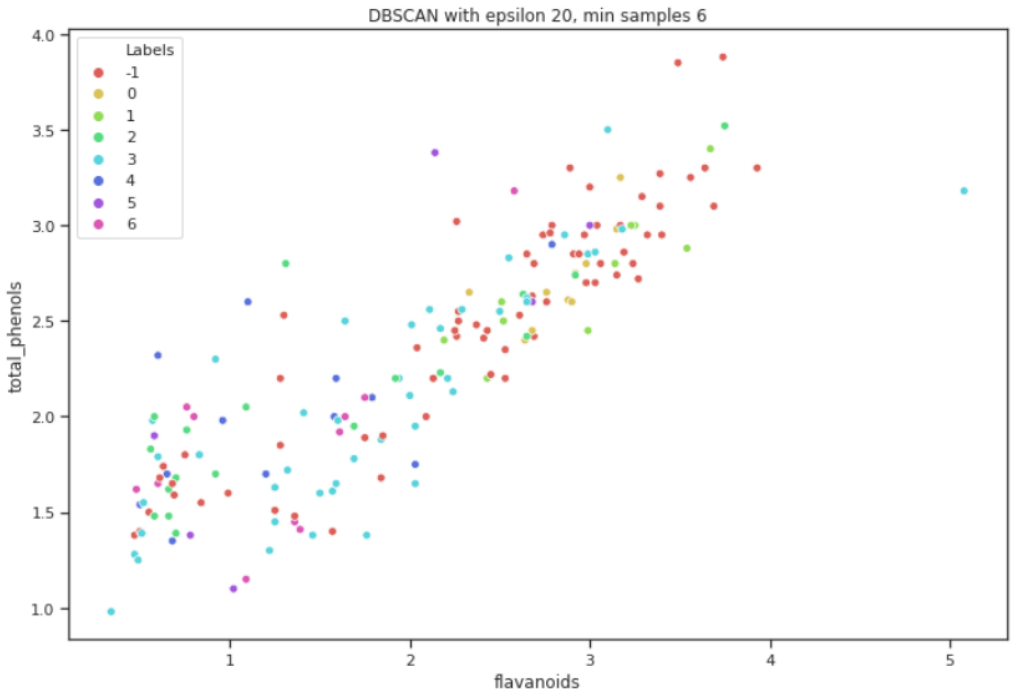
```
db = DBSCAN(eps=11, min_samples=6).fit(X)

X['Labels'] = db.labels_
plt.figure(figsize=(12, 8))
sns.scatterplot(X['flavanoids'], X['total_phenols'], hue=X['Labels'],
                palette=sns.color_palette('hls', np.unique(db.labels_).shape[0]))
plt.title('DBSCAN with epsilon 11, min samples 6')
plt.show()
```



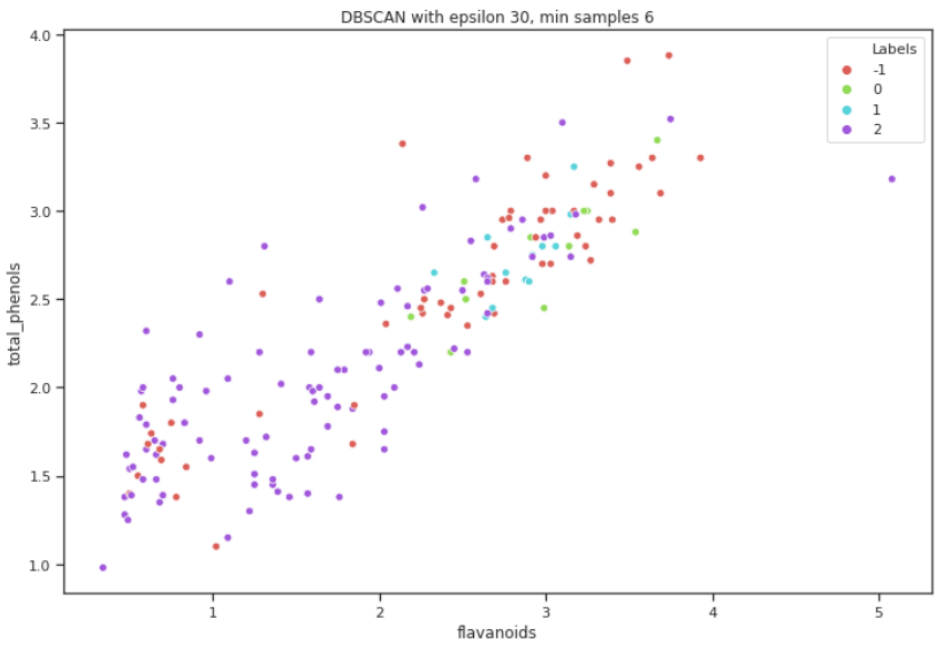
```
db = DBSCAN(eps=20, min_samples=6).fit(X)

X['Labels'] = db.labels_
plt.figure(figsize=(12, 8))
sns.scatterplot(X['flavanoids'], X['total_phenols'], hue=X['Labels'],
                palette=sns.color_palette('hls', np.unique(db.labels_).shape[0]))
plt.title('DBSCAN with epsilon 20, min samples 6')
plt.show()
```



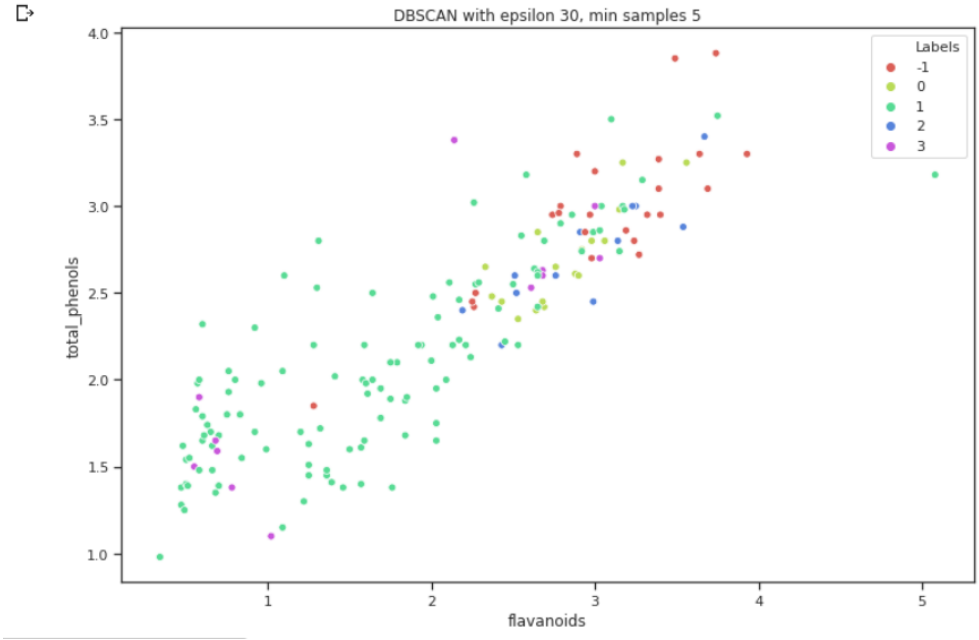

```
db = DBSCAN(eps=30, min_samples=10).fit(X)

X['Labels'] = db.labels_
plt.figure(figsize=(12, 8))
sns.scatterplot(X['flavanoids'], X['total_phenols'], hue=X['Labels'],
                palette=sns.color_palette('hls', np.unique(db.labels_).shape[0]))
plt.title('DBSCAN with epsilon 30, min samples 10')
plt.show()
```



```
db = DBSCAN(eps=30, min_samples=6).fit(X)

X['Labels'] = db.labels_
plt.figure(figsize=(12, 8))
sns.scatterplot(X['flavanoids'], X['total_phenols'], hue=X['Labels'],
                palette=sns.color_palette('hls', np.unique(db.labels_).shape[0]))
plt.title('DBSCAN with epsilon 30, min samples 5')
plt.show()
```



7. Метрики качества кластеризации

1) Adjusted Rand index

Метрика возвращает результат в диапазоне [-1;+1]. Значение близкое к +1 говорит об очень хорошем качестве кластеризации. Значение близкое к 0 соответствует случайным разбиениям. Отрицательные значения говорят о плохом качестве кластеризации.

2) Adjusted Mutual Information

Значение близкое к +1 говорит об очень хорошем качестве кластеризации. Значение близкое к 0 соответствует случайным разбиениям.

3) Homogeneity, completeness, V-measure

Homogeneity - каждый кластер содержит только представителей единственного класса (под классом понимается истинное значение метки кластера). Значение в диапазоне [0;1], 1 говорит об очень хорошем качестве кластеризации.

Completeness - все элементы одного класса помещены в один и тот же кластер. Значение в диапазоне [0;1], 1 говорит об очень хорошем качестве кластеризации.

V-measure - среднее гармоническое от Homogeneity и Completeness.

4) Коэффициент силуэта

Данный метод не требует знания истинных значений меток кластеров,ииспользуется подбора оптимального числа кластеров— выбирается число кластеров, максимизирующее значение силуэта.

Силуэтом выборки показывает, насколько среднее расстояние до объектов своего кластера отличается от среднего расстояния до объектов других кластеров. Данная величина лежит в диапазоне [-1;1]. Значения, близкие к -1, соответствуют плохим (разрозненным) кластеризациям, значения, близкие к нулю, говорят о том, что кластеры пересекаются и накладываются друг на друга, значения, близкие к 1, соответствуют "плотным" четко выделенным кластерам. Таким образом, чем больше силуэт, тем более четко выделены кластеры, и они представляют собой компактные, плотно сгруппированные облака точек.

```
from sklearn import metrics
import pandas as pd
from sklearn.cluster import KMeans, DBSCAN

algorithms = []
algorithms.append(KMeans(n_clusters=3, random_state=1))
algorithms.append(DBSCAN(eps=30, min_samples=10))


y= wine.target
data = []
for algo in algorithms:
    algo.fit(X)
    data.append(({
        'ARI': metrics.adjusted_rand_score(y, algo.labels_),
        'AMI': metrics.adjusted_mutual_info_score(y, algo.labels_),
        'Homogenity': metrics.homogeneity_score(y, algo.labels_),
        'Completeness': metrics.completeness_score(y, algo.labels_),
        'V-measure': metrics.v_measure_score(y, algo.labels_),
        'Silhouette': metrics.silhouette_score(X, algo.labels_)}))

results = pd.DataFrame(data=data, columns=['ARI', 'AMI', 'Homogenity',
                                           'Completeness', 'V-measure',
                                           'Silhouette'],
                       index=['K-means', 'DBSCAN'])

results
```

	ARI	AMI	Homogenity	Completeness	V-measure	Silhouette
K-means	0.371114	0.422687	0.428812	0.428701	0.428757	0.571157
DBSCAN	0.292739	0.362274	0.365761	0.380503	0.372987	0.363418

8. Выводы

По данным полученным данным метрик (ARI, AMI, Homogenity, Completeness, V-measure, Silhouette) можем сделать следующие выводы:

- Алгоритм K-Means осуществляет более качественную кластеризацию на используемом в данной работе наборе данных, чем алгоритм DBSCAN, так как значения метрик, полученные для K-Means более близки к 1, что свидетельствует о более высоком качестве кластеризации.