

```
In [1]:
```

```
import os  
os.environ["OMP_NUM_THREADS"] = '2'
```

```
In [2]:
```

```
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
import scipy.cluster.hierarchy as sch  
import plotly.express as px  
from sklearn.metrics import silhouette_score, adjusted_rand_score, v_measure_score, homogeneity_score,  
from sklearn.preprocessing import StandardScaler  
from sklearn.decomposition import PCA  
from sklearn.cluster import AgglomerativeClustering, KMeans, DBSCAN  
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster  
from scipy.stats import multivariate_normal  
from sklearn.manifold import TSNE  
from fcmeans import FCM  
from umap import UMAP  
  
%matplotlib inline
```

Кластерный анализ

Задача кластерного анализа – выделение «сгущений» точек, каждой из которых соответствует строка исследуемого набора данных, на однородные группы (кластеры) объектов.

В задаче классификации (при обучении с учителем) известны метки классов для всех объектов обучающей выборки, требуется классифицировать объекты тестовой выборки (новые объекты) – определить метки классов для всех объектов тестовой выборки (для новых объектов).

В задаче кластеризации (при обучении без учителя) ни для одного объекта обучающей выборки метка кластера неизвестна, тем не менее, требуется классифицировать все объекты обучающей выборки – назначить им метки кластеров.

Получаем данные

In [3]:

```
data = pd.read_csv("new_horse_data.csv")
data.head(5)
```

Out[3]:

	surgery	age	hospital_number	rectal_temperature	pulse	respiratory_rate	temperature_of_extremities	perip
0	2.0	1	530101	38.5	66.0	28.0		3.0
1	1.0	1	534817	39.2	88.0	20.0		NaN
2	2.0	1	530334	38.3	40.0	24.0		1.0
3	1.0	9	5290409	39.1	164.0	84.0		4.0
4	2.0	1	530255	37.3	104.0	35.0		NaN

5 rows × 28 columns

Проводим анализ типов и пропусков

In [4]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300 entries, 0 to 299
Data columns (total 28 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   surgery          299 non-null    float64
 1   age              300 non-null    int64  
 2   hospital_number  300 non-null    int64  
 3   rectal_temperature 240 non-null    float64
 4   pulse             276 non-null    float64
 5   respiratory_rate 242 non-null    float64
 6   temperature_of_extremities 244 non-null    float64
 7   peripheral_pulse 231 non-null    float64
 8   mucous_membranes 253 non-null    float64
 9   capillary_refill_time 268 non-null    float64
 10  pain              245 non-null    float64
 11  peristalsis       256 non-null    float64
 12  abdominal_distension 244 non-null    float64
 13  nasogastric_tube  196 non-null    float64
 14  nasogastric_tube.1 194 non-null    float64
 15  nasogastric_reflux_PH 53 non-null    float64
 16  rectal_examination 198 non-null    float64
 17  abdomen           182 non-null    float64
 18  packed_cell_volume 271 non-null    float64
 19  total_protein     267 non-null    float64
 20  abdominocentesis_appearance 135 non-null    float64
 21  abdomecentesis_total_protein 102 non-null    float64
 22  outcome            299 non-null    float64
 23  surgical_lesion    300 non-null    int64  
 24  first_type_of_lesion 300 non-null    int64  
 25  second_type_of_lesion 300 non-null    int64  
 26  third_type_of_lesion 300 non-null    int64  
 27  cp_data            300 non-null    int64  
dtypes: float64(21), int64(7)
memory usage: 65.8 KB
```

1. : surgery?

```
1 = Yes, it had surgery
2 = It was treated without surgery
```

2. : Age

- 1 = Adult horse
- 2 = Young (< 6 months)

3. : Hospital Number

- numeric id
- the case number assigned to the horse
- (may not be unique if the horse is treated > 1 time)

4. : rectal temperature

- linear
- in degrees celsius.
- An elevated temp may occur due to infection.
- temperature may be reduced when the animal is in late shock
- normal temp is 37.8
- this parameter will usually change as the problem progresses
 - eg. may start out normal, then become elevated because of the lesion, passing back through the normal range as the horse goes into shock

5. : pulse

- linear
- the heart rate in beats per minute
- is a reflection of the heart condition: 30 -40 is normal for adults
- rare to have a lower than normal rate although athletic horses may have a rate of 20-25
- animals with painful lesions or suffering from circulatory shock may have an elevated heart rate

6. : respiratory rate

- linear
- normal rate is 8 to 10
- usefulness is doubtful due to the great fluctuations

7. : temperature of extremities

- a subjective indication of peripheral circulation
- possible values:
 - 1 = Normal
 - 2 = Warm
 - 3 = Cool
 - 4 = Cold
- cool to cold extremities indicate possible shock
- hot extremities should correlate with an elevated rectal temp.

8. : peripheral pulse

- subjective
- possible values are:
 - 1 = normal
 - 2 = increased
 - 3 = reduced
 - 4 = absent
- normal or increased p.p. are indicative of adequate circulation while reduced or absent indicate poor perfusion

9. : mucous membranes

- a subjective measurement of colour
- possible values are:
 - 1 = normal pink
 - 2 = bright pink
 - 3 = pale pink
 - 4 = pale cyanotic
 - 5 = bright red / injected
 - 6 = dark cyanotic
- 1 and 2 probably indicate a normal or slightly increased circulation
- 3 may occur in early shock
- 4 and 6 are indicative of serious circulatory compromise
- 5 is more indicative of a septicemia

10. : capillary refill time

- a clinical judgement. The longer the refill, the poorer the circulation
- possible values
 - 1 = < 3 seconds
 - 2 = >= 3 seconds

11. : pain - a subjective judgement of the horse's pain level

- possible values:
 - 1 = alert, no pain
 - 2 = depressed
 - 3 = intermittent mild pain
 - 4 = intermittent severe pain
 - 5 = continuous severe pain
- should NOT be treated as a ordered or discrete variable!
- In general, the more painful, the more likely it is to require surgery
- prior treatment of pain may mask the pain level to some extent

12. : peristalsis

- an indication of the activity in the horse's gut. As the gut becomes more distended or the horse becomes more toxic, the activity decreases
- possible values:
 - 1 = hypermotile
 - 2 = normal
 - 3 = hypomotile
 - 4 = absent

13. : abdominal distension

- An IMPORTANT parameter.
- possible values
 - 1 = none
 - 2 = slight
 - 3 = moderate
 - 4 = severe
- an animal with abdominal distension is likely to be painful and have reduced gut motility.
- a horse with severe abdominal distension is likely to require surgery just to relieve the pressure

14. : nasogastric tube

- this refers to any gas coming out of the tube
- possible values:
 - 1 = none
 - 2 = slight
 - 3 = significant
- a large gas cap in the stomach is likely to give the horse discomfort

15. : nasogastric reflux

- possible values
 - 1 = none
 - 2 = > 1 liter
 - 3 = < 1 liter
- the greater amount of reflux, the more likelihood that there is some serious obstruction to the fluid passage from the rest of the intestine

16. : nasogastric reflux PH

- linear
- scale is from 0 to 14 with 7 being neutral
- normal values are in the 3 to 4 range

17. : rectal examination - feces

- possible values
 - 1 = normal
 - 2 = increased
 - 3 = decreased
 - 4 = absent
- absent feces probably indicates an obstruction

18. : abdomen

- possible values
 - 1 = normal
 - 2 = other
 - 3 = firm feces in the large intestine
 - 4 = distended small intestine
 - 5 = distended large intestine
- 3 is probably an obstruction caused by a mechanical impaction and is normally treated medically
- 4 and 5 indicate a surgical lesion

19. : packed cell volume

- linear
- the # of red cells by volume in the blood
- normal range is 30 to 50. The level rises as the circulation becomes compromised or as the animal becomes dehydrated.

20. : total protein

- linear
- normal values lie in the 6-7.5 (gms/dL) range
- the higher the value the greater the dehydration

21. : abdominocentesis appearance

- a needle is put in the horse's abdomen and fluid is obtained from the abdominal cavity
- possible values:
 - 1 = clear
 - 2 = cloudy
 - 3 = serosanguinous
- normal fluid is clear while cloudy or serosanguinous indicates a compromised gut

22. : abdomcentesis total protein

- linear
- the higher the level of protein the more likely it is to have a compromised gut. Values are in gms/dL

23. : outcome

- what eventually happened to the horse?
- possible values:
 - 1 = lived
 - 2 = died
 - 3 = was euthanized

24. : surgical lesion?

- retrospectively, was the problem (lesion) surgical?
- all cases are either operated upon or autopsied so that this value and the lesion type are always known
- possible values:
 - 1 = Yes
 - 2 = No

25, 26, 27: type of lesion

```
- first number is site of lesion  
1 = gastric  
2 = sm intestine  
3 = lg colon  
4 = lg colon and cecum  
5 = cecum  
6 = transverse colon  
7 = retum/descending colon  
8 = uterus  
9 = bladder  
11 = all intestinal sites
```

Удалим неинформативные данные, второй и третий типы повреждений, и данные с количеством пропусков больше 30%:

In [5]:

```
data = data.drop(data.columns[[2, 6, 9, 10, 14, 15, 18, 19, 20, 21, 25, 26, 27]], axis = 1)  
data
```

Out[5]:

	surgery	age	rectal_temperature	pulse	respiratory_rate	peripheral_pulse	mucous_membranes	peristalsis
0	2.0	1	38.5	66.0	28.0	3.0	NaN	4.
1	1.0	1	39.2	88.0	20.0	NaN	4.0	4.
2	2.0	1	38.3	40.0	24.0	1.0	3.0	3.
3	1.0	9	39.1	164.0	84.0	1.0	6.0	4.
4	2.0	1	37.3	104.0	35.0	NaN	6.0	NaN
...
295	1.0	1	NaN	120.0	70.0	NaN	4.0	4.
296	2.0	1	37.2	72.0	24.0	2.0	4.0	3.
297	1.0	1	37.5	72.0	30.0	3.0	4.0	4.
298	1.0	1	36.5	100.0	24.0	3.0	3.0	3.
299	1.0	1	37.2	40.0	20.0	NaN	NaN	NaN

300 rows × 15 columns

Заполним оставшиеся данные медианными значениями:

In [6]:

```
data.fillna(data.median(numeric_only=True).round(1), inplace=True)
```

Проведем стандартизацию данных:

In [7]:

```
column_name = data.columns
```

In [8]:

```
scaler = StandardScaler()
scaler.fit(data)
data = pd.DataFrame(scaler.transform(data), columns=column_name)
data.head()
```

Out[8]:

	surgery	age	rectal_temperature	pulse	respiratory_rate	peripheral_pulse	mucouc_membranes
0	1.233292	-0.294884		0.498160	-0.192031	-0.079662	1.080954
1	-0.810838	-0.294884		1.568924	0.608097	-0.580154	-0.014607
2	1.233292	-0.294884		0.192228	-1.137636	-0.329908	-1.110169
3	-0.810838	3.391165		1.415958	3.372175	3.423782	-1.110169
4	1.233292	-0.294884		-1.337435	1.190008	0.358269	-0.014607

Разделим переменные на целевую и предикторы:

In [9]:

```
target = data[data.columns[[-1]]]
data = data.drop(data[data.columns[[-1]]], axis=1)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300 entries, 0 to 299
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   surgery          300 non-null    float64
 1   age              300 non-null    float64
 2   rectal_temperature 300 non-null    float64
 3   pulse             300 non-null    float64
 4   respiratory_rate  300 non-null    float64
 5   peripheral_pulse 300 non-null    float64
 6   mucouc_membranes  300 non-null    float64
 7   peristalsis       300 non-null    float64
 8   abdominal_distension 300 non-null    float64
 9   nasogastric_tube  300 non-null    float64
 10  rectal_examination 300 non-null    float64
 11  abdomen          300 non-null    float64
 12  outcome           300 non-null    float64
 13  surgical_lesion  300 non-null    float64
dtypes: float64(14)
memory usage: 32.9 KB
```

Преобразуем в целочисленные данные целевые значения:

In [10]:

```
spisok = target.groupby("first_type_of_lesion").count()
clusters = {}
count = 1
for x in spisok.index:
    clusters[x] = count
    count += 1
```

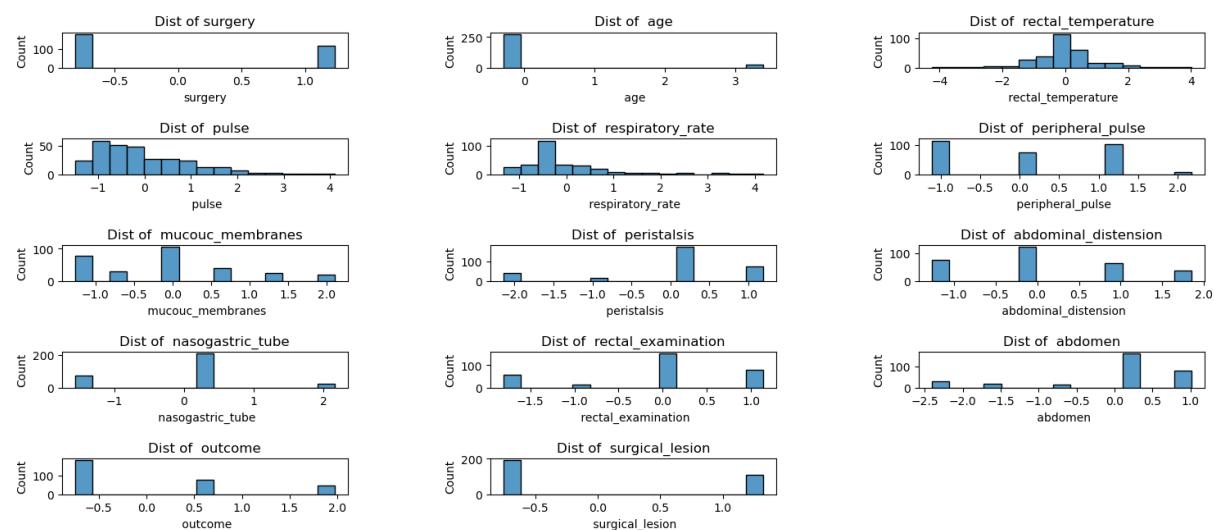
In [11]:

```
for cluster in clusters:  
    target['first_type_of_lesion'].mask(target['first_type_of_lesion'] == cluster, clusters[cluster])
```

Оценим распределение признаков:

In [12]:

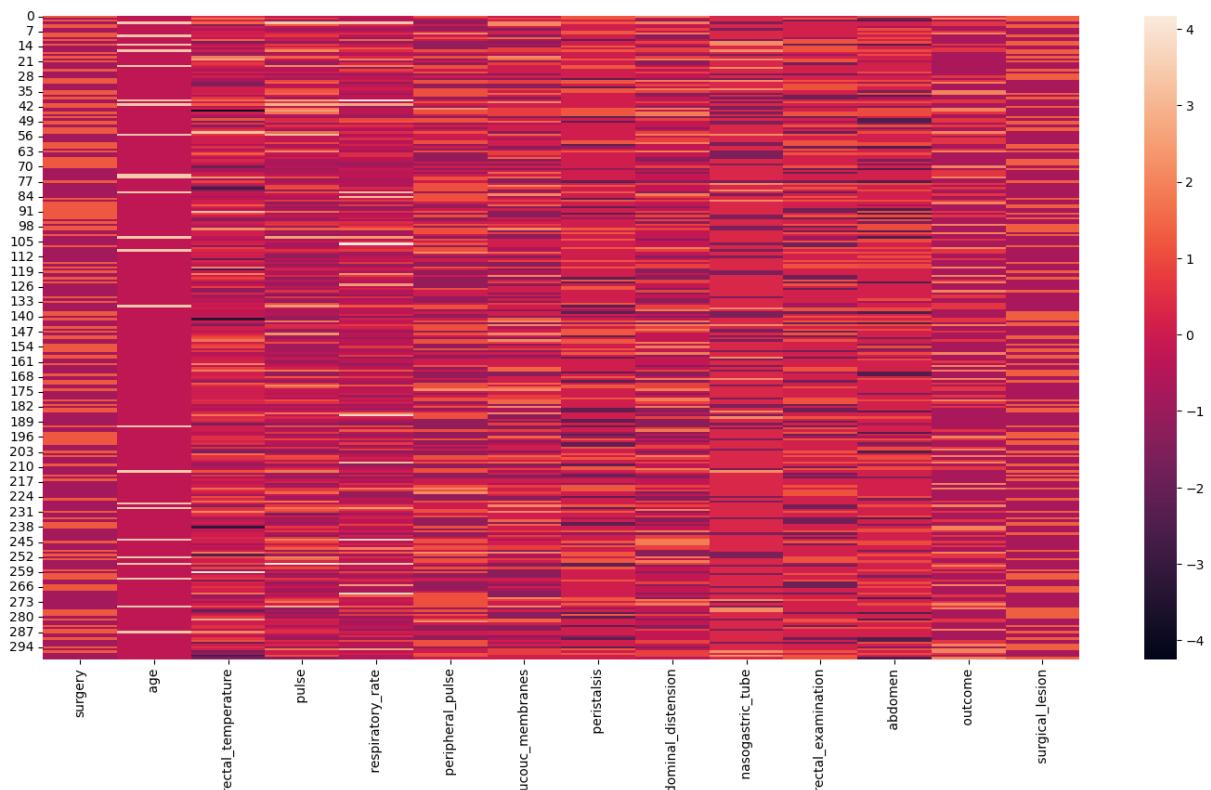
```
plt.figure(1 , figsize = (18, 9))  
n = 0  
for x in data.columns:  
    n += 1  
    plt.subplot(6 , 3 , n)  
    plt.subplots_adjust(hspace = 2, wspace = 0.5)  
    sns.histplot(data[x] , bins = 15)  
    plt.title('Dist of {}'.format(x))  
plt.show()
```



Тепловая карта - это метод визуализации данных, который показывает величину явления в виде цвета в двух измерениях. Изменение цвета может быть связано с оттенком или интенсивностью, что дает читателю очевидные визуальные подсказки о том, как явление сгруппировано или изменяется в пространстве.

In [13]:

```
plt.figure(1, figsize = (18 , 9))
sns.heatmap(data)
plt.show()
```



Изучим нормальность и зависимость данных:

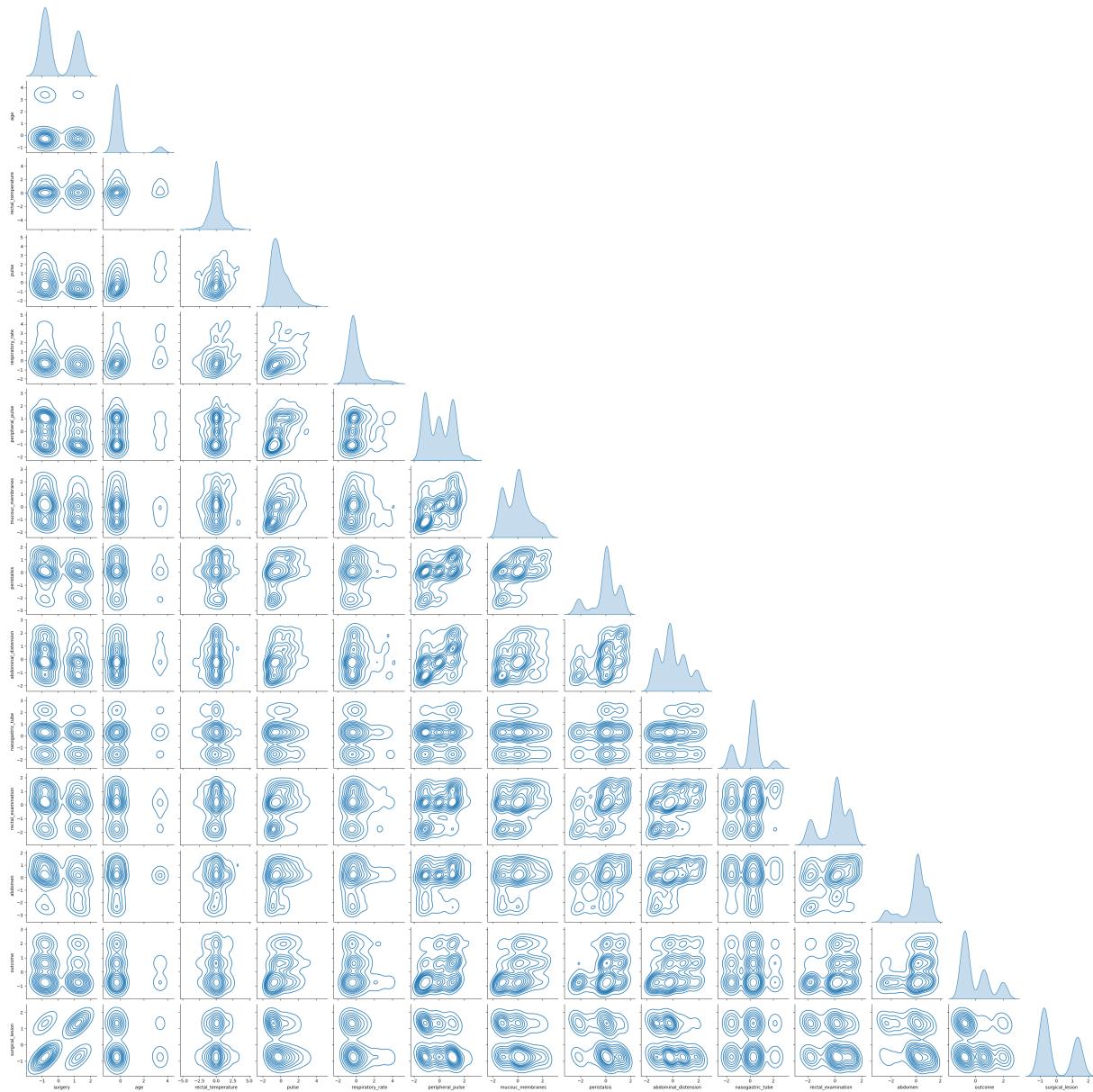
In [14]:

```
plt.figure(1, figsize = (18 ,9))
sns.pairplot(data, kind = 'kde', corner = True)
```

Out[14]:

```
<seaborn.axisgrid.PairGrid at 0x1addaa56370>
```

```
<Figure size 1800x900 with 0 Axes>
```



Данные не являются нормальными.

Иерархический алгоритм кластеризации данных

Агломеративная кластеризация - это подход "снизу вверх": каждое наблюдение начинается в своем собственном кластере, и пары кластеров объединяются по мере продвижения вверх по иерархии.

Дендрограмма - это диаграмма, представляющая дерево. Это схематическое представление часто используется в различных контекстах: при иерархической кластеризации оно иллюстрирует расположение кластеров, полученных в результате соответствующего анализа.

Построим дендрограмму, где расстояние между кластерами вычисляется по методу Варда, а расстояние между объектами по евклидовому.

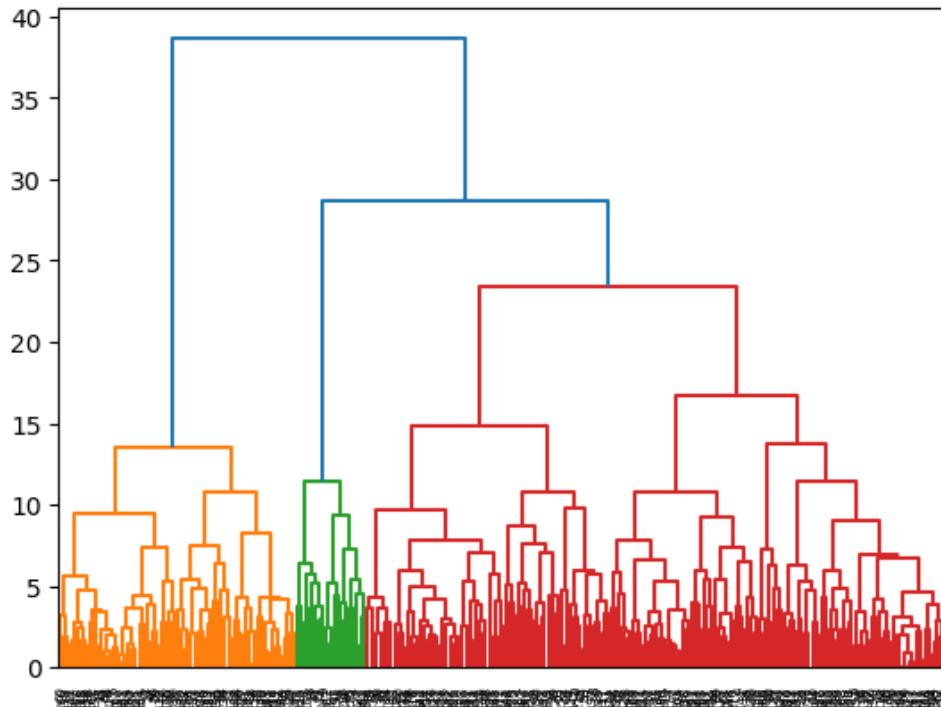
В качестве предикторов возьмем переменные отвечающие за пульс и частоту дыхания.

In [15]:

```
new_data = data.copy()
```

In [16]:

```
link = linkage(new_data, 'ward', 'euclidean')
dn = dendrogram(link)
```



Информацию о кластерах добавляем в таблицу данных в виде столбца по критерию максимального количества кластеров равным 4:

In [17]:

```
new_data["cluster"] = fcluster(link, 4, criterion='maxclust')
new_data.groupby('cluster').mean()
```

Out[17]:

cluster	surgery	age	rectal_temperature	pulse	respiratory_rate	peripheral_pulse	mucous_membrane	
1	1.028879	-0.294884		-0.056343	-0.691656	-0.383476	-0.740417	-0.573
2	-0.299806	3.391165		0.606511	1.714335	1.270624	0.076689	-0.001
3	-0.195136	-0.294884		0.380210	0.595390	0.086541	1.080954	0.723
4	-0.521404	-0.294884		-0.368197	-0.311760	-0.061945	-0.286074	-0.125

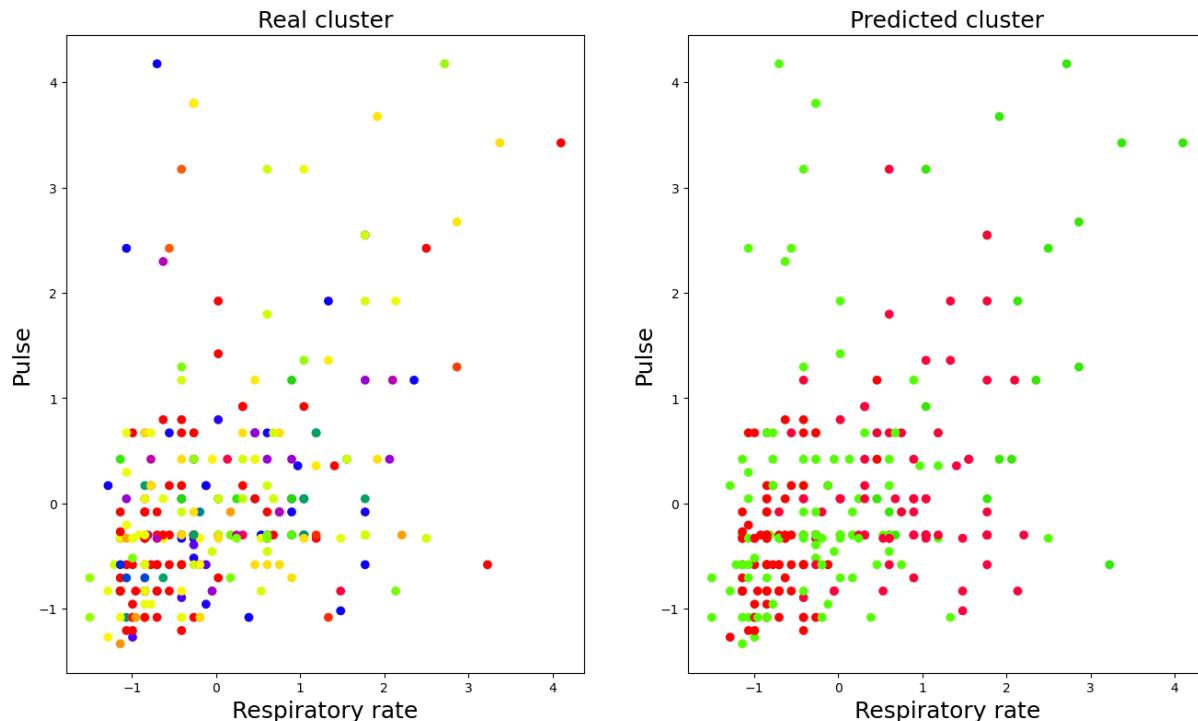
Визуализируем результаты для признаков pulse и respiratory_rate с найденным по методу Варда количеством кластеров:

In [18]:

```
fig, axes = plt.subplots(1, 2, figsize=(16,9))
axes[0].scatter(new_data.iloc[:,3], new_data.iloc[:,4], c=target.values, cmap = plt.cm.prism)
axes[0].set_title("Real cluster", fontsize = 18)
axes[0].set_ylabel("Pulse", fontsize = 18)
axes[0].set_xlabel("Respiratory rate", fontsize = 18)
axes[1].scatter(new_data.iloc[:,3], new_data.iloc[:,4], c=new_data.iloc[:, -1], cmap = plt.cm.prism)
axes[1].set_title("Predicted cluster", fontsize = 18)
axes[1].set_ylabel("Pulse", fontsize = 18)
axes[1].set_xlabel("Respiratory rate", fontsize = 18)
```

Out[18]:

Text(0.5, 0, 'Respiratory rate')



Оценим качество кластеризации:

In [19]:

```
print(f"Скорректированный индекс Рэнда: {np.round(adjusted_rand_score(target.values[:,0], new_data.iloc[:, -1]), 2)}%")
print(f"Коэффициент изменения информации: {np.round(adjusted_mutual_info_score(target.values[:,0], new_data.iloc[:, -1]), 2)}%")
print(f"Коэффициент качества кластеризации: {np.round(silhouette_score(new_data, new_data.iloc[:, -1]), 2)}%")
print(f"Качество однородности: {np.round(homogeneity_score(target.values[:,0], new_data.iloc[:, -1]), 2)}%")
print(f"Качество полноты: {np.round(completeness_score(target.values[:,0], new_data.iloc[:, -1]), 2)}%")
print(f"Метрика V-Measure: {np.round(v_measure_score(target.values[:,0], new_data.iloc[:, -1]), 2)}%")
```

Скорректированный индекс Рэнда: 11.99%

Коэффициент изменения информации: 15.4%

Коэффициент качества кластеризации: 20.43%

Качество однородности: 18.4%

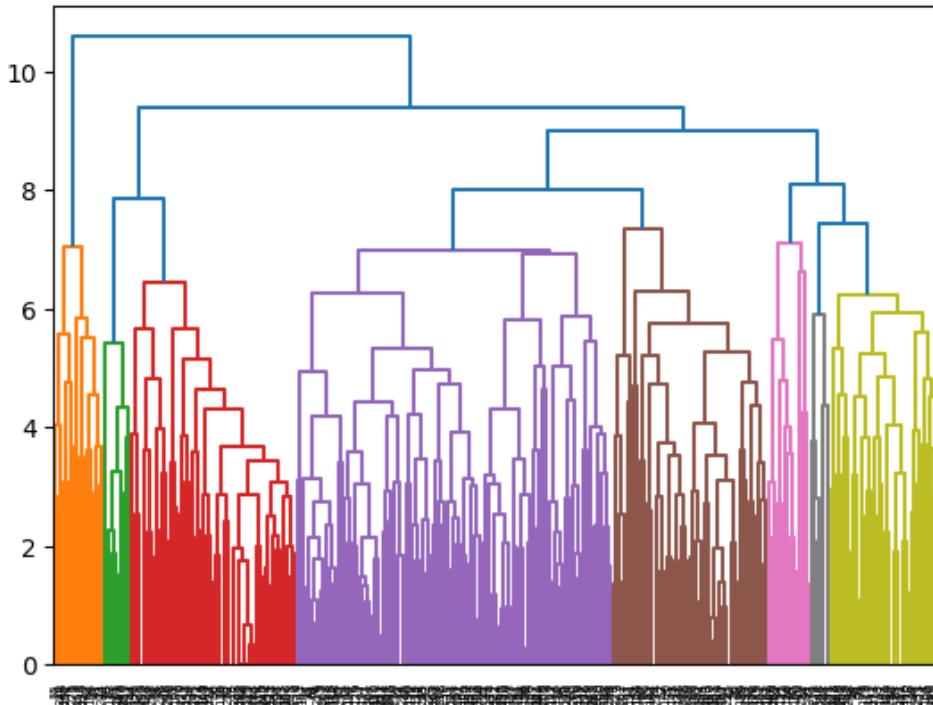
Качество полноты: 47.14%

Метрика V-Measure: 26.46%

Построим дендрограмму, где расстояние между кластерами вычисляется по методу полной связи, а расстояние между объектами по евклидовому.

In [20]:

```
new_data = data.copy()
link = linkage(new_data, 'complete', 'euclidean')
dn = dendrogram(link)
```



Информацию о кластерах добавляем в таблицу данных в виде столбца по критерию максимального количества кластеров равным 9:

In [21]:

```
new_data['cluster'] = fcluster(link, 9, criterion='maxclust')
new_data.groupby('cluster').mean()
```

Out[21]:

cluster	surgery	age	rectal_temperature	pulse	respiratory_rate	peripheral_pulse	mucous_membrane
1	-0.690595	3.391165		0.696116	2.242583	1.799023	0.243172
2	-0.583713	-0.294884		-2.187248	0.248444	0.094120	0.594038
3	-0.263303	-0.294884		-0.209309	0.186601	-0.154512	0.767937
4	-0.027573	-0.294884		-0.232361	-0.540770	-0.346571	-0.659658
5	0.649255	3.391165		0.388899	0.431446	-0.012631	-0.327625
6	0.966666	-0.294884		0.015984	-0.671949	-0.430550	-0.681471
7	-0.372810	-0.294884		1.470589	-0.077727	1.799418	0.141901
8	-0.518820	-0.294884		0.498160	0.826314	-0.110942	1.237463
9	-0.147877	-0.294884		0.456818	0.817467	0.264427	1.110564

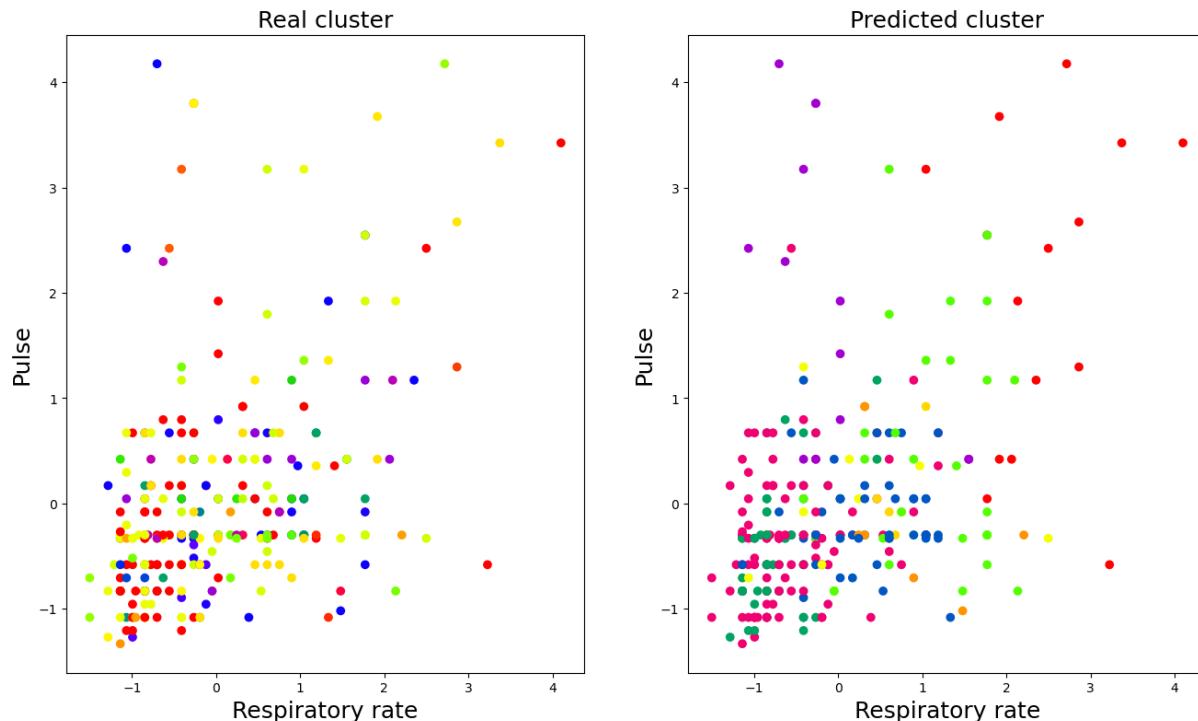
Визуализируем результаты для признаков pulse и respiratory_rate с найденным по методу полной связи количеством кластеров:

In [22]:

```
fig, axes = plt.subplots(1, 2, figsize=(16,9))
axes[0].scatter(new_data.iloc[:,3], new_data.iloc[:,4], c=target.values, cmap = plt.cm.prism)
axes[0].set_title("Real cluster", fontsize = 18)
axes[0].set_ylabel("Pulse", fontsize = 18)
axes[0].set_xlabel("Respiratory rate", fontsize = 18)
axes[1].scatter(new_data.iloc[:,3], new_data.iloc[:,4], c=new_data.iloc[:, -1], cmap = plt.cm.prism)
axes[1].set_title("Predicted cluster", fontsize = 18)
axes[1].set_ylabel("Pulse", fontsize = 18)
axes[1].set_xlabel("Respiratory rate", fontsize = 18)
```

Out[22]:

Text(0.5, 0, 'Respiratory rate')



Оценим качество кластеризации:

In [23]:

```
print(f"Скорректированный индекс Рэнда: {np.round(adjusted_rand_score(target.values[:,0], new_data.iloc[:,3]), 2)*100}%")
print(f"Коэффициент изменения информации: {np.round(adjusted_mutual_info_score(target.values[:,0], new_data.iloc[:,3]), 2)*100}%")
print(f"Коэффициент качества кластеризации: {np.round(silhouette_score(new_data, new_data.iloc[:, -1]), 2)*100}%")
print(f"Качество однородности: {np.round(homogeneity_score(target.values[:,0], new_data.iloc[:, -1]), 3)*100}%")
print(f"Качество полноты: {np.round(completeness_score(target.values[:,0], new_data.iloc[:, -1]), 2)*100}%")
print(f"Метрика V-Measure: {np.round(v_measure_score(target.values[:,0], new_data.iloc[:, -1]), 4)*100}%")
```

Скорректированный индекс Рэнда: 6.7%

Коэффициент изменения информации: 10.86%

Коэффициент качества кластеризации: 14.96%

Качество однородности: 22.8%

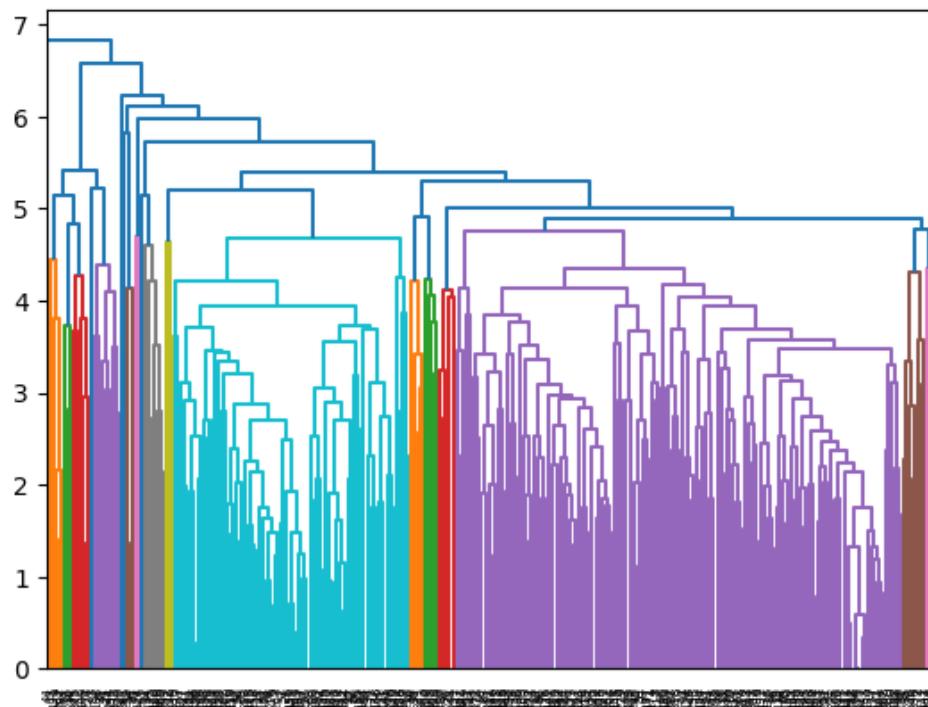
Качество полноты: 41.0%

Метрика V-Measure: 29.29%

Построим дендрограмму, где расстояние между кластерами вычисляется по методу средней связи, а расстояние между объектами по евклидовому.

In [24]:

```
new_data = data.copy()
link = linkage(new_data, 'average', 'euclidean')
dn = dendrogram(link)
```



Информацию о кластерах добавляем в таблицу данных в виде столбца по критерию максимального количества кластеров равным 15:

In [25]:

```
new_data['cluster'] = fcluster(link, 15, criterion='maxclust')
new_data.groupby('cluster').mean()
```

Out[25]:

cluster	surgery	age	rectal_temperature	pulse	respiratory_rate	peripheral_pulse	mucous_membrane	
1	1.233292	3.391165		0.528753	0.709932	0.101767	-0.452832	-0.589
2	-0.810838	3.391165		0.719111	1.424389	0.153206	0.472309	0.456
3	-0.583713	3.391165		0.447171	2.378077	2.798167	0.107122	-0.365
4	-0.810838	3.391165		1.415958	3.372175	3.423782	-1.110169	2.101
5	0.551915	-0.294884		3.506497	1.044531	0.545953	1.080954	-1.262
6	1.233292	-0.294884		2.792654	0.244403	-0.298627	1.080954	1.428
7	1.233292	-0.294884		-1.184469	-0.192031	0.295707	-0.014607	1.092
8	-0.518820	-0.294884		0.869650	-0.472595	3.012664	0.298410	-0.782
9	1.233292	-0.294884		0.039261	-0.555725	2.422798	-1.110169	-1.262
10	-0.810838	-0.294884		-1.082491	-1.161883	-0.830400	-0.744982	-0.814
11	0.952224	-0.294884		-0.023837	-0.705295	-0.421795	-0.836279	-0.665
12	0.211227	-0.294884		1.094729	1.226378	1.052701	1.080954	1.428
13	-0.419149	-0.294884		-0.214461	0.029017	-0.190549	0.306845	0.276
14	-0.810838	-0.294884		2.639688	0.026186	1.922306	-1.110169	1.428
15	-0.810838	-0.294884		-4.243794	2.499309	-0.329908	1.080954	0.755

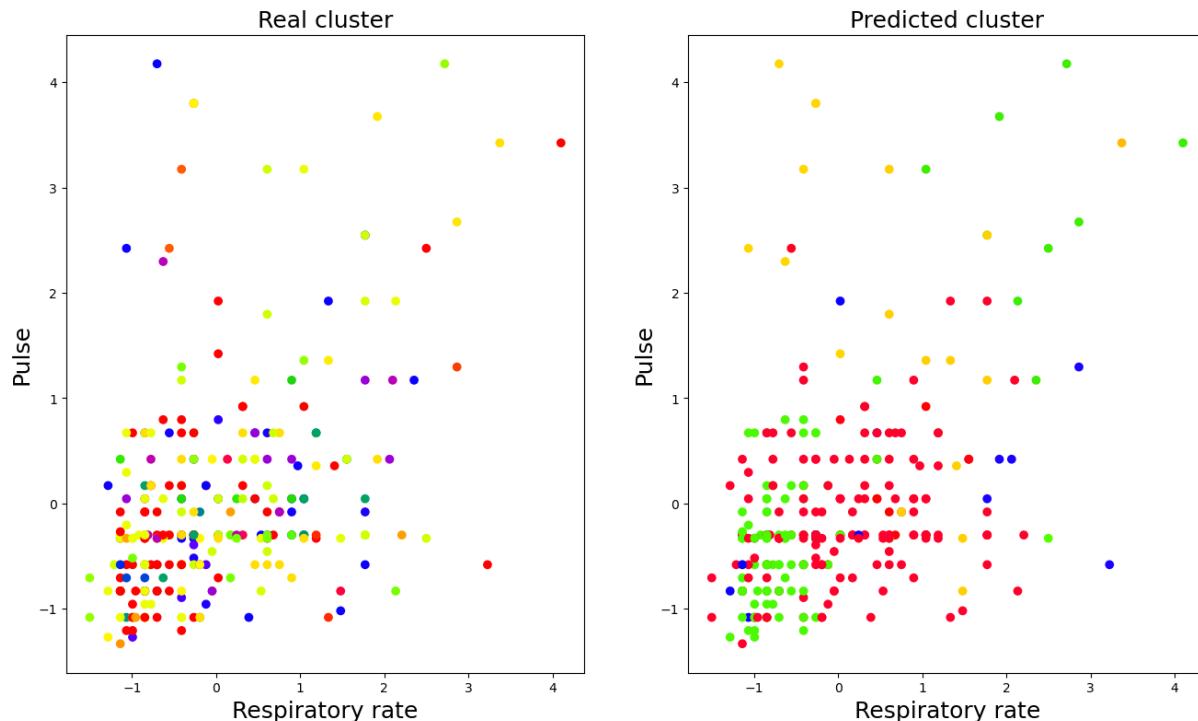
Визуализируем результаты для признаков pulse и respiratory_rate с найденным по методу средней связи количеством кластеров:

In [26]:

```
fig, axes = plt.subplots(1, 2, figsize=(16,9))
axes[0].scatter(new_data.iloc[:,3], new_data.iloc[:,4], c=target.values, cmap = plt.cm.prism)
axes[0].set_title("Real cluster", fontsize = 18)
axes[0].set_ylabel("Pulse", fontsize = 18)
axes[0].set_xlabel("Respiratory rate", fontsize = 18)
axes[1].scatter(new_data.iloc[:,3], new_data.iloc[:,4], c=new_data.iloc[:, -1], cmap = plt.cm.prism)
axes[1].set_title("Predicted cluster", fontsize = 18)
axes[1].set_ylabel("Pulse", fontsize = 18)
axes[1].set_xlabel("Respiratory rate", fontsize = 18)
```

Out[26]:

Text(0.5, 0, 'Respiratory rate')



Оценим качество кластеризации:

In [27]:

```
print(f"Скорректированный индекс Рэнда: {np.round(adjusted_rand_score(target.values[:,0], new_data.iloc[:,4]), 2)}%")
print(f"Коэффициент изменения информации: {np.round(adjusted_mutual_info_score(target.values[:,0], new_data.iloc[:,4]), 2)}%")
print(f"Коэффициент качества кластеризации: {np.round(silhouette_score(new_data, new_data.iloc[:, -1]), 2)}%")
print(f"Качество однородности: {np.round(homogeneity_score(target.values[:,0], new_data.iloc[:, -1]), 2)}%")
print(f"Качество полноты: {np.round(completeness_score(target.values[:,0], new_data.iloc[:, -1]), 2)}%")
print(f"Метрика V-Measure: {np.round(v_measure_score(target.values[:,0], new_data.iloc[:, -1]), 2)}%")
```

Скорректированный индекс Рэнда: 8.9%

Коэффициент изменения информации: 19.0%

Коэффициент качества кластеризации: 18.9%

Качество однородности: 24.9%

Качество полноты: 59.13%

Метрика V-Measure: 35.05%

Итерационные алгоритмы кластерного анализа

Посмотрим график для сравнения оптимального количества кластеров:

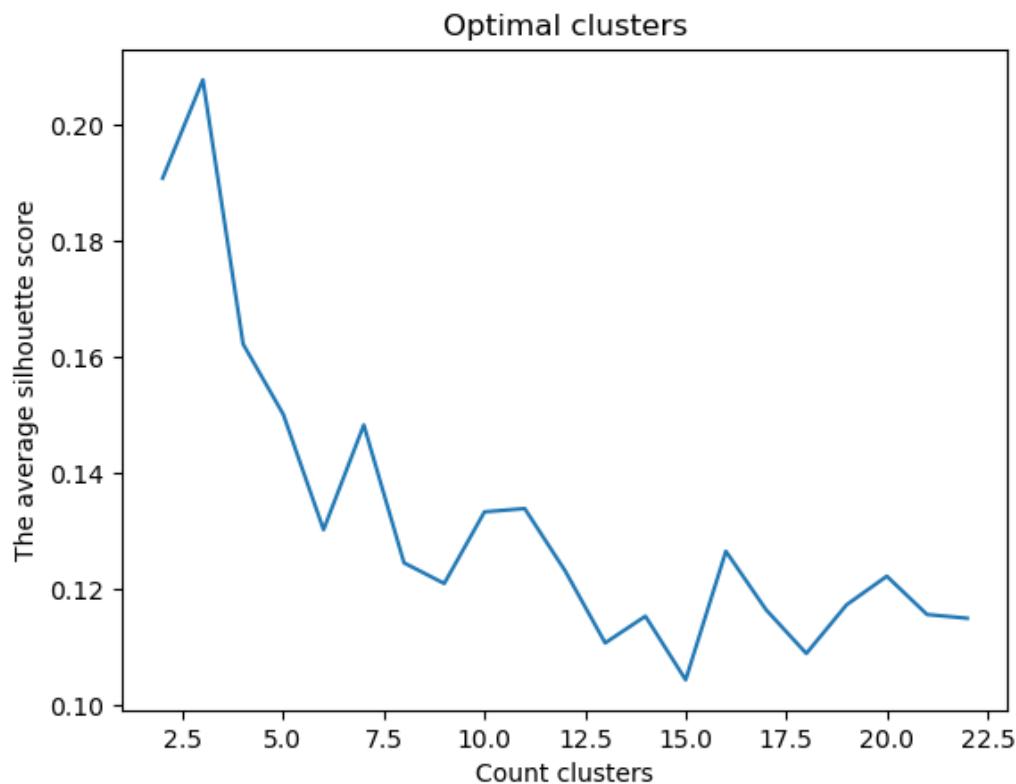
In [28]:

```
count_clusters = np.arange(2, 23)
silh_coeffs = []

for n_clusters in count_clusters:
    model = KMeans(n_clusters = n_clusters, n_init=10)
    cluster_labels = model.fit_predict(data)
    silh_coeffs.append(silhouette_score(data, cluster_labels))
```

In [29]:

```
plt.plot(count_clusters, silh_coeffs)
plt.ylabel("The average silhouette score")
plt.xlabel("Count clusters")
plt.title("Optimal clusters")
plt.show()
```



Можно сделать вывод, что наиболее оптимальное количество кластеров было подобрано методом Варда и Евклидовым расстоянием, однако коэффициент Рэнда и V-метрики говорят о наиболее оптимальном количестве кластеров равным 9, которые были найдены по методу полной связи.

Метод k-средних

Метод k-средних - это итеративный алгоритм кластеризации, основанный на минимизации суммарных квадратичных отклонений точек кластеров от центроидов (средних координат) этих кластеров.

Возьмем количество кластеров равное 3, как было найдено ранее.

In [30]:

```
new_data = data.copy()
```

Посмотрим оптимальное значение кластеров, используя метод elbow. Оценим количество кластеров для прогнозируемых данных от 1 до 30 и скопируем значения SSE в sse.

In [31]:

```
count_clusters = np.arange(1, 31)
sse = []
for n_clusters in count_clusters:
    model = KMeans(n_clusters=n_clusters, n_init=10)
    model.fit(new_data[["pulse", "respiratory_rate"]])
    sse.append(model.inertia_)
```

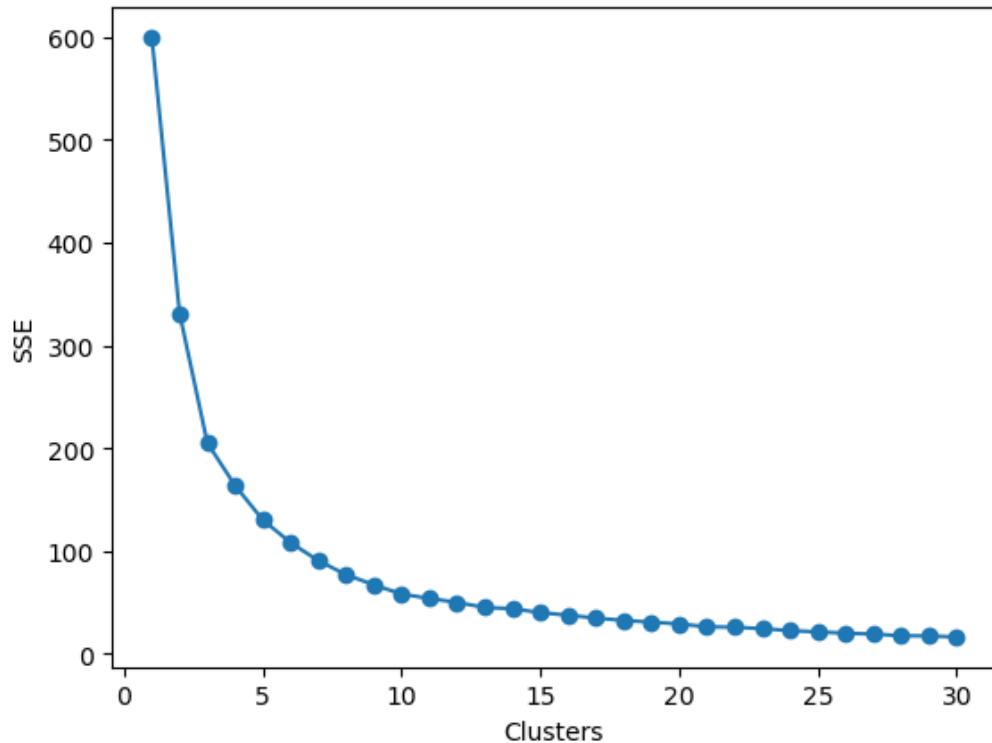
In [32]:

```
plt.xlabel('Clusters')
plt.ylabel('SSE')

plt.plot(count_clusters, sse)
plt.scatter(count_clusters, sse)
```

Out[32]:

```
<matplotlib.collections.PathCollection at 0x1ade5f71430>
```



Описываем модель и провдим моделирование для трех кластеров:

In [33]:

```
model = KMeans(n_clusters=3, n_init = 10)
model.fit(new_data)
```

Out[33]:

```
▼      KMeans
KMeans(n_clusters=3, n_init=10)
```

Делаем прогноз:

In [34]:

```
prediction = model.predict(new_data)
prediction[:10]
```

Out[34]:

```
array([2, 2, 1, 0, 2, 1, 2, 2, 2, 0])
```

Подготавливаем данные для графика:

In [35]:

```
new_data["cluster"] = prediction
```

In [36]:

```
data0=new_data[new_data.cluster==0]
data1=new_data[new_data.cluster==1]
data2=new_data[new_data.cluster==2]
```

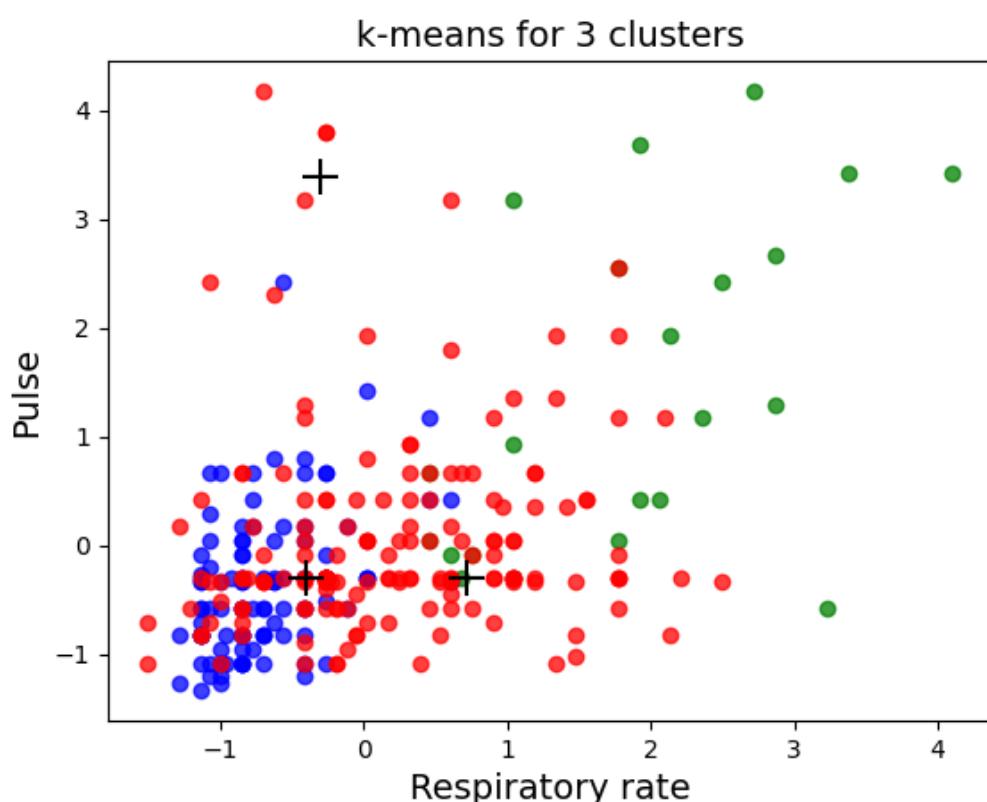
Строим график для признаков пульса и частотой дыхания:

In [37]:

```
plt.scatter(data0[" pulse"],data0[" respiratory_rate"],color='green', alpha=0.75)
plt.scatter(data1[" pulse"],data1[" respiratory_rate"],color='blue', alpha=0.75)
plt.scatter(data2[" pulse"],data2[" respiratory_rate"],color='red', alpha=0.75)
plt.scatter(model.cluster_centers_[:,0],model.cluster_centers_[:,1], s = 200, color='black',marker='+')
plt.title("k-means for 3 clusters", fontsize = 14)
plt.ylabel("Pulse", fontsize = 14)
plt.xlabel("Respiratory rate", fontsize = 14)
```

Out[37]:

```
Text(0.5, 0, 'Respiratory rate')
```



Оценим качество кластеризации:

In [38]:

```
print(f"Скорректированный индекс Рэнда: {np.round(adjusted_rand_score(target.values[:,0], new_data.iloc[:, -1]), 4)}%")
print(f"Коэффициент изменения информации: {np.round(adjusted_mutual_info_score(target.values[:,0], new_data.iloc[:, -1]), 4)}%")
print(f"Коэффициент качества кластеризации: {np.round(silhouette_score(new_data, new_data.iloc[:, -1]), 4)}%")
print(f"Качество однородности: {np.round(homogeneity_score(target.values[:,0], new_data.iloc[:, -1]), 4)}%")
print(f"Качество полноты: {np.round(completeness_score(target.values[:,0], new_data.iloc[:, -1]), 4)*100}%")
print(f"Метрика V-Measure: {np.round(v_measure_score(target.values[:,0], new_data.iloc[:, -1]), 4)*100}%"
```

```
Скорректированный индекс Рэнда: 8.5%
Коэффициент изменения информации: 14.09%
Коэффициент качества кластеризации: 22.5%
Качество однородности: 14.17%
Качество полноты: 52.04%
Метрика V-Measure: 22.27%
```

Количество кластеров равное 3 является оптимальным на первый взгляд, но методы оценки кластеризации говорят об обратном, поэтому для чистоты эксперимента имеет смысл проверить количество кластеров равное 6.

In [39]:

```
new_data = data.copy()
model = KMeans(n_clusters=6, n_init = 10)
model.fit(new_data)
prediction = model.predict(new_data)
```

In [40]:

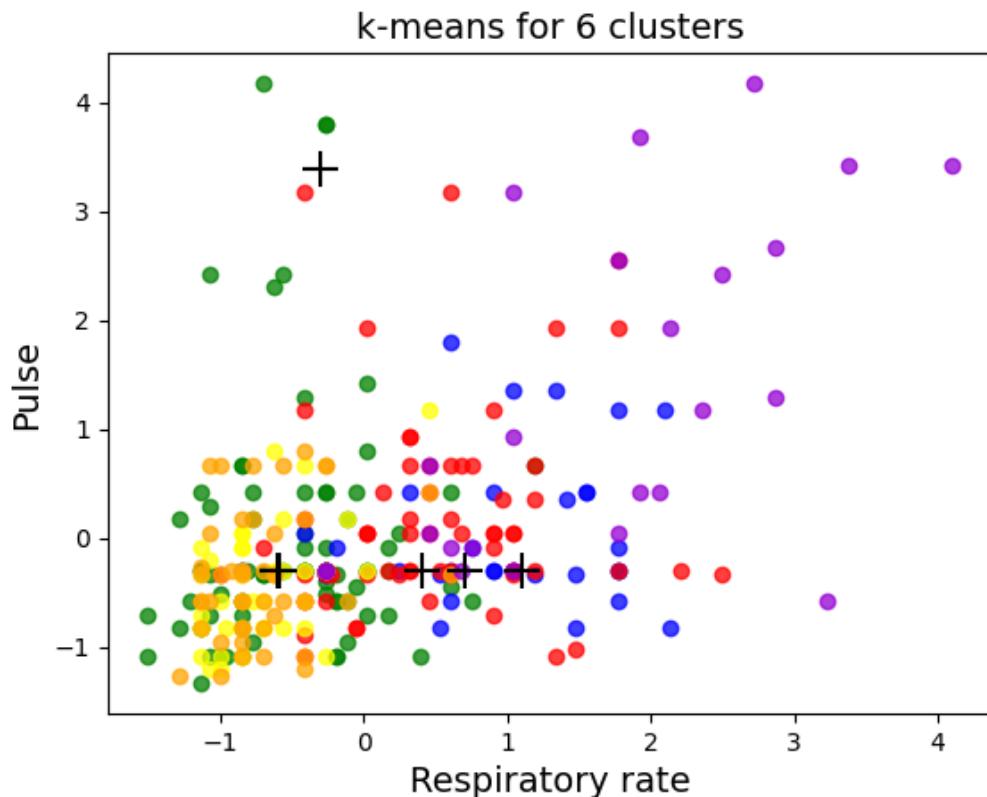
```
new_data["cluster"] = prediction

data0=new_data[new_data.cluster==0]
data1=new_data[new_data.cluster==1]
data2=new_data[new_data.cluster==2]
data3=new_data[new_data.cluster==3]
data4=new_data[new_data.cluster==4]
data5=new_data[new_data.cluster==5]

plt.scatter(data0[" pulse"],data0[" respiratory_rate"],color='green', alpha=0.75)
plt.scatter(data1[" pulse"],data1[" respiratory_rate"],color='blue', alpha=0.75)
plt.scatter(data2[" pulse"],data2[" respiratory_rate"],color='red', alpha=0.75)
plt.scatter(data3[" pulse"],data3[" respiratory_rate"],color='yellow', alpha=0.75)
plt.scatter(data4[" pulse"],data4[" respiratory_rate"],color='orange', alpha=0.75)
plt.scatter(data5[" pulse"],data5[" respiratory_rate"],color='darkviolet', alpha=0.75)
plt.scatter(model.cluster_centers_[:,0],model.cluster_centers_[:,1], s = 200, color='black',marker='+')
plt.title("k-means for 6 clusters", fontsize = 14)
plt.ylabel("Pulse", fontsize = 14)
plt.xlabel("Respiratory rate", fontsize = 14)
```

Out[40]:

Text(0.5, 0, 'Respiratory rate')



Оценим качество кластеризации:

In [41]:

```
print(f"Скорректированный индекс Рэнда: {np.round(adjusted_rand_score(target.values[:,0], new_data.iloc[:,0]), 3)}%")
print(f"Коэффициент изменения информации: {np.round(adjusted_mutual_info_score(target.values[:,0], new_data.iloc[:,0]), 3)}%")
print(f"Коэффициент качества кластеризации: {np.round(silhouette_score(new_data, new_data.iloc[:, -1]), 3)}%")
print(f"Качество однородности: {np.round(homogeneity_score(target.values[:,0], new_data.iloc[:, -1]), 3)}%")
print(f"Качество полноты: {np.round(completeness_score(target.values[:,0], new_data.iloc[:, -1]), 3)*100}%")
print(f"Метрика V-Measure: {np.round(v_measure_score(target.values[:,0], new_data.iloc[:, -1]), 3)*100}%")
```

Скорректированный индекс Рэнда: 9.700000000000001%
Коэффициент изменения информации: 16.0%
Коэффициент качества кластеризации: 18.4%
Качество однородности: 23.599999999999998%
Качество полноты: 45.11%
Метрика V-Measure: 31.0%

Fuzzy C-Means алгоритм кластеризации

The Algorithm **Fuzzy c-means (FCM)** - это метод кластеризации, который позволяет одному фрагменту данных принадлежать двум или более кластерам.

In [42]:

```
new_data = data.copy()
```

Обучим модель нечеткой кластеризации с количеством кластеров равным 6:

In [43]:

```
model = FCM(n_clusters = 6)
model.fit(new_data.values)
center = model.centers
```

Спрогнозируем значения:

In [44]:

```
prediction = model.predict(data.values)
prediction[:10]
```

Out[44]:

```
array([4, 4, 0, 4, 4, 0, 5, 4, 4, 0], dtype=int64)
```

In [45]:

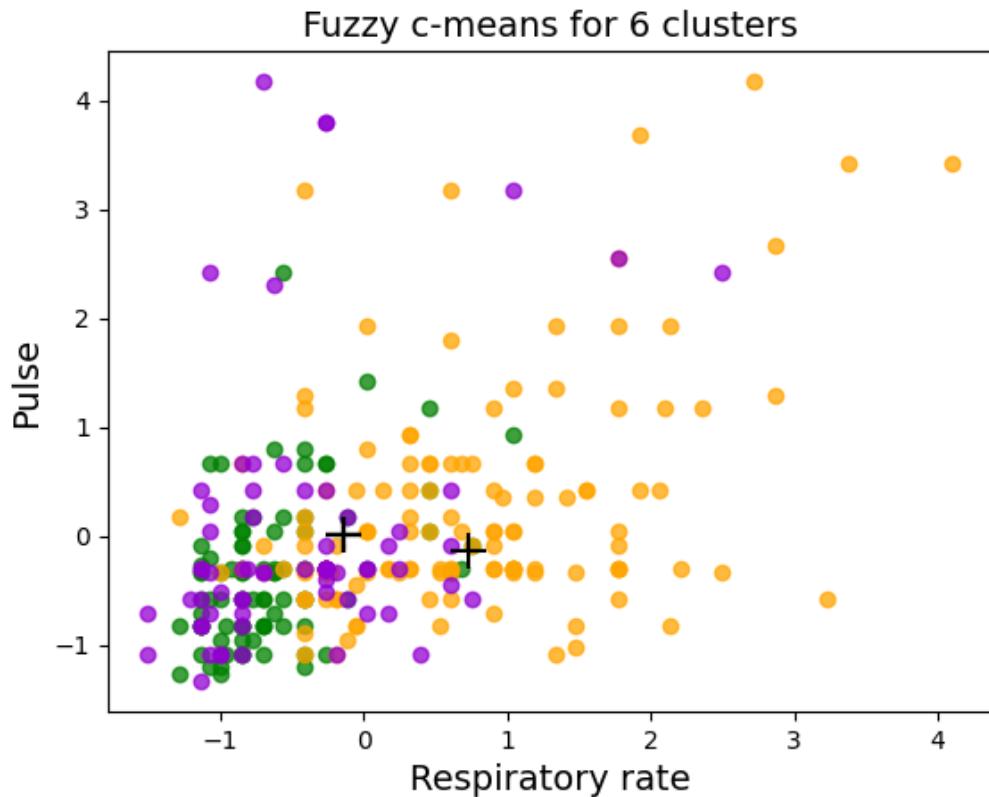
```
new_data["cluster"] = prediction

data0=new_data[new_data.cluster==0]
data1=new_data[new_data.cluster==1]
data2=new_data[new_data.cluster==2]
data3=new_data[new_data.cluster==3]
data4=new_data[new_data.cluster==4]
data5=new_data[new_data.cluster==5]

plt.scatter(data0[" pulse"],data0[" respiratory_rate"],color='green', alpha=0.75)
plt.scatter(data1[" pulse"],data1[" respiratory_rate"],color='blue', alpha=0.75)
plt.scatter(data2[" pulse"],data2[" respiratory_rate"],color='red', alpha=0.75)
plt.scatter(data3[" pulse"],data3[" respiratory_rate"],color='yellow', alpha=0.75)
plt.scatter(data4[" pulse"],data4[" respiratory_rate"],color='orange', alpha=0.75)
plt.scatter(data5[" pulse"],data5[" respiratory_rate"],color='darkviolet', alpha=0.75)
plt.scatter(center[:,0],center[:,1], s = 200, color='black',marker='+')
plt.title("Fuzzy c-means for 6 clusters", fontsize = 14)
plt.ylabel("Pulse", fontsize = 14)
plt.xlabel("Respiratory rate", fontsize = 14)
```

Out[45]:

Text(0.5, 0, 'Respiratory rate')



Оценим качество кластеризации:

In [46]:

```
print(f"Скорректированный индекс Рэнда: {np.round(adjusted_rand_score(target.values[:,0], new_data.iloc[:,0]), 2)}%")
print(f"Коэффициент изменения информации: {np.round(adjusted_mutual_info_score(target.values[:,0], new_data.iloc[:,0]), 2)}%")
print(f"Коэффициент качества кластеризации: {np.round(silhouette_score(new_data, new_data.iloc[:, -1]), 2)}%")
print(f"Качество однородности: {np.round(homogeneity_score(target.values[:,0], new_data.iloc[:, -1]), 2)}%")
print(f"Качество полноты: {np.round(completeness_score(target.values[:,0], new_data.iloc[:, -1]), 2)}%")
print(f"Метрика V-Measure: {np.round(v_measure_score(target.values[:,0], new_data.iloc[:, -1]), 2)}%")
```

Скорректированный индекс Рэнда: 12.7%
Коэффициент изменения информации: 16.38%
Коэффициент качества кластеризации: 21.8%
Качество однородности: 16.66%
Качество полноты: 51.41%
Метрика V-Measure: 25.16%

Метод кластеризации на основе плотности DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise, плотностной алгоритм пространственной кластеризации с присутствием шума) – популярный алгоритм кластеризации, используемый в анализе данных в качестве одной из замен метода k-средних.

Метод не требует предварительных предположений о числе кластеров, но нужно настроить два других параметра: `eps` и `min_samples`. Данные параметры – это соответственно максимальное расстояние между соседними точками и минимальное число точек в окрестности (количество соседей), когда можно говорить, что эти экземпляры данных образуют один кластер.

In [47]:

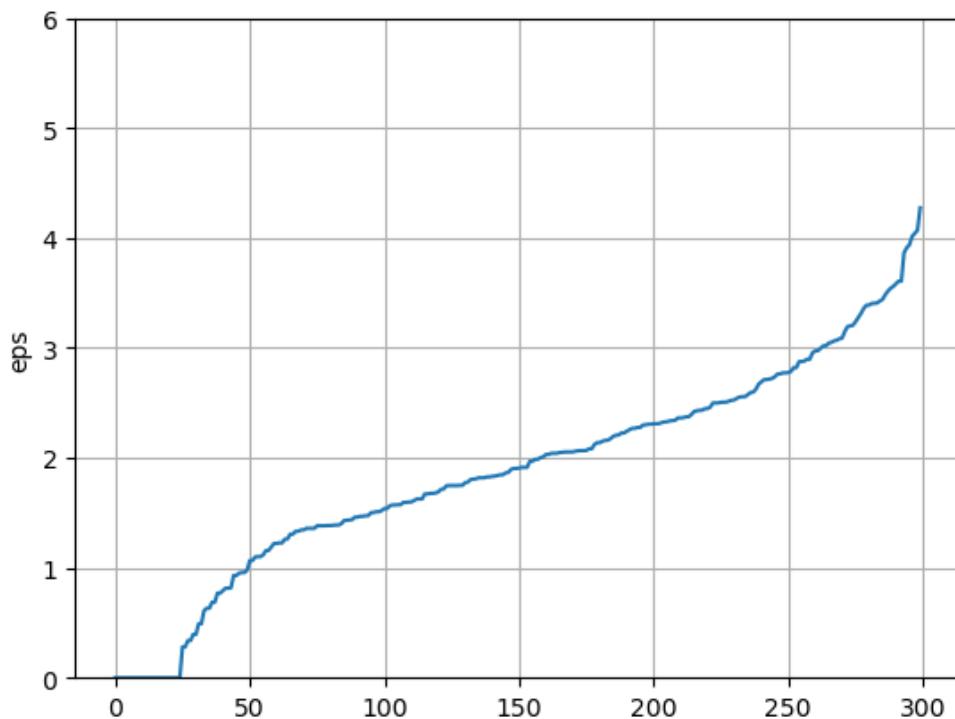
```
new_data = data.copy()
```

Исследуем данные для оптимального определения значения эпсилон для дальнейшей кластеризации:

In [48]:

```
from sklearn.neighbors import NearestNeighbors
neighbors = NearestNeighbors(n_neighbors = 3)
neighbors_fit = neighbors.fit(data)
distances, indices = neighbors_fit.kneighbors(data)

distances = np.sort(distances, axis = 0)
distances = distances[:,1]
plt.ylim(0, 6)
plt.plot(distances)
plt.ylabel('eps')
plt.grid()
```



Обучим модель:

In [49]:

```
model = DBSCAN(eps = 2.7, min_samples = 3)
model.fit_predict(new_data)[:10]
```

Out[49]:

```
array([ 0, -1,  0, -1, -1,  0,  0,  0,  1], dtype=int64)
```

Визуальный анализ и сравнение результатов кластеризации:

In [50]:

```
final_data = target.copy()
```

In [51]:

```
DBSCAN_method = pd.DataFrame(model.labels_)

final_data["DBSCAN"] = DBSCAN_method
final_data.head(5)
```

Out[51]:

	first_type_of_lesion	DBSCAN
0	56.0	0
1	14.0	-1
2	1.0	0
3	14.0	-1
4	37.0	-1

In [52]:

```
db = model.fit(new_data)
labels = db.labels_
```

In [53]:

```
comp = PCA(n_components = 2).fit_transform(new_data)
dff = pd.DataFrame(comp, columns = ["pulse", "respiratory_rate"])
final_data = final_data.join(dff)
final_data.head(5)
```

Out[53]:

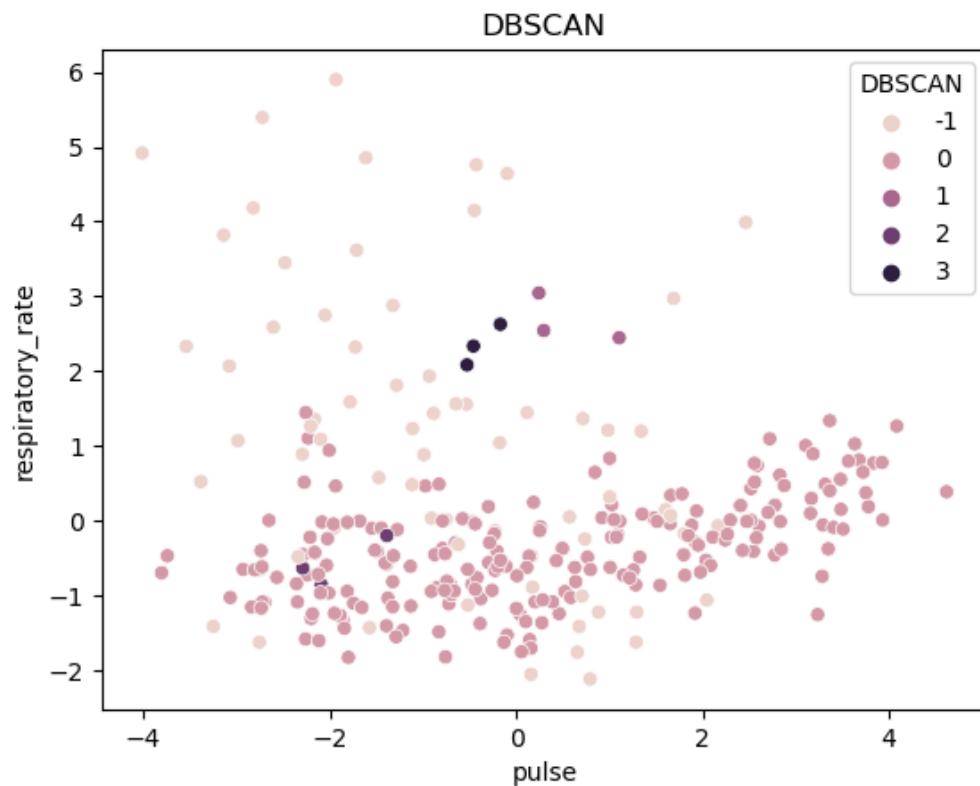
	first_type_of_lesion	DBSCAN	pulse	respiratory_rate
0	56.0	0	-1.135175	-0.612257
1	14.0	-1	-0.909673	0.030850
2	1.0	0	3.315544	0.484955
3	14.0	-1	-4.014631	4.916979
4	37.0	-1	-0.467603	-0.007187

In [54]:

```
scatterplot(x = final_data["pulse"], y = final_data["respiratory_rate"], hue = final_data["DBSCAN"], da
title("DBSCAN")
```

Out[54]:

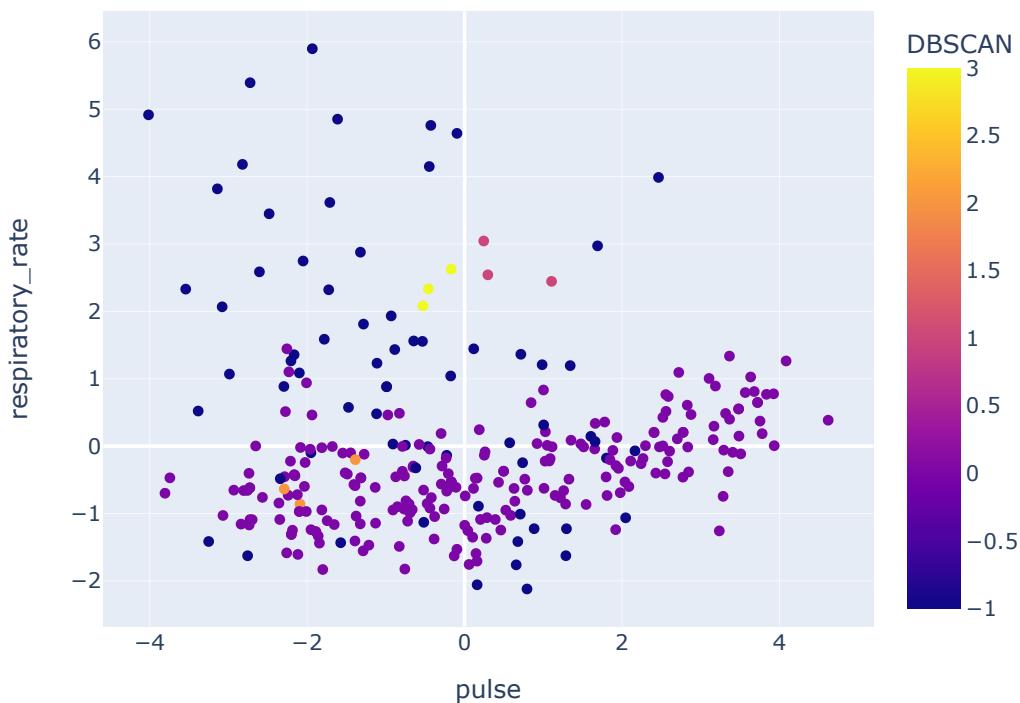
```
Text(0.5, 1.0, 'DBSCAN')
```



In [55]:

```
px.scatter(final_data, x = "pulse", y = "respiratory_rate", color = "DBSCAN", title = 'DBSCAN Clustering')
```

DBSCAN Clustering



Оценим кластеризацию:

In [58]:

```
if -1 in labels else 0)))
l)))
round(adjusted_rand_score(final_data[" first_type_of_lesion"], final_data["DBSCAN"]),2)*100) + "%")
round(adjusted_mutual_info_score(final_data[" first_type_of_lesion"], final_data["DBSCAN"]),4)*100) + "%")
round(silhouette_score(new_data, new_data.iloc[:, -1]),4)*100}%"')
eneity_score(final_data[" first_type_of_lesion"], final_data["DBSCAN"]),4)*100) + "%")
ss_score(final_data[" first_type_of_lesion"], final_data["DBSCAN"]),4)*100) + "%")
_l_score(final_data[" first_type_of_lesion"], final_data["DBSCAN"]),4)*100) + "%")
```

Число кластеров: 4

Количество шума: 69

Скорректированный индекс Рэнда: 3.0%

Коэффициент изменения информации: 5.54%

Коэффициент качества кластеризации: 16.0%

Качество однородности: 8.7%

Качество полноты: 40.75%

Метрика V-Measure: 14.34%

t-SNE метод понижения размерности

Метод t-SNE (**t-distributed stochastic neighbor embedding**) представляет собой один из методов обучения без учителя, используемых для визуализации, например, отображения пространства высокой размерности в двух- или трехмерное пространство. t-SNE расшифровывается как распределенное стохастическое соседнее вложение.

Метод моделирует каждый объект пространства высокой размерности в двух- или трехкоординатную точку таким образом, что близкие по характеристикам элементы данных в многомерном пространстве (например, датасете с большим числом столбцов) проецируются в соседние точки, а разнородные объекты с большей вероятностью моделируются точками, далеко отстоящими друг от друга.

In [59]:

```
new_data = data.copy()
```

Определяем модель и задаем скорость обучения:

In [60]:

```
model = TSNE(perplexity=10, learning_rate=200, n_iter=5000)
transformed = model.fit_transform(new_data)
```

Визуализируем данные:

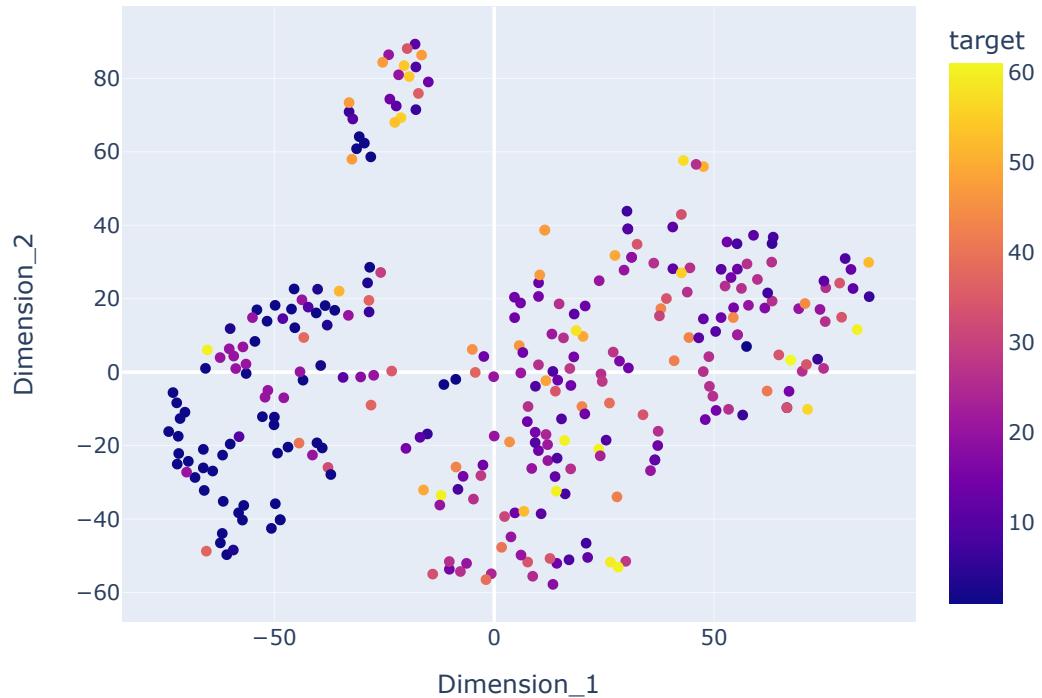
In [61]:

```
umap_data = pd.DataFrame(transformed, columns = ["Dimension_1", "Dimension_2"])
umap_data["target"] = target
```

In [62]:

```
px.scatter(umap_data, x = "Dimension_1", y = "Dimension_2", color = "target", title = 't-SNE')
```

t-SNE



Получаем параметры модели:

In [63]:

```
print(f"Расхождение Кулбека-Лейблера после оптимизации: {np.round(model.kl_divergence_, 3)*100}%)")
```

Расхождение Кулбека-Лейблера после оптимизации: 83.7%

Uniform Manifold Approximation and Projection (UMAP)

Uniform Manifold Approximation and Projection (UMAP) — алгоритм машинного обучения, выполняющий нелинейное снижение размерности.

In [64]:

```
new_data = data.copy()
```

Обучаем модель:

In [65]:

```
model = UMAP(random_state=3)
transformed = model.fit_transform(new_data, target)
```

Трансформируем данные:

In [66]:

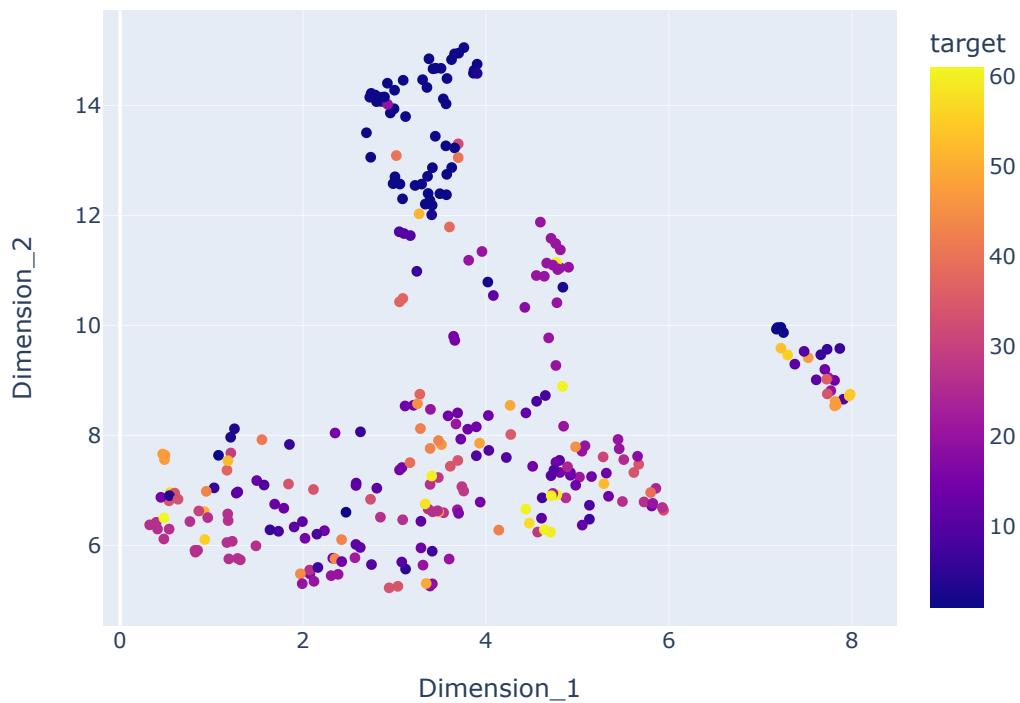
```
umap_data = pd.DataFrame(transformed, columns = ["Dimension_1", "Dimension_2"])
umap_data["target"] = target
```

Визуализируем результат:

In [67]:

```
px.scatter(umap_data, x = "Dimension_1", y = "Dimension_2", color = "target", title = 'UMAP')
```

UMAP



Повторный анализ

Анализ полученных данных:

На основе данных со сниженной размерностью, проведем повторный кластерный анализ на основе итерационного алгоритма с нечеткими центроидами fuzzy c-means.

Разделим данные и удалим информацию о целевой переменной из трансформированных данных:

In [78]:

```
new_data = umap_data.drop(umap_data.columns[-1], axis=1)
predicted_data = new_data.copy()
```

Fuzzy c-means

Построим модель нечетких центроидов с количеством кластеров равным 11:

In [69]:

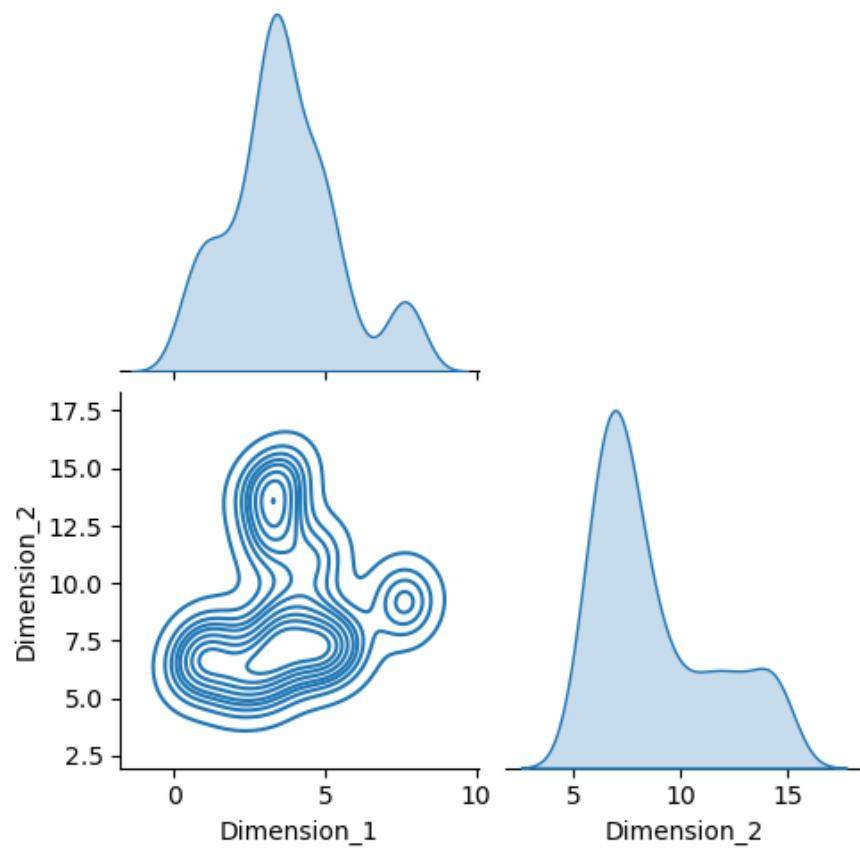
```
model = FCM(n_clusters = 11)
model.fit(new_data.values)
center = model.centers
```

In [70]:

```
sns.pairplot(new_data, kind = 'kde', corner = True)
```

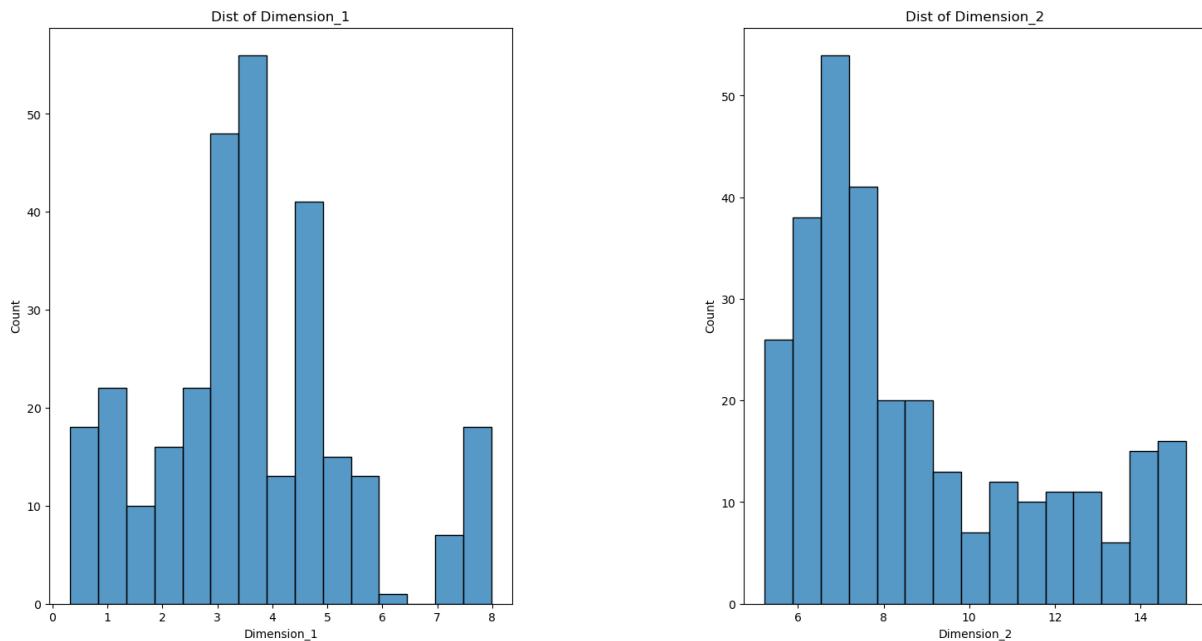
Out[70]:

```
<seaborn.axisgrid.PairGrid at 0x1adf00238e0>
```



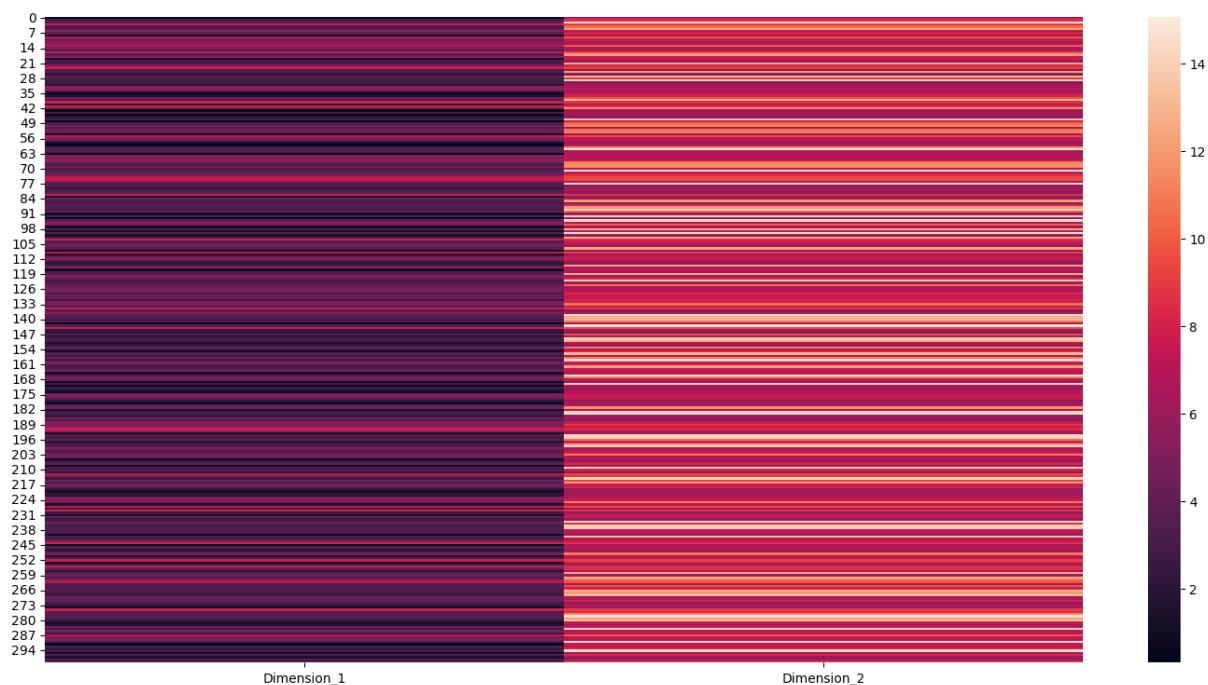
In [71]:

```
plt.figure(1 , figsize = (18, 9))
n = 0
for x in new_data.columns:
    n += 1
    plt.subplot(1 , 2 , n)
    plt.subplots_adjust(hspace = 2, wspace = 0.5)
    sns.histplot(new_data[x] , bins = 15)
    plt.title('Dist of {}'.format(x))
plt.show()
```



In [72]:

```
plt.figure(1, figsize = (18 , 9))
sns.heatmap(new_data)
plt.show()
```



Оценим данные:

In [73]:

```
prediction = model.predict(predicted_data.values)
new_data["cluster"] = prediction
new_data.groupby("cluster").mean()
```

Out[73]:

Dimension_1 Dimension_2

cluster	Dimension_1	Dimension_2
0	7.616313	9.212461
1	3.381557	6.718484
2	5.130397	7.117776
3	1.332567	7.397972
4	0.844942	6.344403
5	4.678091	11.011384
6	3.331737	12.537333
7	2.492476	5.738811
8	3.273615	14.372331
9	3.575795	10.493878
10	3.782455	8.189824

Разделим значения для точечного анализа:

In [74]:

```
data0=new_data[new_data.cluster==0]
data1=new_data[new_data.cluster==1]
data2=new_data[new_data.cluster==2]
data3=new_data[new_data.cluster==3]
data4=new_data[new_data.cluster==4]
data5=new_data[new_data.cluster==5]
data6=new_data[new_data.cluster==6]
data7=new_data[new_data.cluster==7]
data8=new_data[new_data.cluster==8]
data9=new_data[new_data.cluster==9]
data10=new_data[new_data.cluster==10]
```

Построим точечный график для полученных кластеров:

In [75]:

```
fig, axes = plt.subplots(1, 2, figsize=(20,16))

axes[0].scatter(new_data.iloc[:,0], new_data.iloc[:,1], c=target.values, cmap = plt.cm.prism)
axes[0].set_title("Real cluster", fontsize = 18)

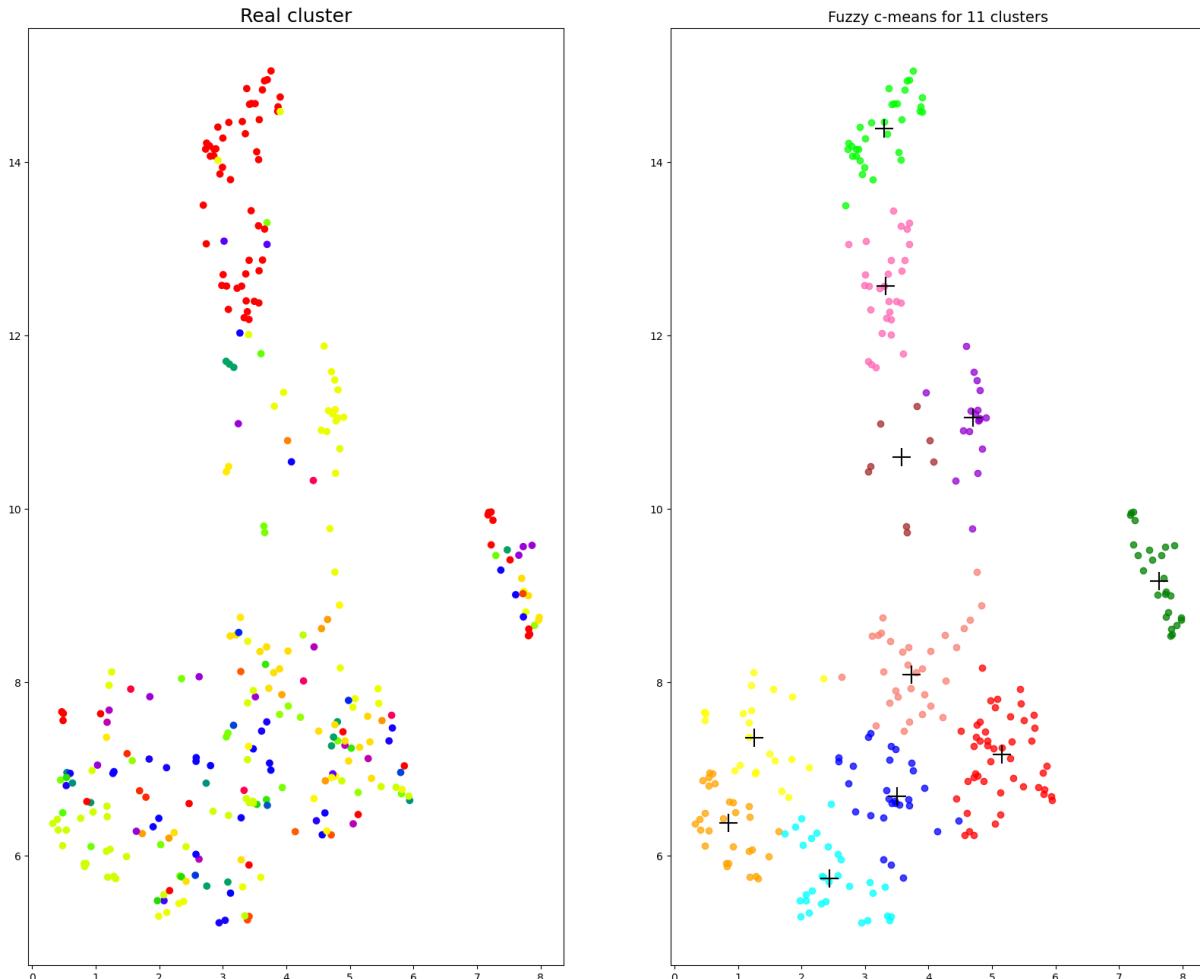
axes[1].scatter(data0["Dimension_1"],data0["Dimension_2"],color='green', alpha=0.75)
axes[1].scatter(data1["Dimension_1"],data1["Dimension_2"],color='blue', alpha=0.75)
axes[1].scatter(data2["Dimension_1"],data2["Dimension_2"],color='red', alpha=0.75)
axes[1].scatter(data3["Dimension_1"],data3["Dimension_2"],color='yellow', alpha=0.75)
axes[1].scatter(data4["Dimension_1"],data4["Dimension_2"],color='orange', alpha=0.75)
axes[1].scatter(data5["Dimension_1"],data5["Dimension_2"],color='darkviolet', alpha=0.75)
axes[1].scatter(data6["Dimension_1"],data6["Dimension_2"],color='hotpink', alpha=0.75)
axes[1].scatter(data7["Dimension_1"],data7["Dimension_2"],color='aqua', alpha=0.75)
axes[1].scatter(data8["Dimension_1"],data8["Dimension_2"],color='lime', alpha=0.75)
axes[1].scatter(data9["Dimension_1"],data9["Dimension_2"],color='brown', alpha=0.75)
axes[1].scatter(data10["Dimension_1"],data10["Dimension_2"],color='salmon', alpha=0.75)
axes[1].set_title("Predicted cluster", fontsize = 18)

plt.scatter(center[:,0],center[:,1], s = 300, color='black',marker='+')

plt.title("Fuzzy c-means for 11 clusters", fontsize = 14)
```

Out[75]:

Text(0.5, 1.0, 'Fuzzy c-means for 11 clusters')



Оценим кластеризацию:

In [77]:

```
декс Рэнда: {np.round(adjusted_rand_score(target.values[:,0], new_data.iloc[:, -1]), 4)*100}%"}
я информации: {np.round(adjusted_mutual_info_score(target.values[:,0], new_data.iloc[:, -1]), 3)*100}%"}
кластеризации: {np.round(silhouette_score(new_data, new_data.iloc[:, -1]), 2)*100}%"}
и: {np.round(homogeneity_score(target.values[:,0], new_data.iloc[:, -1]), 4)*100}%"}
p.round(completeness_score(target.values[:,0], new_data.iloc[:, -1]), 2)*100}%"}
np.round(v_measure_score(target.values[:,0], new_data.iloc[:, -1]), 2)*100}%"}
```

◀

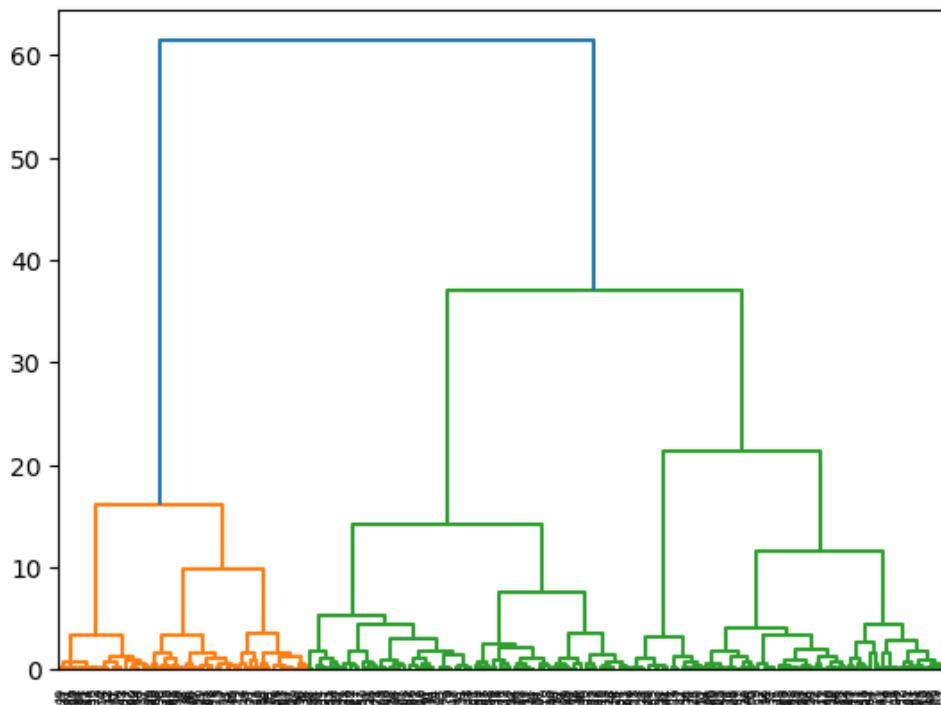
▶

Скорректированный индекс Рэнда: 22.09%
Коэффициент изменения информации: 26.8%
Коэффициент качества кластеризации: 68.0%
Качество однородности: 38.79%
Качество полноты: 54.0%
Метрика V-Measure: 45.0%

Hierarchic algorithm by method Ward

In [79]:

```
link = linkage(new_data, 'ward', 'euclidean')
dn = dendrogram(link)
```



In [80]:

```
new_data["cluster"] = fcluster(link, 2, criterion='maxclust')
new_data.groupby('cluster').mean()
```

Out[80]:

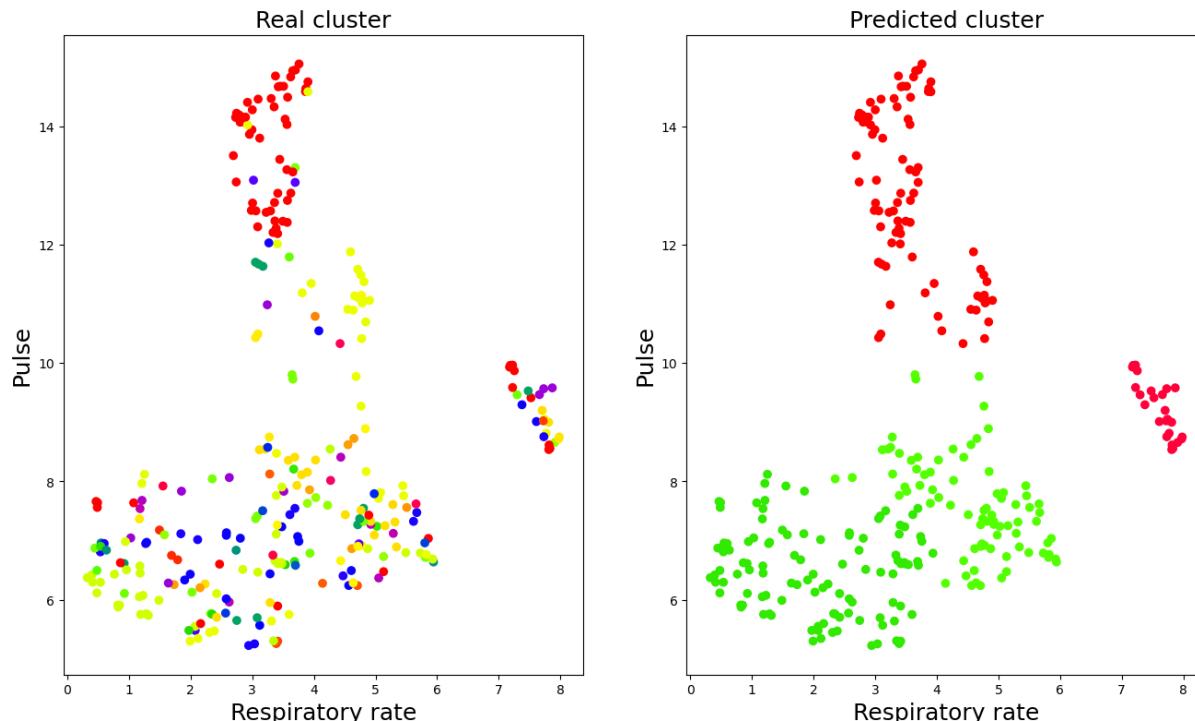
	Dimension_1	Dimension_2
cluster		
1	3.597588	12.813692
2	2.047979	6.495684
3	7.616313	9.212461
4	4.578617	7.604463

In [83]:

```
fig, axes = plt.subplots(1, 2, figsize=(16,9))
axes[0].scatter(new_data.iloc[:,0], new_data.iloc[:,1], c=target.values, cmap = plt.cm.prism)
axes[0].set_title("Real cluster", fontsize = 18)
axes[0].set_ylabel("Pulse", fontsize = 18)
axes[0].set_xlabel("Respiratory rate", fontsize = 18)
axes[1].scatter(new_data.iloc[:,0], new_data.iloc[:,1], c=new_data.iloc[:, -1], cmap = plt.cm.prism)
axes[1].set_title("Predicted cluster", fontsize = 18)
axes[1].set_ylabel("Pulse", fontsize = 18)
axes[1].set_xlabel("Respiratory rate", fontsize = 18)
```

Out[83]:

Text(0.5, 0, 'Respiratory rate')



In [88]:

```
print(f"Скорректированный индекс Рэнда: {np.round(adjusted_rand_score(target.values[:,0], new_data.iloc[:,0]), 2)}%")
print(f"Коэффициент изменения информации: {np.round(adjusted_mutual_info_score(target.values[:,0], new_data.iloc[:,0]), 2)}%")
print(f"Коэффициент качества кластеризации: {np.round(silhouette_score(new_data, new_data.iloc[:, -1]), 2)}%")
print(f"Качество однородности: {np.round(homogeneity_score(target.values[:,0], new_data.iloc[:, -1]), 2)}%")
print(f"Качество полноты: {np.round(completeness_score(target.values[:,0], new_data.iloc[:, -1]), 2)*100}%")
print(f"Метрика V-Measure: {np.round(v_measure_score(target.values[:,0], new_data.iloc[:, -1]), 1)*100}%")
```

Скорректированный индекс Рэнда: 15.0%

Коэффициент изменения информации: 18.6%

Коэффициент качества кластеризации: 60.0%

Качество однородности: 20.4%

Качество полноты: 52.0%

Метрика V-Measure: 30.0%

После изменения размерности данных и подбора наиболее эффективного алгоритма обучения без учителя, которым оказался нечеткий анализ к-средних, были получены кластеры, которые на фоне остальных методов описываются лучшим образом, позволяя по определенному набору предикторов правильно категоризировать тип заболевания лошади (целевая переменная).