```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.cluster.hierarchy as sch
from sklearn.metrics import silhouette_score, adjusted_rand_score,
v_measure_score, homogeneity_score, completeness_score,
adjusted_mutual_info_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 sklearn.neighbors import NearestNeighbors
from fcmeans import FCM
from umap import UMAP

%matplotlib inline
```

Подготовка к анализу

Получаем данные:

```
df = pd.read_csv("primary-tumor.csv")
df.head(5)
```

```
   class  age  sex  histologic-type  degree-of-diffe  bone  bone-
marrow  lung  \
0      1    1  1.0              NaN              3.0     2
2     1
1      1    1  1.0              NaN              3.0     2
2     2
2      1    1  2.0              2.0              3.0     1
2     2
3      1    1  2.0              NaN              3.0     1
2     1
4      1    1  2.0              NaN              3.0     1
2     1

   pleura  pertioneum  liver  brain  skin  neck  supraclavicular
axillar  \
0       2           2      2      2   2.0     2                2
2.0
1       2           2      1      2   2.0     2                1
2.0
2       2           2      2      2   2.0     2                2
2.0
3       1           2      2      2   2.0     2                2
```

```
2.0
4       1           2      2      2   2.0     2                  2
2.0

   mediastinum  adbdominal
0            2           2
1            1           2
2            1           2
3            1           2
4            1           2
```

Анализируем данные на пропуски и составляющие:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 339 entries, 0 to 338
Data columns (total 18 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   class            339 non-null    int64
 1   age              339 non-null    int64
 2   sex              338 non-null    float64
 3   histologic-type  272 non-null    float64
 4   degree-of-diffe  184 non-null    float64
 5   bone             339 non-null    int64
 6   bone-marrow      339 non-null    int64
 7   lung             339 non-null    int64
 8   pleura           339 non-null    int64
 9   pertioneum       339 non-null    int64
 10  liver            339 non-null    int64
 11  brain            339 non-null    int64
 12  skin             338 non-null    float64
 13  neck             339 non-null    int64
 14  supraclavicular  339 non-null    int64
 15  axillar          338 non-null    float64
 16  mediastinum      339 non-null    int64
 17  adbdominal       339 non-null    int64
dtypes: float64(5), int64(13)
memory usage: 47.8 KB
```

1. class: lung, head & neck, esophasus, thyroid, stomach, duoden & sm.int, colon, rectum, anus, salivary glands, pancreas, gallblader, liver, kidney, bladder, testis, prostate, ovary, corpus uteri, cervix uteri, vagina, breast
2. age: <30, 30-59, >=60
3. sex: male, female
4. histologic-type: epidermoid, adeno, anaplastic
5. degree-of-diffe: well, fairly, poorly
6. bone: yes, no

7. bone-marrow: yes, no
8. lung: yes, no
9. pleura: yes, no
10. peritoneum: yes, no
11. liver: yes, no
12. brain: yes, no
13. skin: yes, no
14. neck: yes, no
15. supraclavicular: yes, no
16. axillar: yes, no
17. mediastinum: yes, no
18. abdominal: yes, no

Заполняем пропуски медианными значениями:

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

Разделим переменные на целевую и предикторы:

```
target = df[df.columns[[0]]]
df = df.drop(df[df.columns[[0]]], axis=1)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 339 entries, 0 to 338
Data columns (total 17 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   age              339 non-null    int64  
 1   sex              339 non-null    float64
 2   histologic-type  339 non-null    float64
 3   degree-of-diffe  339 non-null    float64
 4   bone             339 non-null    int64  
 5   bone-marrow      339 non-null    int64  
 6   lung             339 non-null    int64  
 7   pleura           339 non-null    int64  
 8   pertioneum       339 non-null    int64  
 9   liver            339 non-null    int64  
 10  brain            339 non-null    int64  
 11  skin             339 non-null    float64
 12  neck             339 non-null    int64  
 13  supraclavicular  339 non-null    int64  
 14  axillar          339 non-null    float64
 15  mediastinum      339 non-null    int64  
 16  adbdominal       339 non-null    int64  
dtypes: float64(5), int64(12)
memory usage: 45.1 KB
```

Изучим количество кластеров:

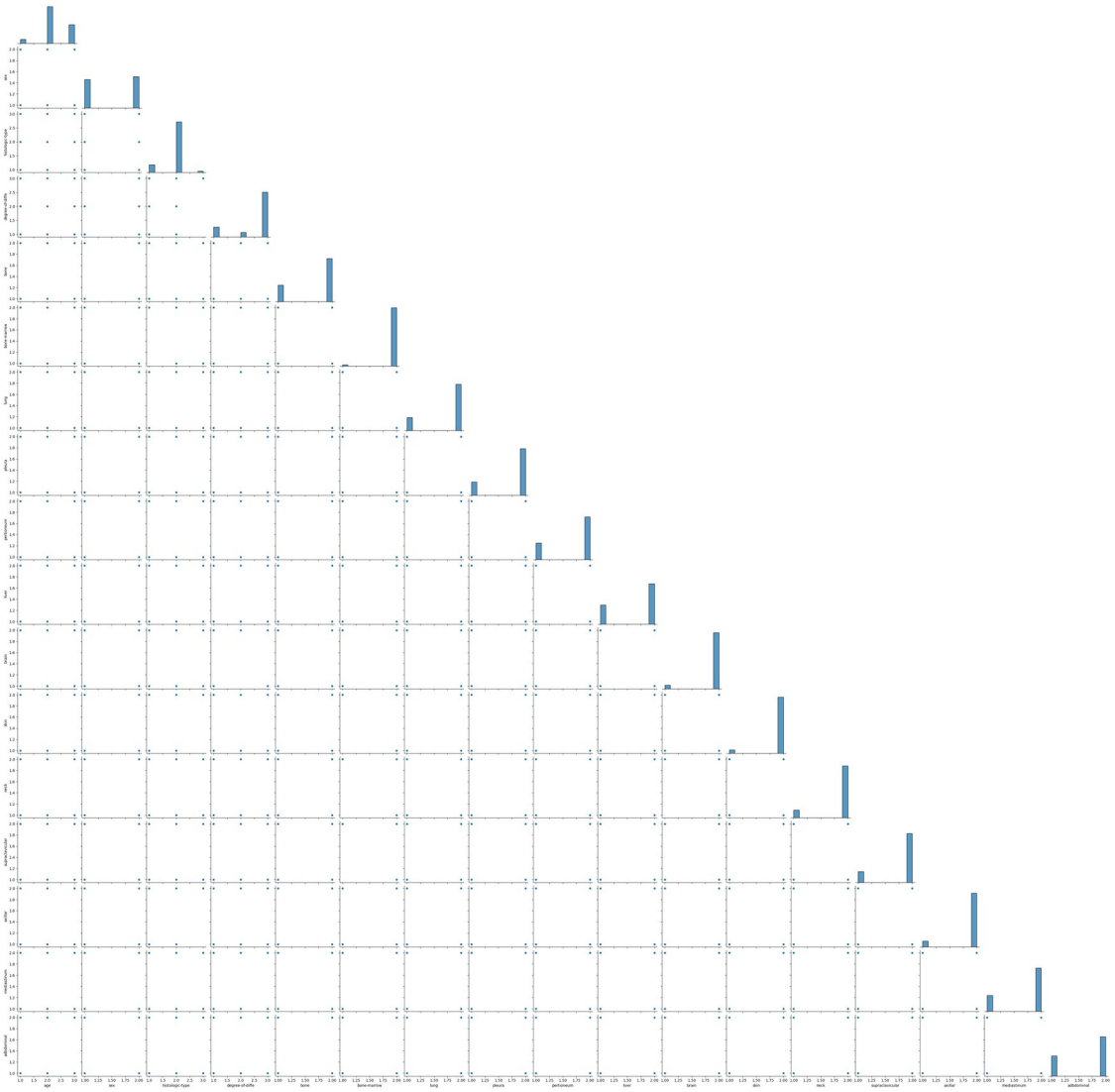
```
len(target.groupby("class").count())
```

```
21
```

Изучим нормальность и зависимость данных:

```
sns.pairplot(df, corner = True)
```

```
<seaborn.axisgrid.PairGrid at 0x19684f54e50>
```



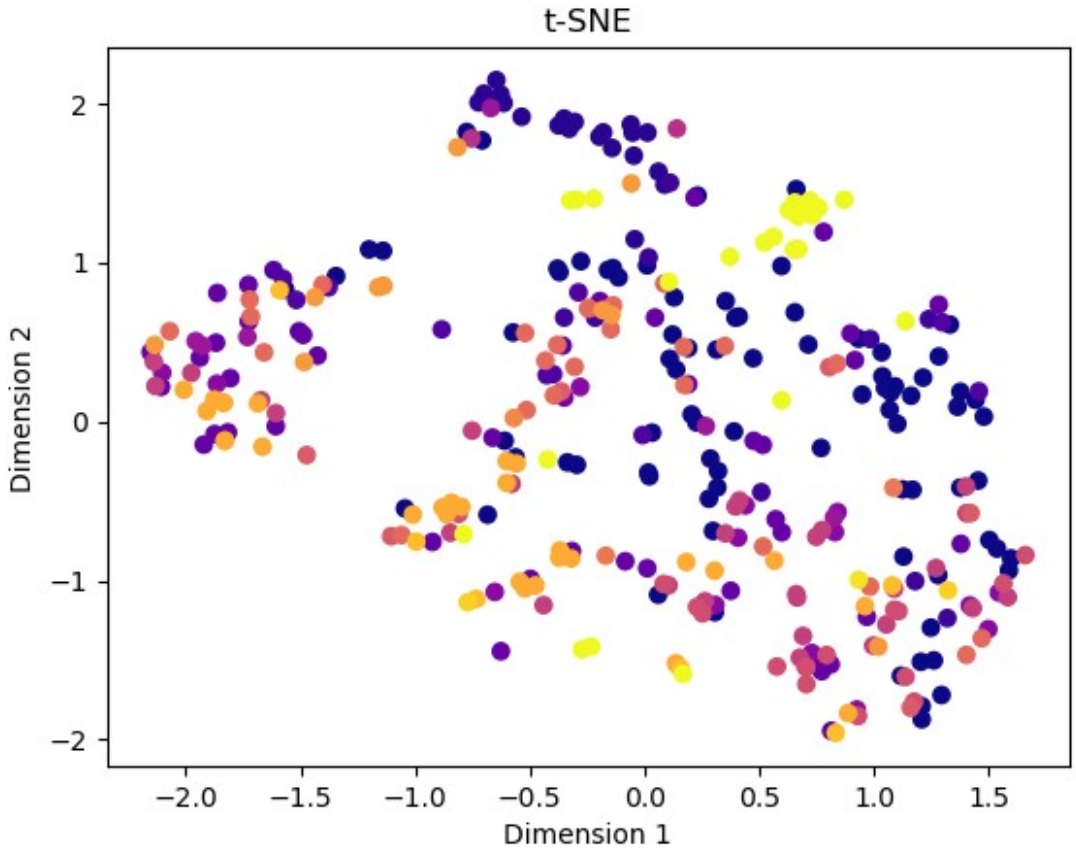
Большая часть данных распределена равномерно.

Понижение размерности t-SNE

```
new_df = df.copy()
model = TSNE(perplexity=10, learning_rate=400, n_iter=5000)
transformed = model.fit_transform(new_df)
```

```
scaler = StandardScaler()
scaler.fit(transformed)
df = pd.DataFrame(scaler.transform(transformed), columns =
["Dimension_1", "Dimension_2"])

plt.scatter(df.iloc[:, 0], df.iloc[:, 1], c = target.values, cmap =
plt.cm.plasma)
plt.xlabel("Dimension 1")
plt.ylabel("Dimension 2")
plt.title("t-SNE")
plt.show()
```



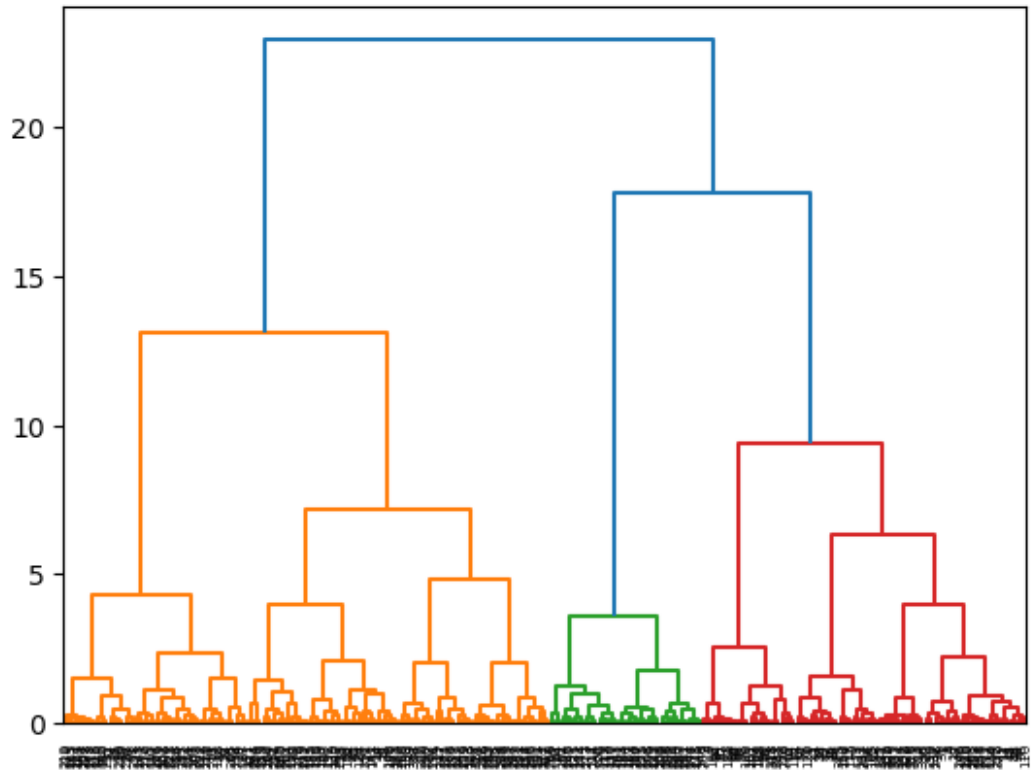
Методом t-SNE была понижена размерность с 17 до двух для дальнейшего анализа.

Иерархические алгоритмы кластеризации данных

Метод Варда

Построим дендрограмму с использованием метода Варда и Евклидового расстояния:

```
new_df = df.copy()
link = linkage(new_df, 'ward', 'euclidean')
dn = dendrogram(link)
```

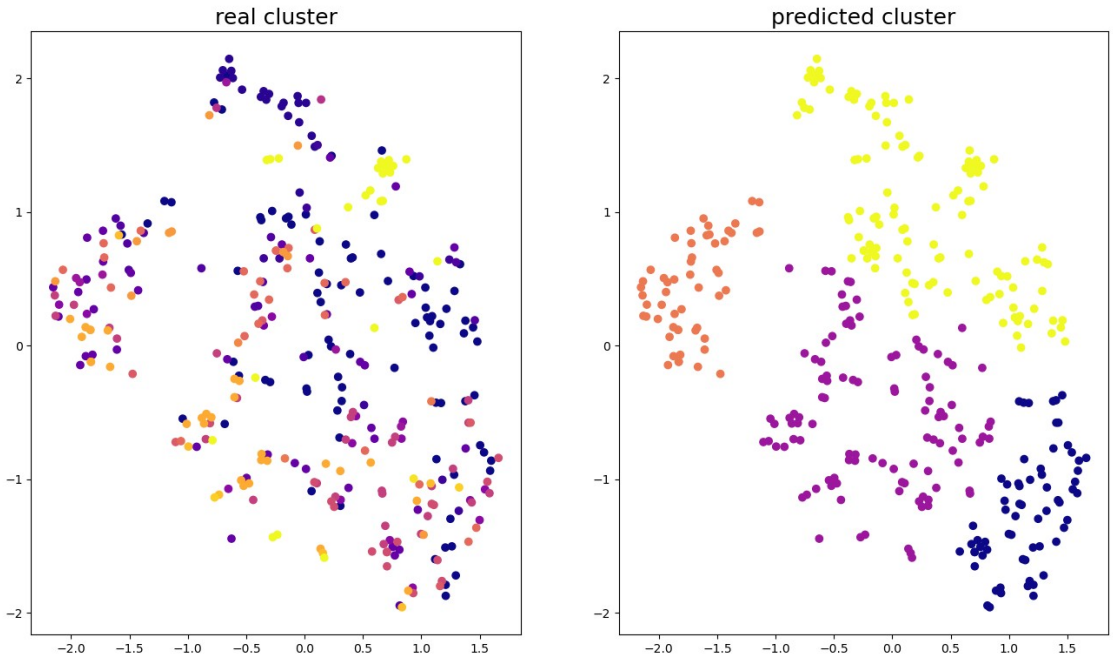


Методом Варда было выделено 4 кластера:

```
result = pd.DataFrame(fcluster(link, 4, criterion='maxclust'),
columns=['target'])

fig, axes = plt.subplots(1, 2, figsize=(16,9))
axes[0].scatter(new_df.iloc[:,0], new_df.iloc[:,1], c=target.values,
cmap = plt.cm.plasma)
axes[0].set_title("real cluster", fontsize = 18)
axes[1].scatter(new_df.iloc[:,0], new_df.iloc[:,1], c=result.values,
cmap = plt.cm.plasma)
axes[1].set_title("predicted cluster", fontsize = 18)

Text(0.5, 1.0, 'predicted cluster')
```



Оценим качество кластеризации:

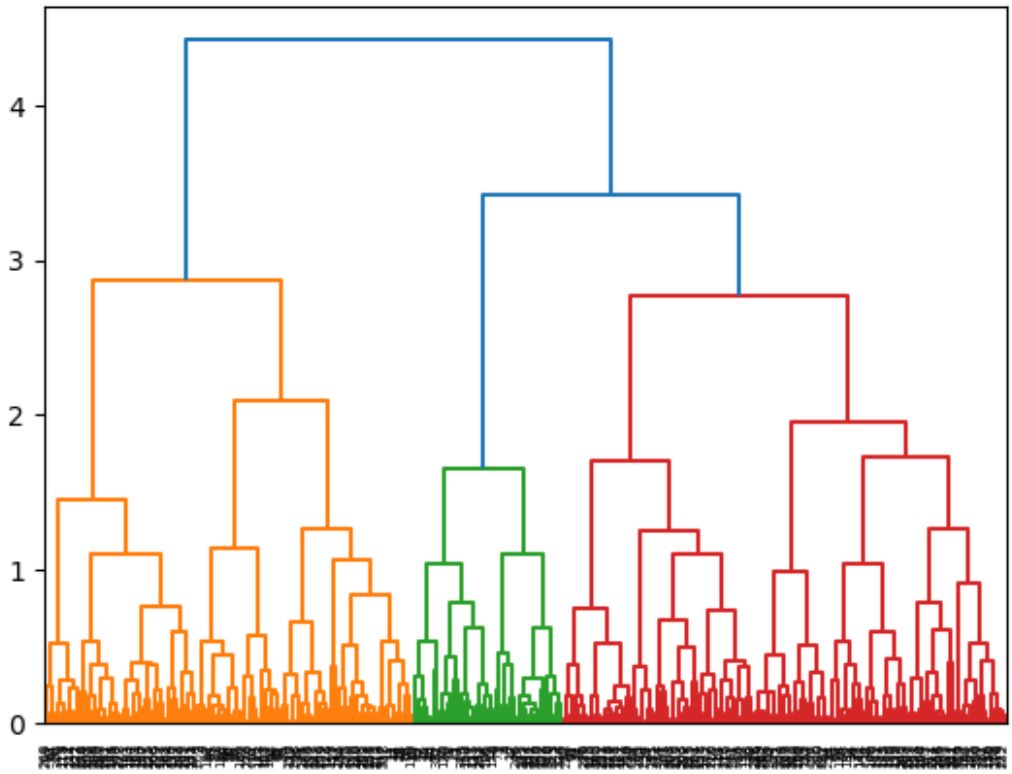
```
print("Скорректированный индекс Рэнда: %0.3f" %
adjusted_rand_score(target.values[:,0], result.values[:,0]))
print("Коэффициент изменения информации: %0.3f" %
adjusted_mutual_info_score(target.values[:,0], result.values[:,0]))
print("Коэффициент качества кластеризации: %0.3f" %
silhouette_score(new_df, result.values[:,0]))
print("Качество однородности: %0.3f" %
homogeneity_score(target.values[:,0], result.values[:,0]))
print("Качество полноты: %0.3f" %
completeness_score(target.values[:,0], result.values[:,0]))
print("Метрика V-Measure: %0.3f" % v_measure_score(target.values[:,0],
result.values[:,0]))
```

```
Скорректированный индекс Рэнда: 0.071
Коэффициент изменения информации: 0.148
Коэффициент качества кластеризации: 0.411
Качество однородности: 0.146
Качество полноты: 0.275
Метрика V-Measure: 0.190
```

Метод полной связи

Построим дендрограмму с использованием метода полной связи и Евклидового расстояния:

```
new_df = df.copy()
link = linkage(new_df, 'complete', 'euclidean')
dn = dendrogram(link)
```

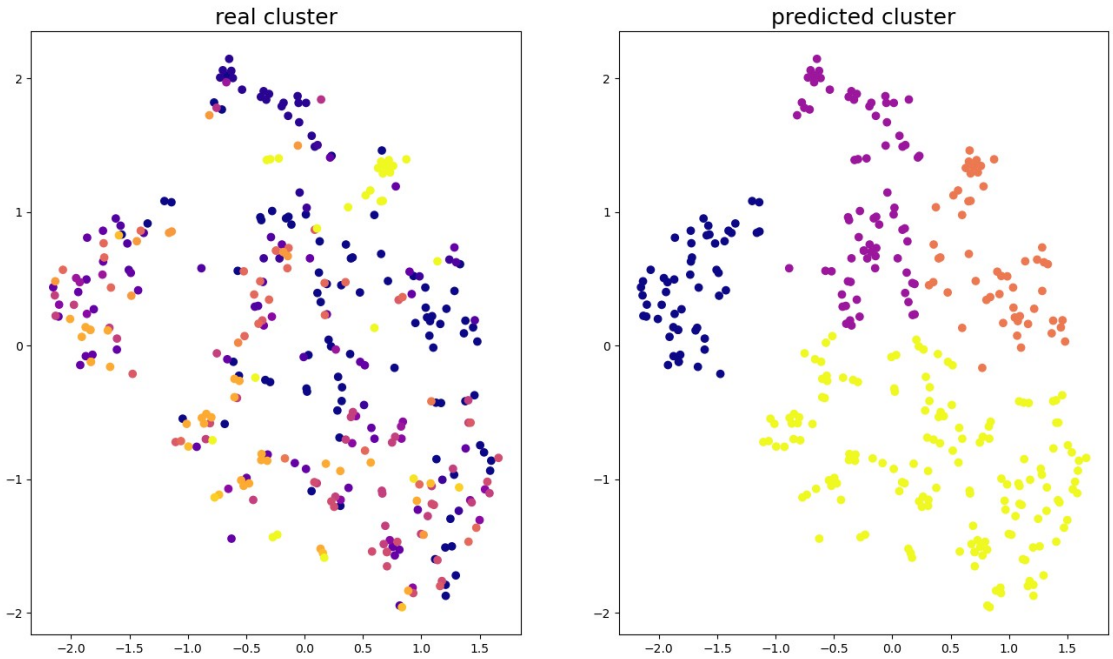


Методом полной связи было выделено 4 кластера:

```
result = pd.DataFrame(fcluster(link, 4, criterion='maxclust'),
columns=['target'])

fig, axes = plt.subplots(1, 2, figsize=(16,9))
axes[0].scatter(new_df.iloc[:,0], new_df.iloc[:,1], c=target.values,
cmap = plt.cm.plasma)
axes[0].set_title("real cluster", fontsize = 18)
axes[1].scatter(new_df.iloc[:,0], new_df.iloc[:,1], c=result.values,
cmap = plt.cm.plasma)
axes[1].set_title("predicted cluster", fontsize = 18)

Text(0.5, 1.0, 'predicted cluster')
```



Оценим качество кластеризации:

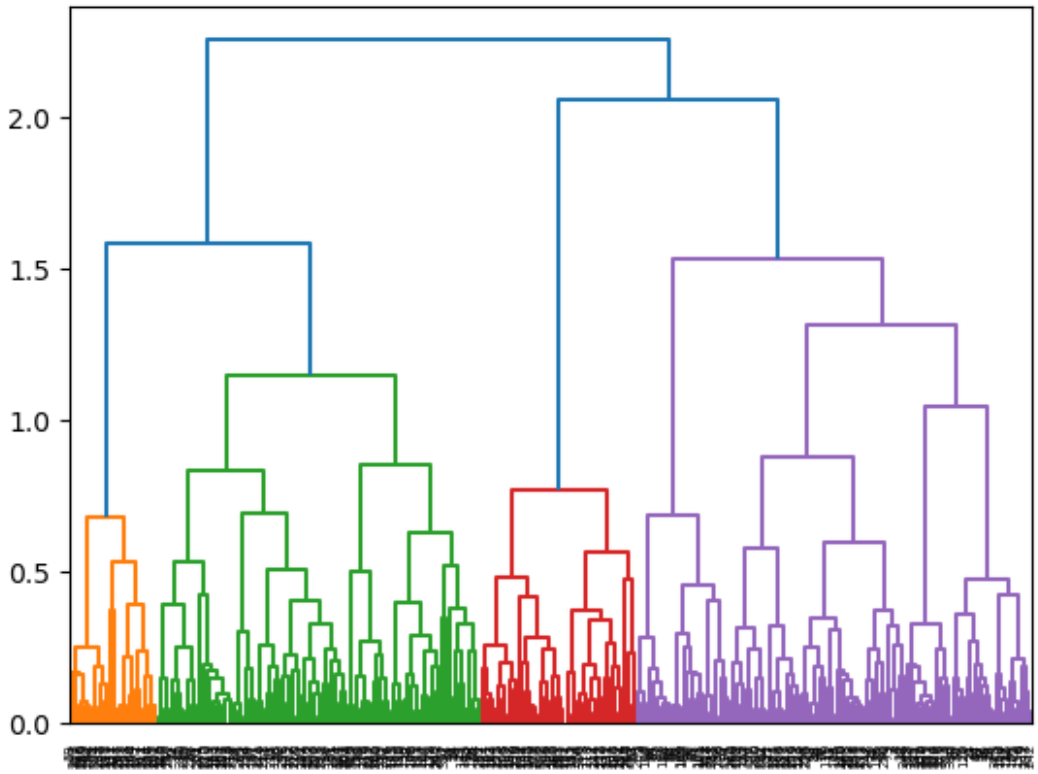
```
print("Скорректированный индекс Рэнда: %0.3f" %
adjusted_rand_score(target.values[:,0], result.values[:,0]))
print("Коэффициент изменения информации: %0.3f" %
adjusted_mutual_info_score(target.values[:,0], result.values[:,0]))
print("Коэффициент качества кластеризации: %0.3f" %
silhouette_score(new_df, result.values[:,0]))
print("Качество однородности: %0.3f" %
homogeneity_score(target.values[:,0], result.values[:,0]))
print("Качество полноты: %0.3f" %
completeness_score(target.values[:,0], result.values[:,0]))
print("Метрика V-Measure: %0.3f" % v_measure_score(target.values[:,0],
result.values[:,0]))
```

```
Скорректированный индекс Рэнда: 0.057
Коэффициент изменения информации: 0.174
Коэффициент качества кластеризации: 0.385
Качество однородности: 0.162
Качество полноты: 0.320
Метрика V-Measure: 0.215
```

Метод средней связи

Построим дендрограмму с использованием с метода средней и Евклидового расстояния:

```
new_df = df.copy()
link = linkage(new_df, 'average', 'euclidean')
dn = dendrogram(link)
```

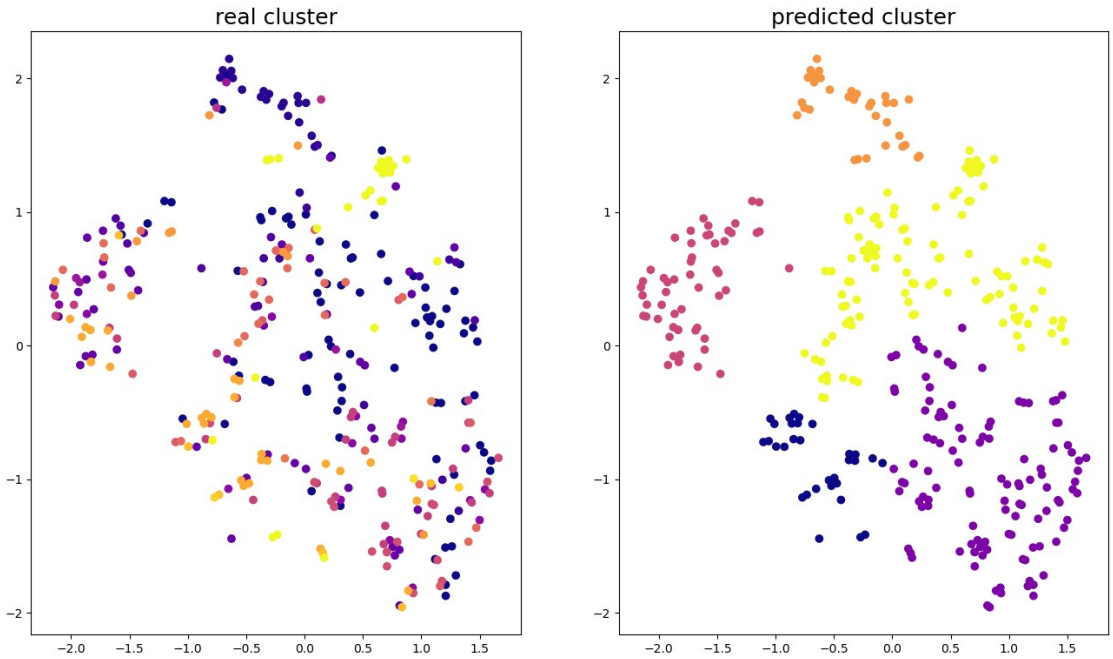


Методом средней связи было выделено 5 кластеров:

```
result = pd.DataFrame(fcluster(link, 5, criterion='maxclust'),
columns=['target'])

fig, axes = plt.subplots(1, 2, figsize=(16,9))
axes[0].scatter(new_df.iloc[:,0], new_df.iloc[:,1], c=target.values,
cmap = plt.cm.plasma)
axes[0].set_title("real cluster", fontsize = 18)
axes[1].scatter(new_df.iloc[:,0], new_df.iloc[:,1], c=result.values,
cmap = plt.cm.plasma)
axes[1].set_title("predicted cluster", fontsize = 18)

Text(0.5, 1.0, 'predicted cluster')
```



Оценим качество кластеризации:

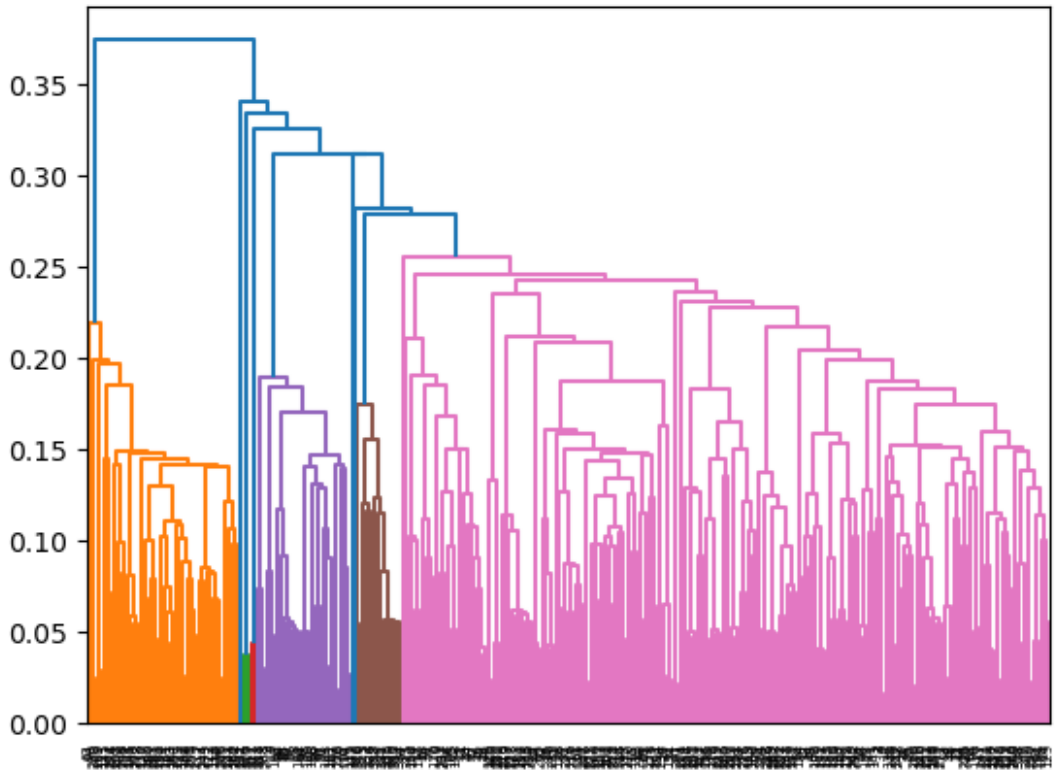
```
print("Скорректированный индекс Рэнда: %0.3f" %
adjusted_rand_score(target.values[:,0], result.values[:,0]))
print("Коэффициент изменения информации: %0.3f" %
adjusted_mutual_info_score(target.values[:,0], result.values[:,0]))
print("Коэффициент качества кластеризации: %0.3f" %
silhouette_score(new_df, result.values[:,0]))
print("Качество однородности: %0.3f" %
homogeneity_score(target.values[:,0], result.values[:,0]))
print("Качество полноты: %0.3f" %
completeness_score(target.values[:,0], result.values[:,0]))
print("Метрика V-Measure: %0.3f" % v_measure_score(target.values[:,0],
result.values[:,0]))
```

```
Скорректированный индекс Рэнда: 0.105
Коэффициент изменения информации: 0.186
Коэффициент качества кластеризации: 0.390
Качество однородности: 0.186
Качество полноты: 0.320
Метрика V-Measure: 0.236
```

Метод одиночной связи

Построим дендрограмму с использованием метода одиночной связи и Евклидового расстояния:

```
new_df = df.copy()
link = linkage(new_df, 'single', 'euclidean')
dn = dendrogram(link)
```

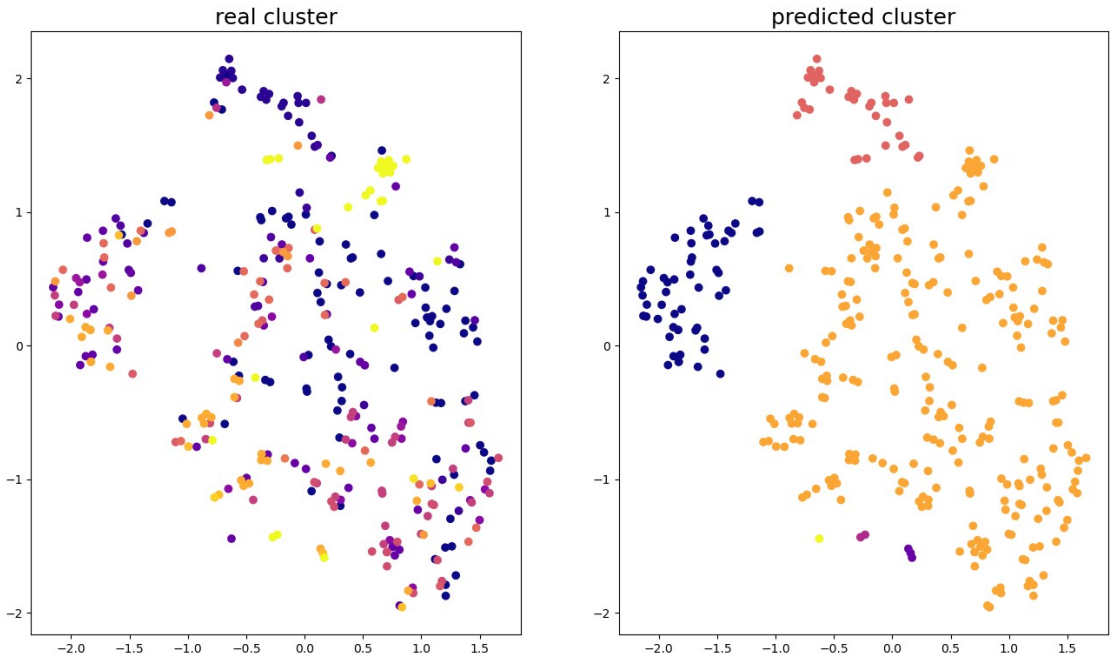


Методом одиночной связи было выделено 6 кластеров:

```
result = pd.DataFrame(fcluster(link, 6, criterion='maxclust'),
columns=['target'])

fig, axes = plt.subplots(1, 2, figsize=(16,9))
axes[0].scatter(new_df.iloc[:,0], new_df.iloc[:,1], c=target.values,
cmap = plt.cm.plasma)
axes[0].set_title("real cluster", fontsize = 18)
axes[1].scatter(new_df.iloc[:,0], new_df.iloc[:,1], c=result.values,
cmap = plt.cm.plasma)
axes[1].set_title("predicted cluster", fontsize = 18)

Text(0.5, 1.0, 'predicted cluster')
```



Оценим качество кластеризации:

```
print("Скорректированный индекс Рэнда: %0.3f" %
adjusted_rand_score(target.values[:,0], result.values[:,0]))
print("Коэффициент изменения информации: %0.3f" %
adjusted_mutual_info_score(target.values[:,0], result.values[:,0]))
print("Коэффициент качества кластеризации: %0.3f" %
silhouette_score(new_df, result.values[:,0]))
print("Качество однородности: %0.3f" %
homogeneity_score(target.values[:,0], result.values[:,0]))
print("Качество полноты: %0.3f" %
completeness_score(target.values[:,0], result.values[:,0]))
print("Метрика V-Measure: %0.3f" % v_measure_score(target.values[:,0],
result.values[:,0]))
```

```
Скорректированный индекс Рэнда: 0.065
Коэффициент изменения информации: 0.138
Коэффициент качества кластеризации: 0.061
Качество однородности: 0.124
Качество полноты: 0.372
Метрика V-Measure: 0.186
```

Итерационные алгоритмы кластерного анализа

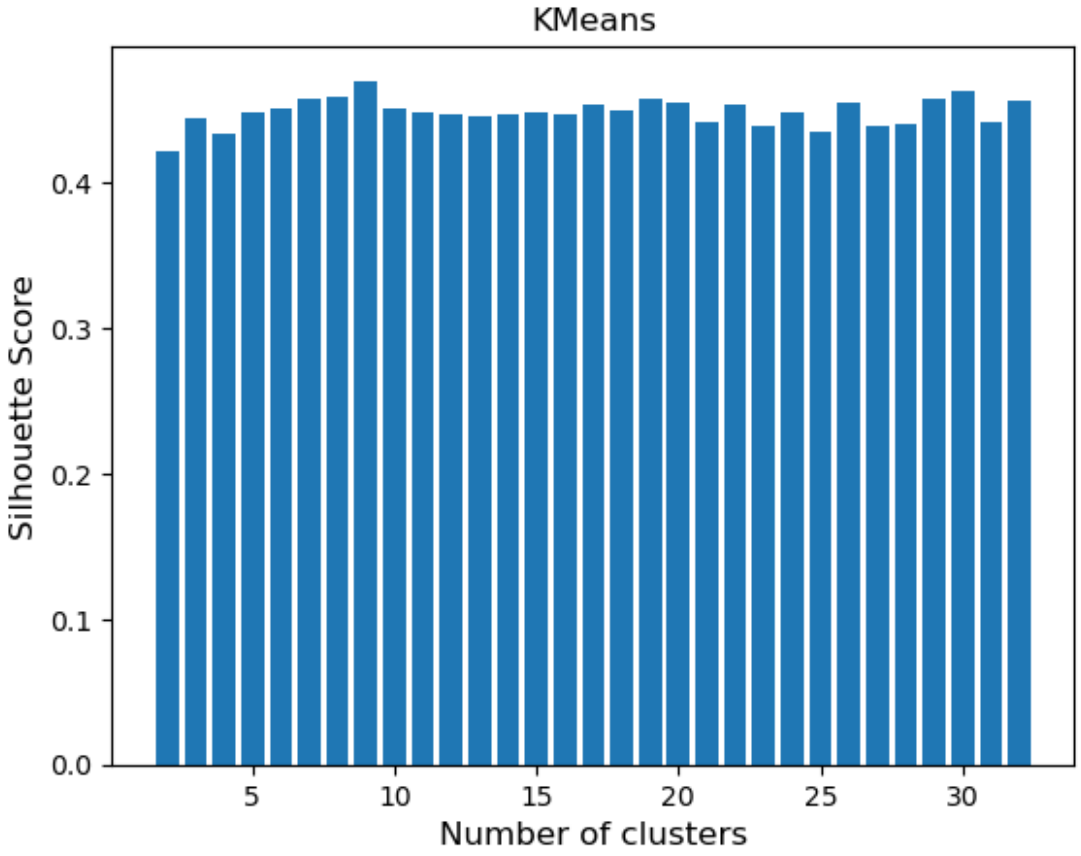
Метод k-средних

Посмотрим график для нахождения оптимального количества кластеров:

```
silhouette_scores = []
count = [x for x in range(2, 33)]

for n_cluster in count:
    silhouette_scores.append(
        silhouette_score(new_df, KMeans(n_clusters = n_cluster, n_init
= 10).fit_predict(new_df)))

plt.bar(count, silhouette_scores)
plt.xlabel('Number of clusters', fontsize = 12)
plt.ylabel('Silhouette Score', fontsize = 12)
plt.title('KMeans')
plt.show()
```



Оптимальное количество кластеров коэффициентом Силха считается 9.

Анализ для неизвестного числа кластеров

```
new_df = df.copy()
```

Описываем модель для девяти кластеров, визуально оцененого оптимального количества кластеров по графику:

```
model = KMeans(n_clusters = 9, n_init = 10)
model.fit(new_df)
```

```
KMeans(n_clusters=9, n_init=10)
```

Прогнозируем:

```
prediction = model.predict(new_df)
result = pd.DataFrame(prediction, columns=['target'])
result.head()
```

```
   target
0       3
1       3
2       4
3       4
4       4
```

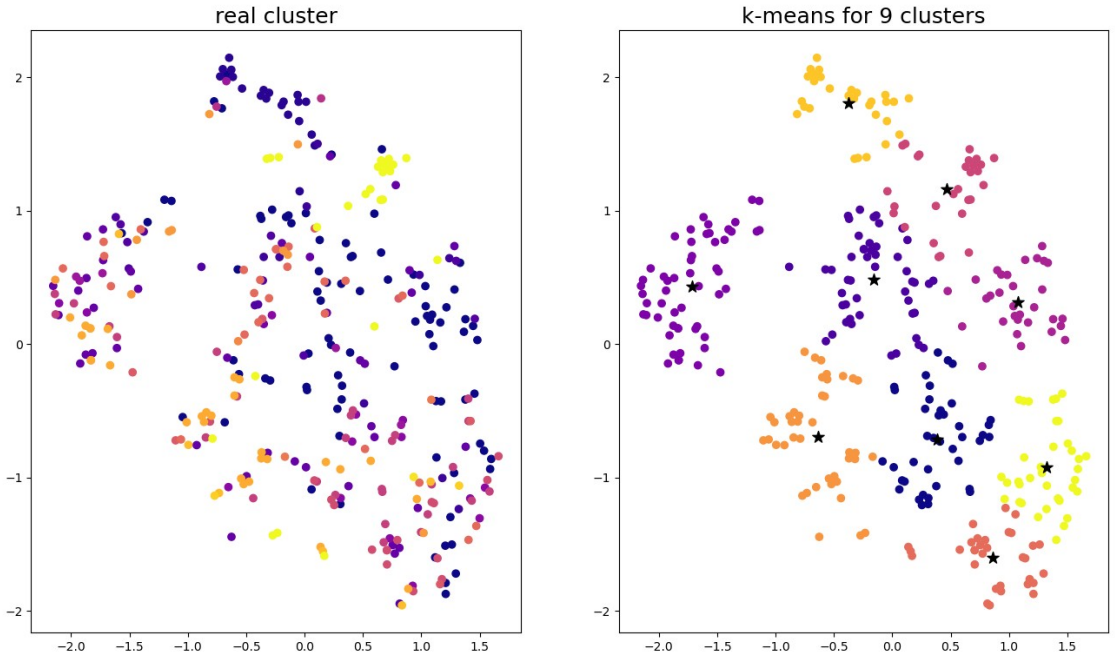
Подготавливаем данные для графика:

```
result = pd.DataFrame(prediction, columns=['target'])

fig, axes = plt.subplots(1, 2, figsize=(16,9))
axes[0].scatter(new_df.iloc[:,0], new_df.iloc[:,1], c=target.values,
cmap = plt.cm.plasma)
axes[0].set_title("real cluster", fontsize = 18)
axes[1].scatter(new_df.iloc[:,0], new_df.iloc[:,1], c=result.values,
cmap = plt.cm.plasma)
axes[1].set_title("k-means for 9 clusters", fontsize = 18)

plt.scatter(model.cluster_centers_[:,0], model.cluster_centers_[:,1],
s = 100, color='black', marker='*')
```

```
<matplotlib.collections.PathCollection at 0x1969bca5f40>
```



Оценим качество кластеризации:

```
print("Скорректированный индекс Рэнда: %0.3f" %
adjusted_rand_score(target.values[:,0], result.values[:,0]))
print("Коэффициент изменения информации: %0.3f" %
adjusted_mutual_info_score(target.values[:,0], result.values[:,0]))
print("Коэффициент качества кластеризации: %0.3f" %
silhouette_score(new_df, result.values[:,0]))
print("Качество однородности: %0.3f" %
homogeneity_score(target.values[:,0], result.values[:,0]))
print("Качество полноты: %0.3f" %
completeness_score(target.values[:,0], result.values[:,0]))
print("Метрика V-Measure: %0.3f" % v_measure_score(target.values[:,0],
result.values[:,0]))
```

```
Скорректированный индекс Рэнда: 0.098
Коэффициент изменения информации: 0.212
Коэффициент качества кластеризации: 0.463
Качество однородности: 0.271
Качество полноты: 0.315
Метрика V-Measure: 0.291
```

Анализ для известного числа кластеров

Так как известно изначальное количество категорий, проведем анализ для 21 кластера:

```
new_df = df.copy()

model = KMeans(n_clusters=21, n_init = 10)
model.fit(new_df)

KMeans(n_clusters=21, n_init=10)

prediction = model.predict(new_df)
result = pd.DataFrame(prediction, columns=['target'])
result.head()
```

	target
0	13
1	0
2	17
3	17
4	17

Подготавливаем данные для графика:

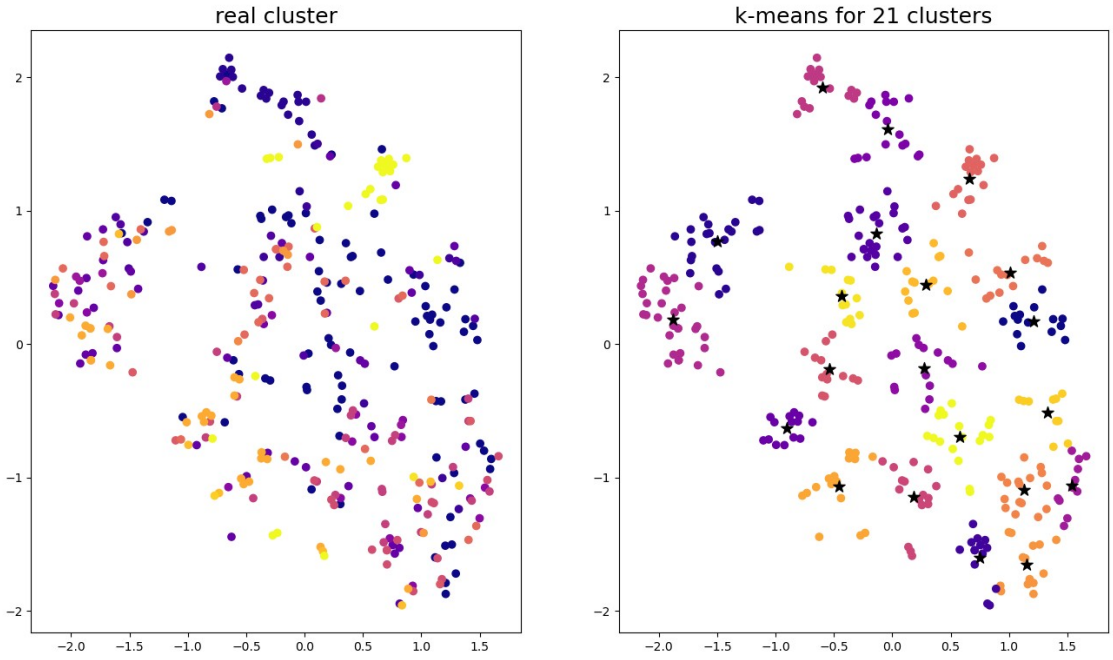
```
result = pd.DataFrame(prediction, columns=['target'])

fig, axes = plt.subplots(1, 2, figsize=(16,9))
axes[0].scatter(new_df.iloc[:,0], new_df.iloc[:,1], c=target.values,
cmap = plt.cm.plasma)
axes[0].set_title("real cluster", fontsize = 18)
```

```
axes[1].scatter(new_df.iloc[:,0], new_df.iloc[:,1], c=result.values,
cmap = plt.cm.plasma)
axes[1].set_title("k-means for 21 clusters", fontsize = 18)

plt.scatter(model.cluster_centers_[:,0], model.cluster_centers_[:,1],
s = 100, color='black', marker='*')
```

```
<matplotlib.collections.PathCollection at 0x1969bd10fd0>
```



Оценим качество кластеризации:

```
print("Скорректированный индекс Рэнда: %0.3f" %
adjusted_rand_score(target.values[:,0], result.values[:,0]))
print("Коэффициент изменения информации: %0.3f" %
adjusted_mutual_info_score(target.values[:,0], result.values[:,0]))
print("Коэффициент качества кластеризации: %0.3f" %
silhouette_score(new_df, result.values[:,0]))
print("Качество однородности: %0.3f" %
homogeneity_score(target.values[:,0], result.values[:,0]))
print("Качество полноты: %0.3f" %
completeness_score(target.values[:,0], result.values[:,0]))
print("Метрика V-Measure: %0.3f" % v_measure_score(target.values[:,0],
result.values[:,0]))
```

```
Скорректированный индекс Рэнда: 0.090
Коэффициент изменения информации: 0.209
Коэффициент качества кластеризации: 0.457
Качество однородности: 0.389
Качество полноты: 0.327
Метрика V-Measure: 0.355
```

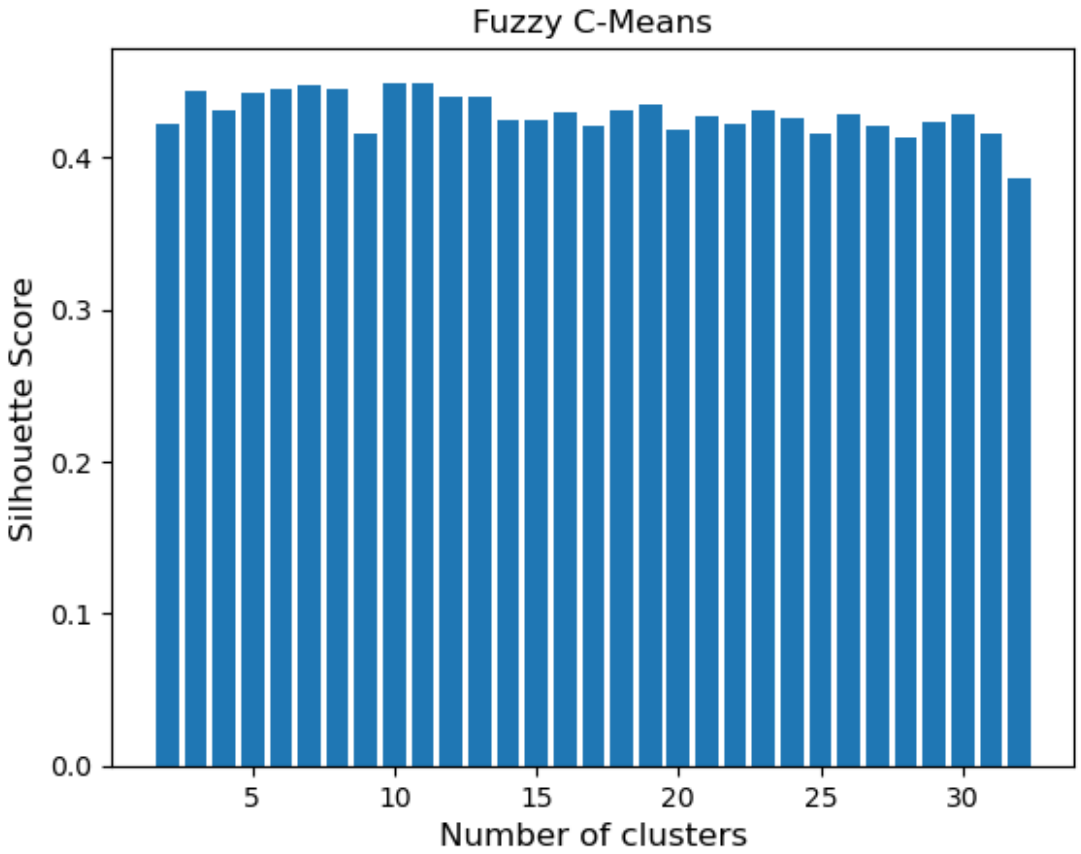
Fuzzy C-Means алгоритм кластеризации

Анализ для неизвестного числа кластеров

```
silhouette_scores = []
count = [x for x in range(2, 33)]

for n_cluster in count:
    model = FCM(n_clusters = n_cluster)
    model.fit(new_df.values)
    silhouette_scores.append(silhouette_score(new_df,
model.predict(new_df.values)))

plt.bar(count, silhouette_scores)
plt.xlabel('Number of clusters', fontsize = 12)
plt.ylabel('Silhouette Score', fontsize = 12)
plt.title('Fuzzy C-Means')
plt.show()
```




```
new_df = df.copy()
```

Обучим модель нечеткой кластеризации с количеством кластеров равным 10:

```
model = FCM(n_clusters = 10)
model.fit(new_df.values)
center = model.centers
```

Спрогнозируем значения:

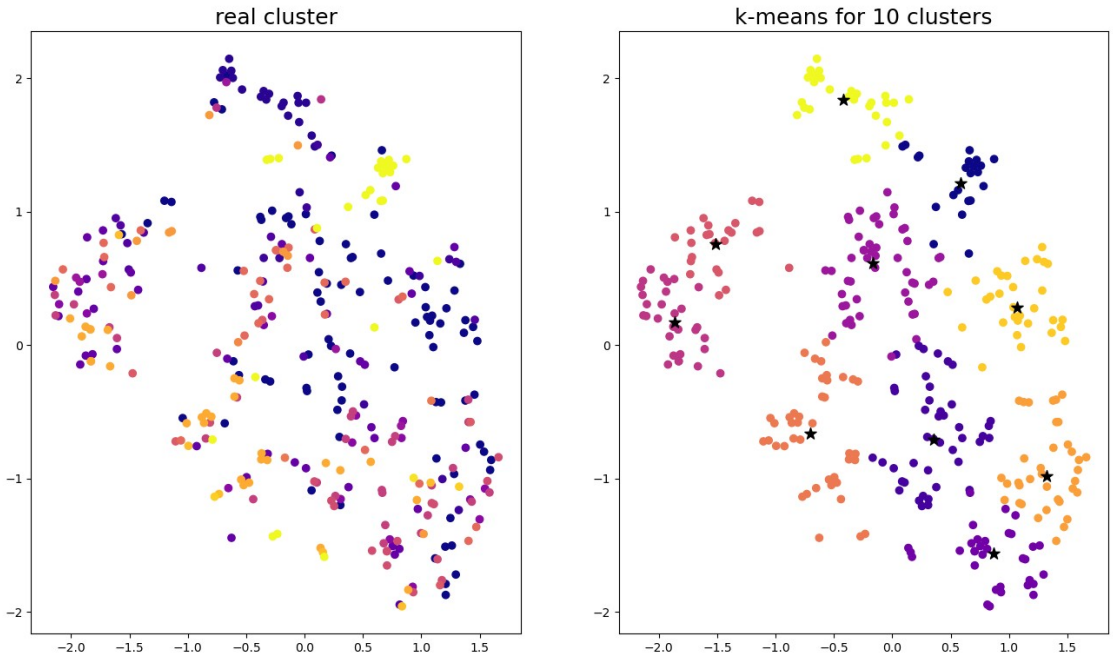
```
result = pd.DataFrame(model.predict(new_df.values),
columns=['target'])
result.head()
```

```
   target
0       8
1       8
2       0
3       3
4       3
```

```
fig, axes = plt.subplots(1, 2, figsize=(16,9))
axes[0].scatter(new_df.iloc[:,0], new_df.iloc[:,1], c=target.values,
cmap = plt.cm.plasma)
axes[0].set_title("real cluster", fontsize = 18)
axes[1].scatter(new_df.iloc[:,0], new_df.iloc[:,1], c=result.values,
cmap = plt.cm.plasma)
axes[1].set_title("k-means for 10 clusters", fontsize = 18)

plt.scatter(center[:,0], center[:,1], s = 100, color='black',
marker='*')
```

```
<matplotlib.collections.PathCollection at 0x1969c793a60>
```



Оценим качество кластеризации:

```
print("Скорректированный индекс Рэнда: %0.3f" %
adjusted_rand_score(target.values[:,0], result.values[:,0]))
print("Коэффициент изменения информации: %0.3f" %
adjusted_mutual_info_score(target.values[:,0], result.values[:,0]))
print("Коэффициент качества кластеризации: %0.3f" %
silhouette_score(new_df, result.values[:,0]))
print("Качество однородности: %0.3f" %
homogeneity_score(target.values[:,0], result.values[:,0]))
print("Качество полноты: %0.3f" %
completeness_score(target.values[:,0], result.values[:,0]))
print("Метрика V-Measure: %0.3f" % v_measure_score(target.values[:,0],
result.values[:,0]))
```

```
Скорректированный индекс Рэнда: 0.106
Коэффициент изменения информации: 0.220
Коэффициент качества кластеризации: 0.449
Качество однородности: 0.289
Качество полноты: 0.321
Метрика V-Measure: 0.305
```

Анализ для известного числа кластеров

```
new_df = df.copy()
```

Модель для известной 21 категории признаков:

```
model = FCM(n_clusters = 21)
model.fit(new_df.values)
center = model.centers
```

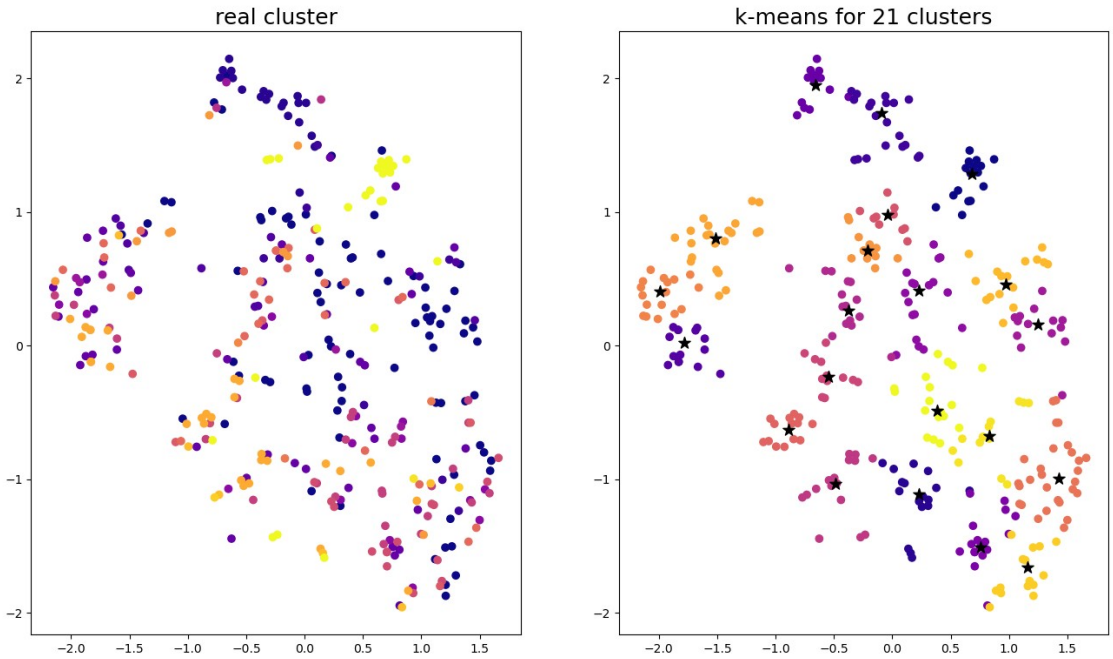
```
result = pd.DataFrame(model.predict(new_df.values),
columns=['target'])
result.head()
```

```
   target
0      17
1       7
2       6
3       6
4       6
```

```
fig, axes = plt.subplots(1, 2, figsize=(16,9))
axes[0].scatter(new_df.iloc[:,0], new_df.iloc[:,1], c=target.values,
cmap = plt.cm.plasma)
axes[0].set_title("real cluster", fontsize = 18)
axes[1].scatter(new_df.iloc[:,0], new_df.iloc[:,1], c=result.values,
cmap = plt.cm.plasma)
axes[1].set_title("k-means for 21 clusters", fontsize = 18)

plt.scatter(center[:,0], center[:,1], s = 100, color='black',
marker='*')
```

```
<matplotlib.collections.PathCollection at 0x1969c819310>
```



Оценим качество кластеризации:

```
print("Скорректированный индекс Рэнда: %0.3f" %
adjusted_rand_score(target.values[:,0], result.values[:,0]))
print("Коэффициент изменения информации: %0.3f" %
adjusted_mutual_info_score(target.values[:,0], result.values[:,0]))
print("Коэффициент качества кластеризации: %0.3f" %
silhouette_score(new_df, result.values[:,0]))
print("Качество однородности: %0.3f" %
homogeneity_score(target.values[:,0], result.values[:,0]))
print("Качество полноты: %0.3f" %
completeness_score(target.values[:,0], result.values[:,0]))
print("Метрика V-Measure: %0.3f" % v_measure_score(target.values[:,0],
result.values[:,0]))
```

```
Скорректированный индекс Рэнда: 0.093
Коэффициент изменения информации: 0.220
Коэффициент качества кластеризации: 0.411
Качество однородности: 0.400
Качество полноты: 0.335
Метрика V-Measure: 0.365
```

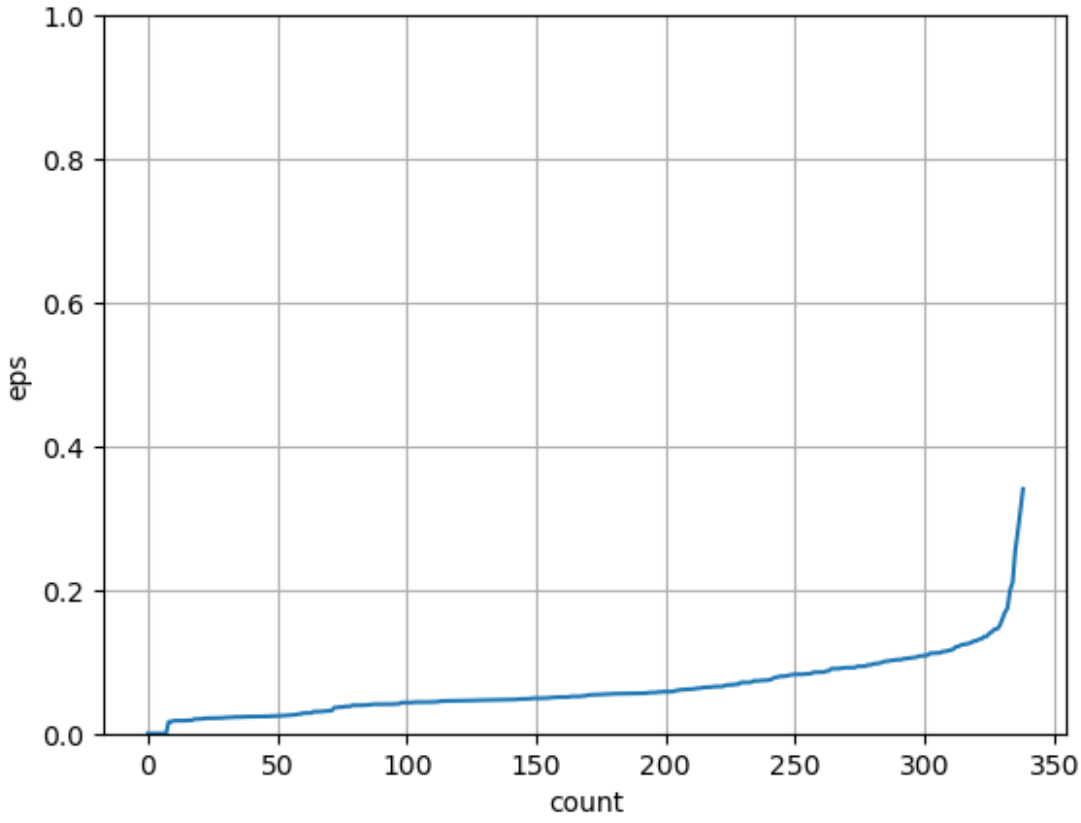
Метод кластеризации на основе плотности DBSCAN

```
new_df = df.copy()
```

Исследуем данные для оптимального определения значения эпсилон для дальнейшей кластеризации:

```
neighbors = NearestNeighbors(n_neighbors = 5)
neighbors_fit = neighbors.fit(df)
distances, indices = neighbors_fit.kneighbors(df)
```

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



Обучим модель:

```
model = DBSCAN(eps = 0.25, min_samples = 5)
model.fit(new_df)
result = pd.DataFrame(model.fit_predict(new_df), columns = ['target'])
```

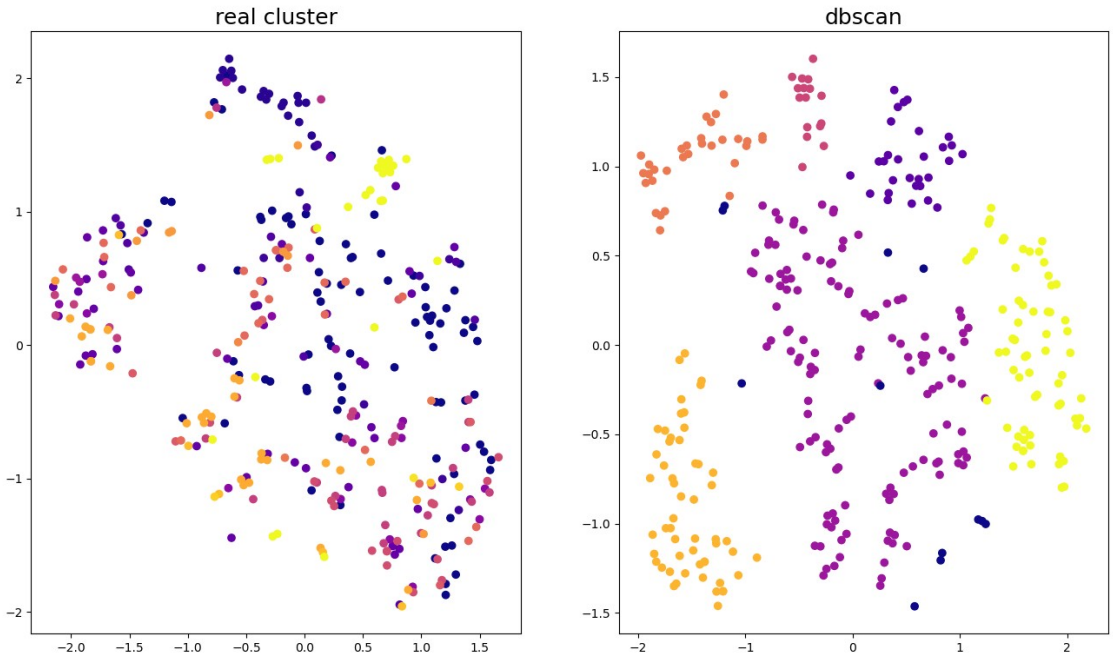
Визуальный анализ и сравнение результатов кластеризации:

```
comp = PCA(n_components = 2).fit_transform(new_df)

fig, axes = plt.subplots(1, 2, figsize=(16,9))
axes[0].scatter(new_df.iloc[:,0], new_df.iloc[:,1], c = target.values,
cmap = plt.cm.plasma)
axes[0].set_title("real cluster", fontsize = 18)
axes[1].scatter(comp[:,0], comp[:,1], c = result.values, cmap =
```

```
plt.cm.plasma)
axes[1].set_title("dbscan", fontsize = 18)

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



Оценим кластеризацию:

```
print("Число кластеров: " + str(len(set(model.labels_))-1))
print("Количество шума: " + str(list(model.labels_).count(-1)))
print("Скорректированный индекс Рэнда: %0.3f" %
adjusted_rand_score(target.values[:,0], result.values[:,0]))
print("Коэффициент изменения информации: %0.3f" %
adjusted_mutual_info_score(target.values[:,0], result.values[:,0]))
print("Коэффициент качества кластеризации: %0.3f" %
silhouette_score(new_df, result.values[:,0]))
print("Качество однородности: %0.3f" %
homogeneity_score(target.values[:,0], result.values[:,0]))
print("Качество полноты: %0.3f" %
completeness_score(target.values[:,0], result.values[:,0]))
print("Метрика V-Measure: %0.3f" % v_measure_score(target.values[:,0],
result.values[:,0]))

Число кластеров: 6
Количество шума: 12
Скорректированный индекс Рэнда: 0.069
Коэффициент изменения информации: 0.182
Коэффициент качества кластеризации: 0.339
Качество однородности: 0.207
Качество полноты: 0.313
Метрика V-Measure: 0.249
```

Вывод

Этот набор данных содержит категориальные и бинарные аттрибуты для задачи классификации первичных опухолей пациентов медицинского центра онкологии Любляна. Для классификации были использованы различные алгоритмы кластерного анализа, однако после понижения размерности наиболее релевантными алгоритмами оказались kmeans, базирующийся на расстоянии между объектами и DBSCAN, основанный на плотности в пространстве, группируя близко расположенные точки.