Лосева Елизавета Юрьевна, группа 8-2

Лабораторная работа № 9

Вариант № 7

Исследование алгоритмов кластеризации

Цель работы: Исследовать методов кластеризации на примере алгоритмов иерархической группировки и k-средних (k-means).

Задание

Содержание работы

Получить у преподавателя вариант задания и написать код, реализующий соответствующий алгоритм обработки информации. Для ответа на поставленные в задании вопросы провести численный эксперимент или статистическое имитационное моделирование и представить соответствующие графики. Провести анализ полученных результатов и представить его в виде выводов по проделанной работе.

7. Сравнить иерархический алгоритм и k-means по критерию вероятности ошибки классификации при использовании определённых метрик: абсолютной, Евклидовой.

Выполнение

Код:

```
import matplotlib
import numpy as np
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
import scipy.cluster.hierarchy as sch
from pyclustering.cluster.center_initializer import random_center_initializer
from pyclustering.cluster.encoder import cluster_encoder
from pyclustering.cluster.encoder import type_encoding
from pyclustering.cluster.kmeans import kmeans as pcc_kmeans
from pyclustering.utils.metric import distance_metric, type_metric
from scipy.optimize import linear_sum_assignment
from sklearn.cluster import AgglomerativeClustering
from sklearn.metrics import confusion_matrix

import sys
import io

sys.stdout = io.TextIOWrapper(sys.stdout.buffer, encoding='utf-8')

random_state = 42
```

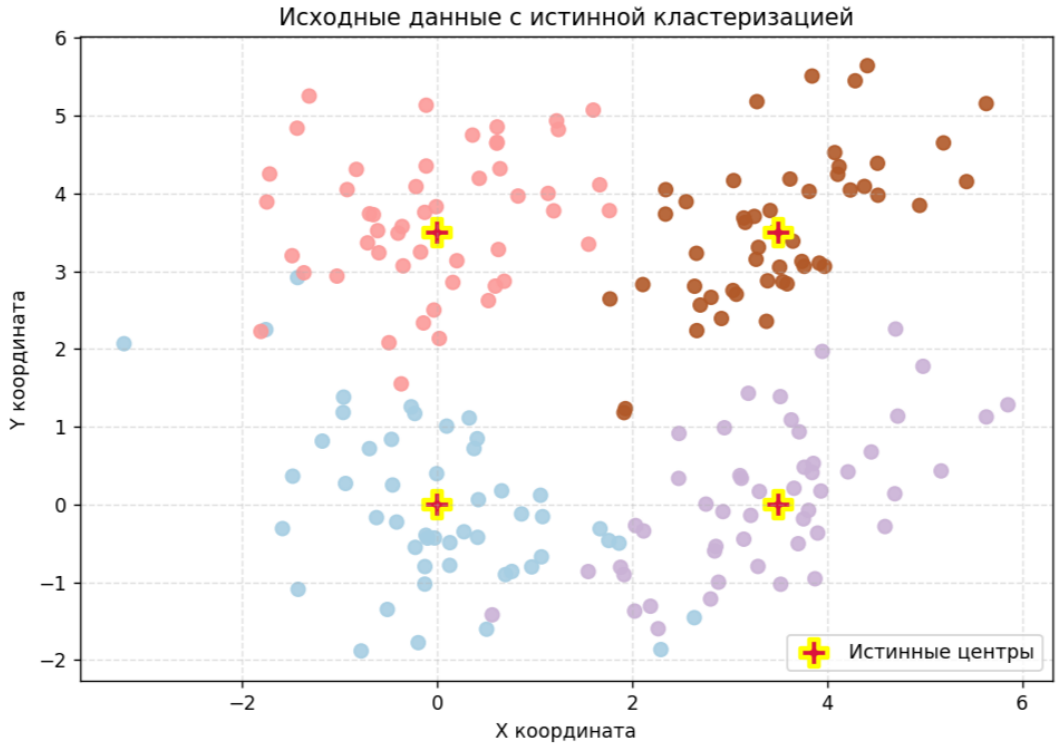
```
np.random.seed(random_state)

n = 2  # размерность
M = 4  # число классов
Ni = 50  # количество объектов в классе
dm = 3.5  # параметр, определяющий среднюю степень пересечения компонентов смесей
ro_min = -0.9
ro_max = 0.9

means_all = np.array([
    [0.0, 0.0],
    [0.0, dm],
    [dm, 0.0],
    [dm, dm],
    [-dm, -dm]
])

# Изменённые названия для метрик
metric_names_dict = {
    'manhattan': 'Манхэттенская метрика',
    'euclidean': 'Евклидова метрика',
    'ward': 'Метод Уорда',
    'complete': 'Метод максимального расстояния',
    'single': 'Метод минимального расстояния',
    'average': 'Метод среднего расстояния'
}

metrics_type_dict = {
    'euclidean': type_metric.EUCLIDEAN,
    'manhattan': type_metric.MANHATTAN
}

def gen_one_class(mean):
    ro = np.random.uniform(ro_min, ro_max)
    cov = np.array([
        [1.0, ro],
        [ro, 1.0]
    ])

    return np.random.multivariate_normal(mean, cov, size=Ni)

def gen_samples():
    means = means_all[:M]
    X_list = []
    true_labels_list = []

    for m in range(M):
        Xm = gen_one_class(means[m])
        X_list.append(Xm)
        true_labels_list.append([m] * Ni)
```

```
    X = np.vstack(X_list)
    true_labels = np.hstack(true_labels_list)
    return X, true_labels, means

def get_2d_coords(array):
    return array[:, 0], array[:, 1]

def match_cluster_labels(actual_labels, predicted_labels):
    cm = confusion_matrix(actual_labels, predicted_labels)

    row_ind, col_ind = linear_sum_assignment(-cm)
    aligned_labels = np.zeros_like(predicted_labels)
    for true_class, cluster in zip(row_ind, col_ind):
        aligned_labels[predicted_labels == cluster] = true_class

    return aligned_labels

def hierarchical_clustering(X, metric='euclidean', linkage='ward'):
    model = AgglomerativeClustering(n_clusters=M, metric=metric, linkage=linkage)
    labels = model.fit_predict(X)

    return labels, model.children_

def build_linkage_tree(children):
    distance = np.arange(children.shape[0])
    observations_count = np.arange(2, children.shape[0] + 2)
    linkage_tree = np.column_stack([children, distance,
observations_count]).astype(float)
    return linkage_tree

def visualize_hierarchical_clustering(X, actual_labels, metric='euclidean',
linkage='ward'):
    predicted_labels, children = hierarchical_clustering(X, metric, linkage)
    aligned_labels = match_cluster_labels(actual_labels, predicted_labels)
    centroids = np.vstack([X[aligned_labels == i].mean(axis=0) for i in
range(M)])
    misclassified = X[actual_labels != aligned_labels]
    error_rate = len(misclassified) / len(aligned_labels)

    print('## Иерархическая кластеризация ##')
    print(f'Используемая метрика: {metric_names_dict[metric]}')
    print(f'Метод объединения: {metric_names_dict[linkage]}')
    print(f'Доля ошибочной классификации: {error_rate:.3f}')
    print()

    fig, axes = plt.subplots(1, 2, figsize=(14, 5), gridspec_kw={'width_ratios':
[2, 3]})

    # Изменены цвета палитры и маркеры
    ax_cluster = axes[0]
```

```
    scatter = ax_cluster.scatter(*get_2d_coords(X), c=aligned_labels,
cmap='tab20b', s=40, alpha=0.8)
    ax_cluster.scatter(*get_2d_coords(centroids), c='darkred', s=150, marker='D',
                       edgecolor='gold', linewidth=2, label='Центроиды')
    ax_cluster.scatter(*get_2d_coords(misclassified), s=120, facecolors='none',
                       edgecolors='navy', linewidth=2.5, label='Ошибки')
    ax_cluster.grid(True, linestyle='--', alpha=0.4)
    ax_cluster.legend(loc='upper right')
    ax_cluster.set_title(f'Кластеризация: {metric_names_dict[metric]},
{metric_names_dict[linkage]}')
    ax_cluster.set_xlabel('X координата')
    ax_cluster.set_ylabel('Y координата')

    # Дендрограмма с изменёнными параметрами
    ax_dendro = axes[1]
    linkage_tree = build_linkage_tree(children)
    sch.dendrogram(linkage_tree, no_labels=True, color_threshold=0.0,
                   ax=ax_dendro, above_threshold_color='maroon')
    ax_dendro.set_title('Дендрограмма кластеров')
    ax_dendro.set_xlabel('Объекты')
    ax_dendro.set_ylabel('Расстояние')

    plt.tight_layout()
    plt.show()

def kmeans_clustering(X, metric='euclidean'):
    initial_centers = random_center_initializer(X, M,
random_state=random_state).initialize()
    model = pcc_kmeans(X, initial_centers,
                       metric=distance_metric(metrics_type_dict[metric]),
                       random_state=random_state)
    model.process()

    clusters = model.get_clusters()
    centers = np.array(model.get_centers())

    encoding = model.get_cluster_encoding()
    encoder = cluster_encoder(encoding, clusters, X)
    result_labels =
encoder.set_encoding(type_encoding.CLUSTER_INDEX_LABELING).get_clusters()

    return np.array([result_labels]).reshape(-1), centers

def visualize_kmeans_clustering(X, actual_labels, metric='euclidean'):
    predicted_labels, centers = kmeans_clustering(X, metric)
    aligned_labels = match_cluster_labels(actual_labels, predicted_labels)
    misclassified = X[actual_labels != aligned_labels]
    error_rate = len(misclassified) / len(aligned_labels)

    print('## K-means кластеризация ##')
    print(f'Используемая метрика: {metric_names_dict[metric]}')
```

```
    print(f'Доля ошибочной классификации: {error_rate:.3f}')
    print()

    plt.figure(figsize=(10, 7))

    # Изменены цвета и маркеры для K-means
    plt.scatter(*get_2d_coords(X), c=aligned_labels, cmap='Set3', s=45,
alpha=0.9)
    plt.scatter(*get_2d_coords(centers), c='darkgreen', s=180, marker='X',
                edgecolor='orange', linewidth=2.5, label='Центры кластеров')
    plt.scatter(*get_2d_coords(misclassified), s=100, facecolors='none',
                edgecolors='purple', linewidth=3, label='Ошибочные точки')

    plt.grid(True, linestyle=':', alpha=0.5)
    plt.legend(loc='upper right')
    plt.title(f'K-means кластеризация: {metric_names_dict[metric]}')
    plt.xlabel('X координата')
    plt.ylabel('Y координата')
    plt.show()

# Генерация данных
X, true_labels, true_centers = gen_samples()

# Визуализация исходных данных с изменёнными параметрами
plt.figure(figsize=(9, 6))
plt.scatter(*get_2d_coords(X), c=true_labels, cmap='Paired', s=50, alpha=0.9)
plt.scatter(*get_2d_coords(true_centers), color='crimson', s=200, marker='P',
            edgecolor='yellow', linewidth=2.5, label='Истинные центры')
plt.grid(True, linestyle='--', alpha=0.4)
plt.legend(loc='lower right')
plt.title('Исходные данные с истинной кластеризацией')
plt.xlabel('X координата')
plt.ylabel('Y координата')
plt.show()

# Тестирование с Манхэттенской метрикой
print('=' * 80)
print('АНАЛИЗ С ИСПОЛЬЗОВАНИЕМ МАНХЭТТЕНСКОЙ МЕТРИКИ')
print('=' * 80)

linkage_methods = ['ward', 'complete', 'average', 'single']
current_metric = 'manhattan'

# Метод Уорда работает только с Евклидовой метрикой
for method in linkage_methods[1:]:
    visualize_hierarchical_clustering(X, true_labels, metric=current_metric,
linkage=method)
    print('-' * 80)

visualize_kmeans_clustering(X, true_labels, metric=current_metric)
```

```python
# Тестирование с Евклидовой метрикой
print('=' * 80)
print('АНАЛИЗ С ИСПОЛЬЗОВАНИЕМ ЕВКЛИДОВОЙ МЕТРИКИ')
print('=' * 80)

current_metric = 'euclidean'

# Метод ближайшего соседа показывает плохие результаты
for method in linkage_methods[:-1]:
    visualize_hierarchical_clustering(X, true_labels, metric=current_metric,
linkage=method)
    print('-' * 80)

visualize_kmeans_clustering(X, true_labels, metric=current_metric)
```

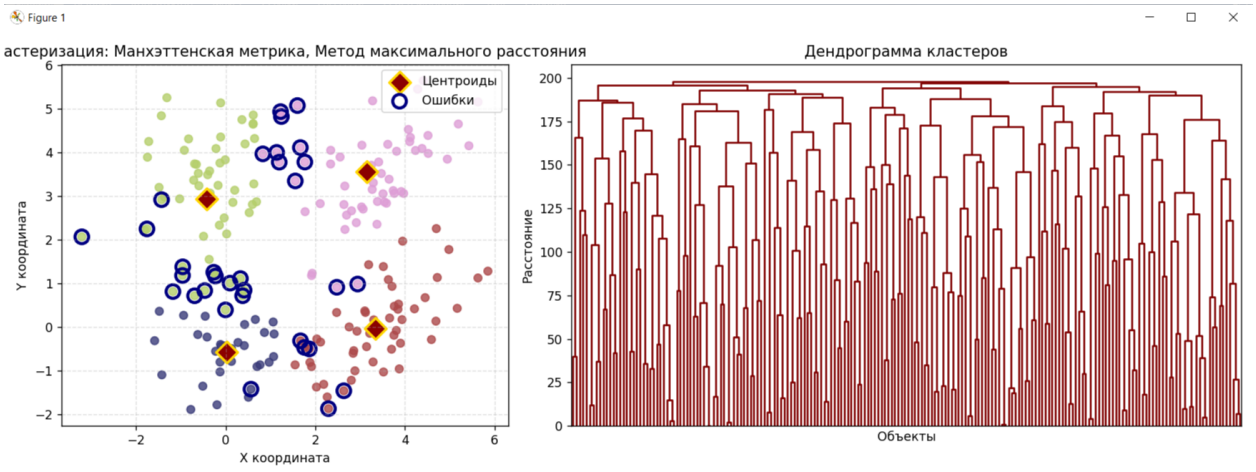


Для иерархического алгоритма использовалось несколько типов расстояний (linkage).

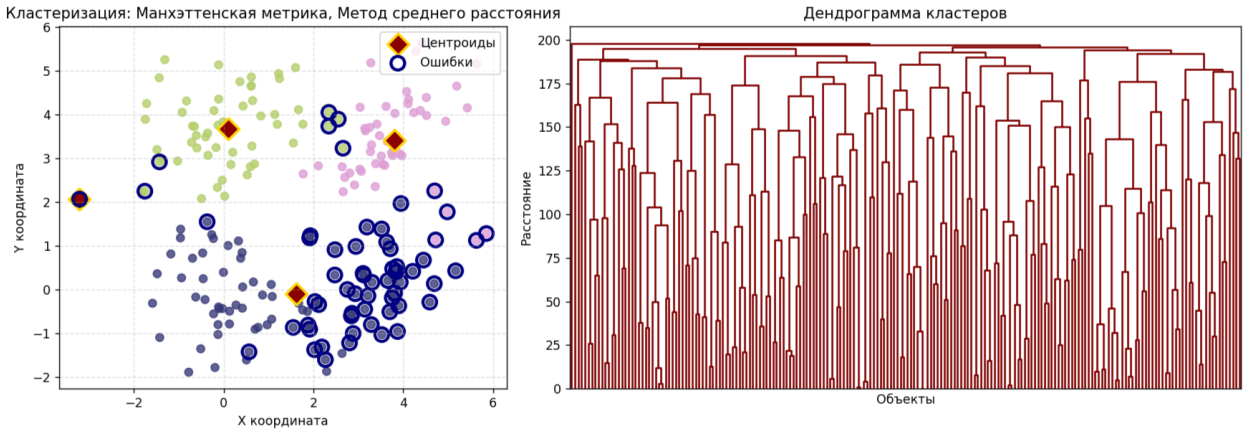
Абсолютная (Манхэттенская) метрика

Иерархический алгоритм

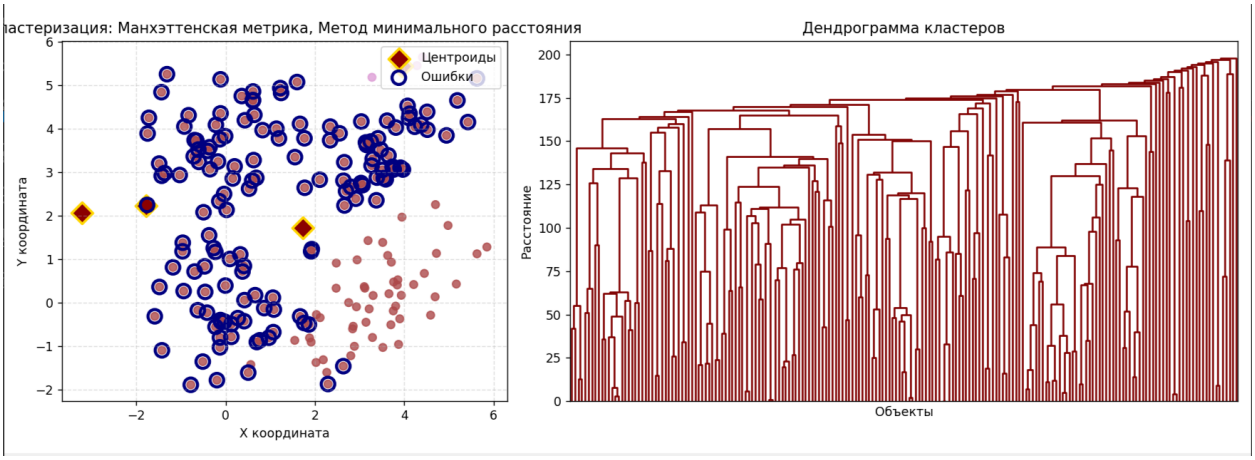
```
## Иерархическая кластеризация ##
Используемая метрика: Манхэттенская метрика
Метод объединения: Метод максимального расстояния
Доля ошибочной классификации: 0.160
```




```
## Иерархическая кластеризация ##
Используемая метрика: Манхэттенская метрика
Метод объединения: Метод среднего расстояния
Доля ошибочной классификации: 0.300
```

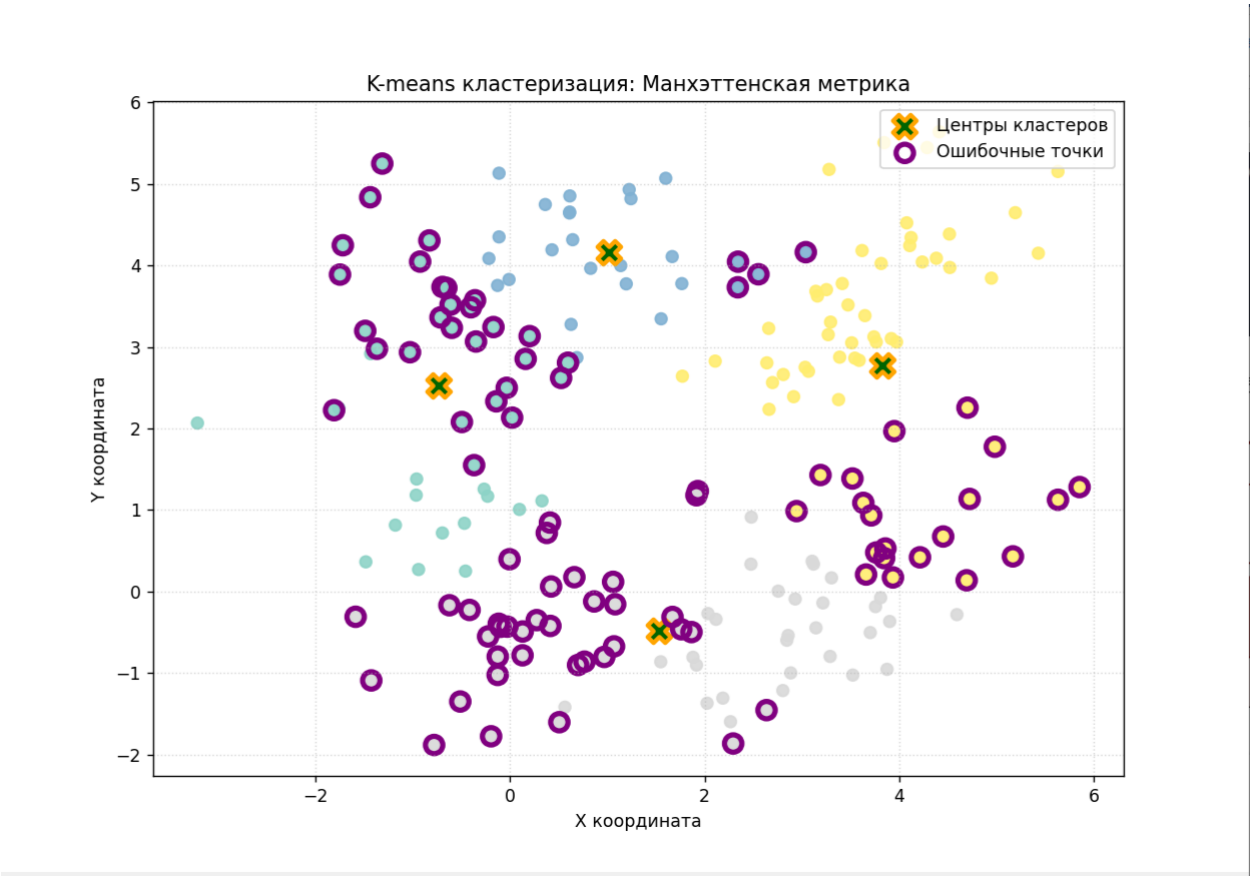


```
## Иерархическая кластеризация ##
Используемая метрика: Манхэттенская метрика
Метод объединения: Метод минимального расстояния
Доля ошибочной классификации: 0.720
```



K-means

```
## K-means кластеризация ##
Используемая метрика: Манхэттенская метрика
Доля ошибочной классификации: 0.445
```

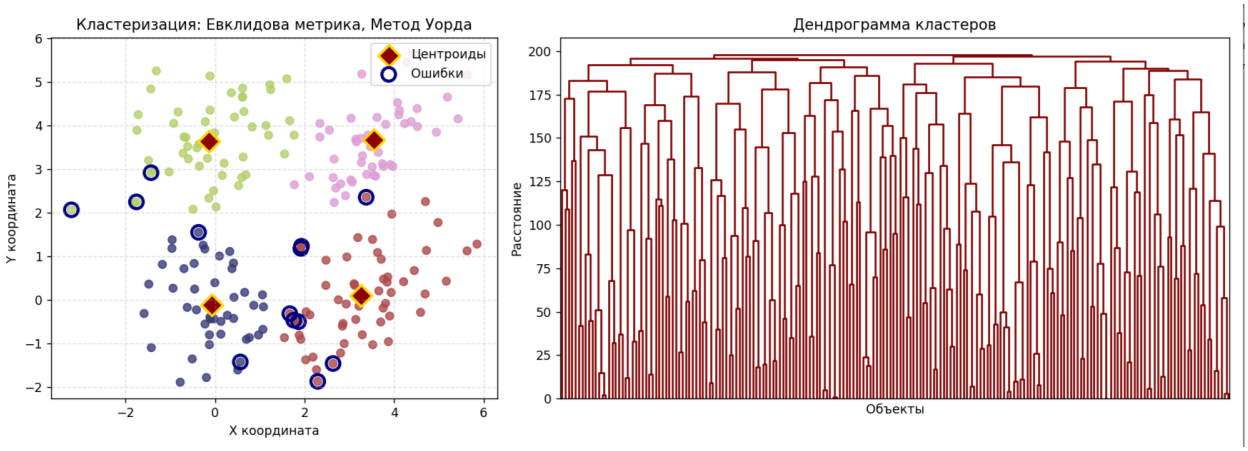


Лучше всего себя показал иерархический алгоритм с linkage – расстояние дальнего соседа.

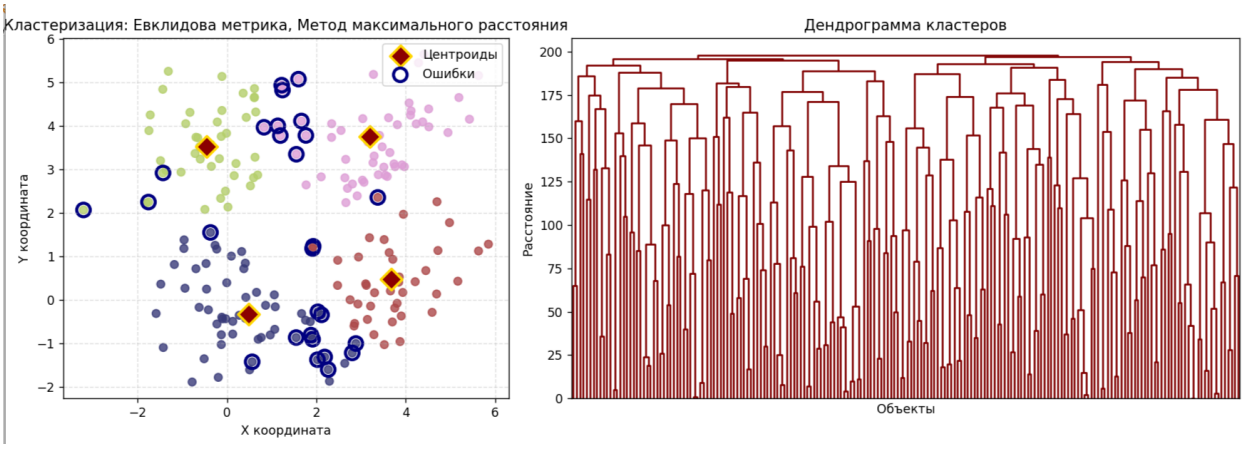
Евклидова метрика

Иерархический алгоритм

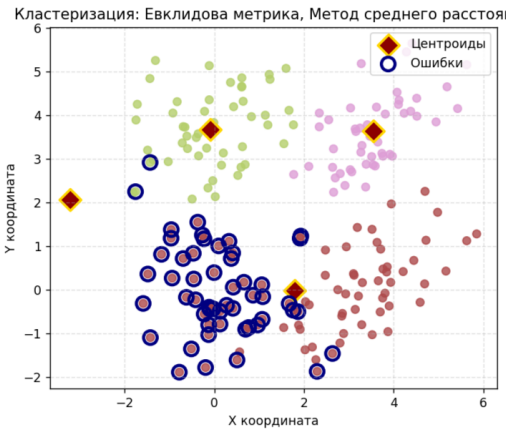
```
## Иерархическая кластеризация ##
Используемая метрика: Евклидова метрика
Метод объединения: Метод Уорда
Доля ошибочной классификации: 0.065
```

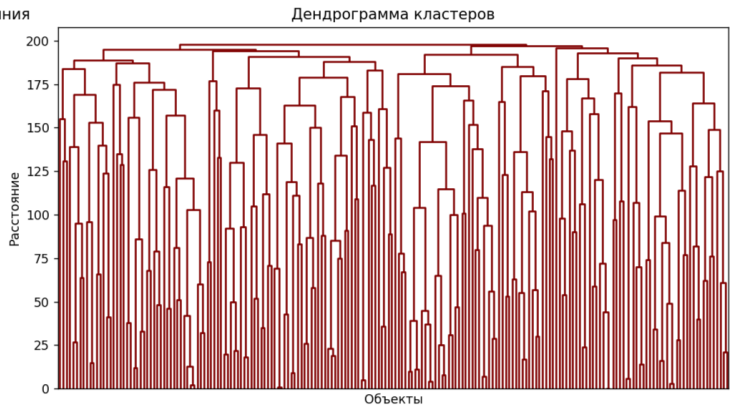



```
## Иерархическая кластеризация ##
Используемая метрика: Евклидова метрика
Метод объединения: Метод максимального расстояния
Доля ошибочной классификации: 0.135
```



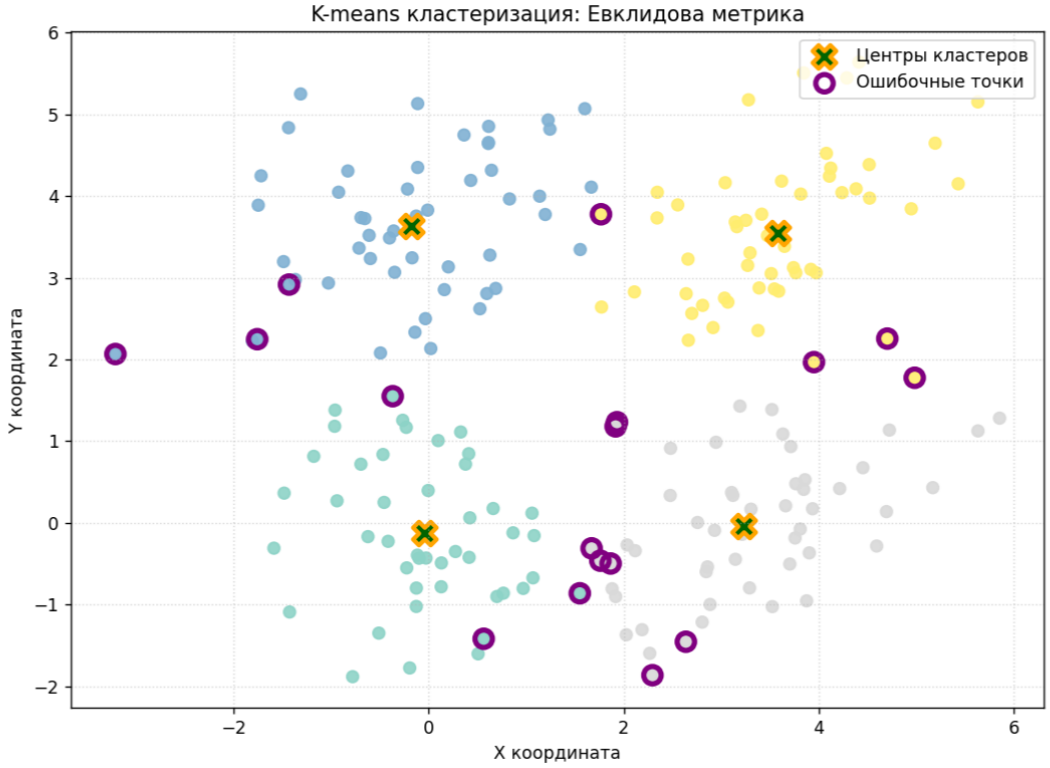
```
## Иерархическая кластеризация ##
Используемая метрика: Евклидова метрика
Метод объединения: Метод среднего расстояния
Доля ошибочной классификации: 0.260
```





K-means

```
## K-means кластеризация ##
Используемая метрика: Евклидова метрика
Доля ошибочной классификации: 0.085
```



Лучше всего себя показал иерархический алгоритм с методом расстояние Уорда. Он оказался лучше всех результатов абсолютной метрики. Кроме того, k-means с евклидовой метрикой оказался также лучше всех результатов абсолютной метрики.

Контрольные вопросы

1. *Что такое дендрограмма?*

Дендрограмма представляет собой древовидную диаграмму, визуализирующую процесс иерархической кластеризации данных. Она показывает последовательность объединения или разделения кластеров с указанием расстояний между ними, где высота ветвей отражает уровень сходства.

2. *Какой критерий минимизируется в алгоритме k-средних?*

Алгоритм k-средних минимизирует сумму квадратов расстояний от точек каждого кластера до его центра (среднего арифметического). Эта величина известна как внутрикластерная инерция или функция потерь.

Выводы

В результате исследования сравнены два основных метода кластеризации: иерархический алгоритм и алгоритм k-средних. Оценено влияние выбора метрики расстояния и способа связывания кластеров на качество разбиения.

При использовании абсолютной (манхэттенской) метрики иерархический алгоритм с расстоянием дальнего соседа показал наилучший результат с вероятностью ошибки классификации 0.16, тогда как расстояние ближнего соседа оказалось неприменимо с ошибкой 0.72. Алгоритм k-means на той же метрике дал вероятность ошибки 0.445.

При использовании евклидовой метрики иерархический алгоритм с расстоянием Уорда продемонстрировал результаты с вероятностью ошибки 0.065, что значительно лучше результатов манхэттенской метрики. Алгоритм k-means с евклидовой метрикой также улучшил свою производительность до 0.085. Выявлено, что выбор метрики расстояния критично влияет на качество кластеризации, а евклидова метрика более эффективна для данного набора данных. Результаты подтверждают необходимость предварительного выбора функции расстояния и параметров связывания для получения надёжной кластеризации.