Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчёт по лабораторной работе №9 по дисциплине "Методы машинного обучения"

Тема Кластерный анализ

Студент Варламова Е. А.

Группа ИУ7-23М

Оценка (баллы) _____

Преподаватели Солодовников Владимир Игоревич

Москва — 2024 г.

СОДЕРЖАНИЕ

1 | Теоретическая часть

В 1936 году Рональд Фишер опубликовал статью, в которой он представил набор данных об ирисах, состоящий из трех различных видов: Iris setosa, Iris versicolor и Iris virginica. Для каждого вида ириса были измерены четыре характеристики: длина чашелистика, ширина чашелистика, длина лепестка и ширина лепестка.

Фишер использовал этот набор данных для демонстрации метода дискриминантного анализа, который позволяет разделить объекты на классы на основе их признаков. Он показал, что можно эффективно классифицировать ирисы по видам, используя комбинацию этих четырех признаков.

Ирисы Фишера стали одним из самых популярных наборов данных в машинном обучении и статистике. Они часто используются для тестирования алгоритмов классификации и кластеризации, а также для демонстрации методов визуализации данных.

Целью данной лабораторной работы является применение алгоритмов кластеризации для решения задачи кластерищации ирисов Фишера. Для этого необходимо решить следующие задачи:

— описать методы кластеризации;

— привести особенности реализации ПО, решающего поставленную задачу;

— провести сравнение результатов работы алгоритмов.

1.1 Постановка задачи

— Провести описательный статистический анализ данных. Рассчитать оценки мат. ожидания, доверительный интервал, среднеквадратичное отклонение. Построить графики и гистограммы.

— Построить корреляционную матрицу. Проверить значимость корреляции.

— Провести кластерный анализ данных. Использовать не менее двух алгоритмов кластеризации (например: иерархический, K-средних, DBSCAN). Варьировать различные значения гиперпараметров и тип расстояний.

— Оценить работу алгоритмов с использованием внешних и внутренних мер оценки качества. Определить оптимальное количество кластеров и их структуру.

1.2 K-means

Цель алгоритма: разделить данные на k кластеров при условии наличия набора данных X с n объектами и p признаками.

Шаги алгоритма:

1. Инициализация: выбрать k случайных точек из X в качестве начальных центроидов $c_1, c_2, ..., c_k$.

2. Назначение: для каждого объекта x_i в X присвоить его ближайшему центроиду c_j, то есть (для метрики Евклидово расстояние):

$$j = argmin_j ||x_i - c_j||^2 \tag{1.1}$$

3. Обновление: для каждого кластера вычислить новый центроид как среднее всех точек в этом кластере.

4. Повторение: повторять шаги 2 и 3, пока центроиды не перестанут существенно меняться или не будет достигнуто максимальное число итераций.

5. Результат: присвоить каждую точку x_i кластеру, которому принадлежит ее ближайший центроид.

1.3 DBSCAN

Цель алгоритма: разделить данные на кластеры, разделяя плотные области от разреженных при наличии набора данных X с n объектами и p признаками, радиуса окрестности ϵ и минимальном количестве точек в кластере $minPts$.

Шаги алгоритма:

1. Инициализация: пометить все точки как не посещенные.

2. Посещение точек: для каждой непосещенной точки x_i найти все точки в окрестности ϵ от x_i. Если количество точек в окрестности больше или равно $minPts$, создать новый кластер. Иначе пометить x_i как шум.

3. Расширение кластеров: для каждой точки в созданном кластере посетить все ее непосещенные соседи в окрестности ϵ. Если количество соседей больше или равно $minPts$, добавить их в кластер.

4. Повторять шаги 2 и 3, пока все точки не будут посещены.

2 | Практическая часть

2.1 Выбор средств разработки

В качестве языка программирования был использован язык Python, поскольку этот язык кроссплатформенный и для него разработано огромное количество библиотек и модулей, решающих разнообразные задачи.

В частности, имеются библиотеки, включающие в себя алгоритмы кластеризации, в частности sklearn [1].

Для создания графиков была выбрана библиотека matplotlib [2], доступная на языке Python, так как она предоставляет удобный интерфейс для работы с данными и их визуализации.

2.2 ПО

В листинге 2.1 представлен код, решающий задачу кластеризации двумя алгоритмами.

Листинг 2.1: Код кластеризации

```
import pandas as pd
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans, DBSCAN
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import umap
import seaborn as sns
import imageio
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score,
    f1_score, confusion_matrix
from sklearn.metrics import mean_absolute_error, mean_squared_error,
    r2_score, adjusted_rand_score
n_cluster = 4
cols = [ "sepal_length","sepal_width","petal_length","petal_width"]
```

```
17 def draw_description(data):
18     confidence_level = 0.95
19
20     for col in data.columns:
21         column_data = data[col]
22
23         mean = np.mean(column_data)
24
25         confidence_interval = stats.t.interval(0.95, len(column_data)-1, loc
               =mean, scale=stats.sem(column_data))
26
27         std_dev = np.std(column_data)
28
29         plt.hist(column_data, bins='auto', alpha=0.7, color='skyblue',
               edgecolor='black', linewidth=1.2)
30         plt.axvline(mean, color='r', linestyle='dashed', linewidth=2, label=
               f'      .                  : {mean:.2f}')
31         plt.axvline(mean - std_dev, color='g', linestyle='dashed', linewidth
               =2, label=f'      .                  - std: {mean - std_dev:.2f
               }')
32         plt.axvline(mean + std_dev, color='g', linestyle='dashed', linewidth
               =2, label=f'      .                  + std: {mean + std_dev:.2f
               }')
33         plt.axvspan(confidence_interval[0], confidence_interval[1], color='
               gray', alpha=0.3, label='
                                (95%)')
34         plt.xlabel('                  ')
35         plt.ylabel('              ')
36         plt.title('                                                          ')
37         plt.legend()
38         plt.savefig(f"{col}.png")
39         plt.clf()
40 def draw_corr(data):
41     corr=data.corr()
42     plt.figure(figsize=(30,20))
43     sns.heatmap(corr, annot=True, cmap="Reds")
44     plt.title('                    ', fontsize=20)
45     plt.savefig("correlation.png")
46     plt.clf()
47
48
49 def read_data(fn):
50     return pd.read_csv(fn)
51
52 def preprocess_data(df):
53     return df[cols].dropna(how='any')
54
55 def count_elbow(df):
56     inertia = []
57     n = 15
```

```
    for k in range(1, n):
        kmeans = KMeans(n_clusters=k)
        kmeans.fit(df)
        inertia.append(kmeans.inertia_)

    plt.figure(figsize=(8, 6))
    plt.plot(range(1, n), inertia, marker='o')
    plt.xlabel('Number of clusters')
    plt.ylabel('Inertia')
    plt.title('Elbow Method for Optimal K')
    plt.savefig("elbow.png")
    plt.clf()

def count_k_means(df):
    kmeans = KMeans(n_cluster)
    kmeans.fit(df)
    return kmeans.labels_
def count_dbscan(df):
    dbscan = DBSCAN(eps=4.2, min_samples=5)
    return dbscan.fit_predict(df)
def count_pca(df):
    pca = PCA(n_components=2)
    return pca.fit_transform(df)
def count_umap(df):
    return umap.UMAP(n_components=2).fit_transform(df)

def plot(title, labels, df_resize, dir_name = "./"):
    plt.figure(figsize=(12, 6))
    sns.scatterplot(x=df_resize[:, 0], y=df_resize[:, 1], hue=labels,
        palette='viridis')
    plt.title(title)
    plt.legend()
    plt.savefig(dir_name + title + ".png")
    plt.clf()

def visual(df, f_cluster, f_resize, title, dir_name = "./"):
    df_resize = f_resize(df)
    labels = f_cluster(df)

    plot(title, labels, df_resize, dir_name = "./")

def vary_metric_dbscan(df, answers):
    metric = ["cosine", "euclidean", "l1", "l2", "manhattan"]
    scores = []
    for m in metric:
        dbscan = DBSCAN(eps=4.2, min_samples=5, metric = m)
        labels = dbscan.fit_predict(df)
        df_resize = count_umap(df)
        plot(f"metric_{m}", labels, df_resize, dir_name = "
            research_dbscan_metric/")
```

```
106         scores.append(adjusted_rand_score(answers, labels))
107
108     plt.figure(figsize=(12, 6))
109     plt.plot(metric, scores)
110     plt.title("scores")
111     plt.legend()
112     plt.savefig("research_dbscan_metric/scores.png")
113     plt.clf()
114 def vary_eps_dbscan(df, answers):
115     epss = np.linspace(2, 6, 21)
116     scores = []
117     for eps in epss:
118         dbscan = DBSCAN(eps=eps, min_samples=5)
119         labels = dbscan.fit_predict(df)
120         df_resize = count_umap(df)
121         eps  = round(eps * 10)
122         plot(f"metric_{eps}", labels, df_resize, dir_name = "
               research_dbscan_eps/")
123         scores.append(adjusted_rand_score(answers, labels))
124
125     plt.figure(figsize=(12, 6))
126     plt.plot(epss, scores)
127     plt.title("scores")
128     plt.legend()
129     plt.savefig("research_dbscan_eps/scores.png")
130 def vary_min_samples_dbscan(df, answers):
131     scores = []
132     x = []
133     for s in range(5, 50, 5): # 150
134         dbscan = DBSCAN(eps=4.2, min_samples=s)
135         labels = dbscan.fit_predict(df)
136         df_resize = count_umap(df)
137         plot(f"samples_{s}", labels, df_resize, dir_name = "
               research_dbscan_samples/")
138         scores.append(adjusted_rand_score(answers, labels))
139         x.append(s)
140
141     plt.figure(figsize=(12, 6))
142     plt.plot(x, scores)
143     plt.title("scores")
144     plt.legend()
145     plt.savefig("research_dbscan_samples/scores.png")
146 def vary_num_clusters_kmeans(df, answers):
147     scores = []
148     x = []
149     for cluster in range(2, 11, 1):
150         kmeans = KMeans(cluster)
151         kmeans.fit(df)
152         labels = kmeans.labels_
153         df_resize = count_umap(df)
```

```
154         plot(f"clusters_{cluster}", labels, df_resize, dir_name = "
                research_kmeans_clusters/")
155         scores.append(adjusted_rand_score(answers, labels))
156         x.append(cluster)
157 
158     plt.figure(figsize=(12, 6))
159     plt.plot(x, scores)
160     plt.title("scores")
161     plt.legend()
162     plt.savefig("research_kmeans_clusters/scores.png")
163 def vary_centroid_algo_kmeans(df, answers):
164     scores = []
165     metrics = ['random', 'k-means++']
166     for metric in metrics:
167         kmeans = KMeans(n_cluster, init=metric)
168         kmeans.fit(df)
169         labels = kmeans.labels_
170         df_resize = count_umap(df)
171         plot(f"algo_{metric}", labels, df_resize, dir_name = "
                research_kmeans_algo/")
172         scores.append(adjusted_rand_score(answers, labels))
173 
174     plt.figure(figsize=(12, 6))
175     plt.plot(metrics, scores)
176     plt.title("scores")
177     plt.legend()
178     plt.savefig("research_kmeans_algo/scores.png")
179 def count_accuracy(df, answers):
180     kmeans = KMeans(4, init = 'k-means++')
181     kmeans.fit(df)
182     labels_kmeans = kmeans.labels_
183     print("kmeans", adjusted_rand_score(answers, labels_kmeans))
184 
185     dbscan = DBSCAN(eps=4.2, min_samples=5)
186     labels_dbscan = dbscan.fit_predict(df)
187     print("dbscan", adjusted_rand_score(answers, labels_dbscan))
```

2.3 Описательный статистический анализ данных

Был проведён описательный статистический анализ данных: рассчитаны оценки мат. ожидания, доверительные интервалы, среднеквадратичное отклонение. Построены графики и гистограммы:

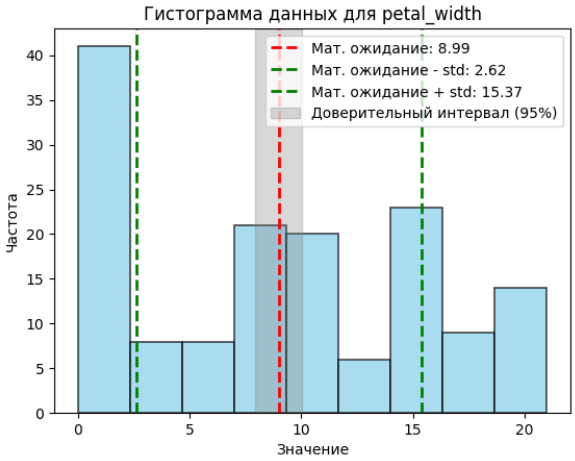

Рис. 2.1: Анализ pental_width

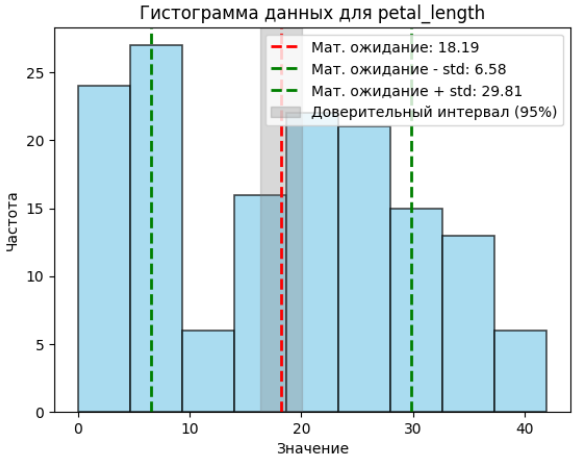

Рис. 2.2: Анализ pental_length

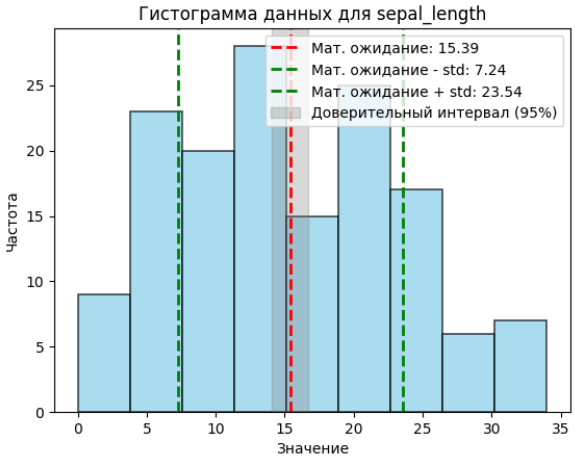

Рис. 2.3: Анализ sepal_length

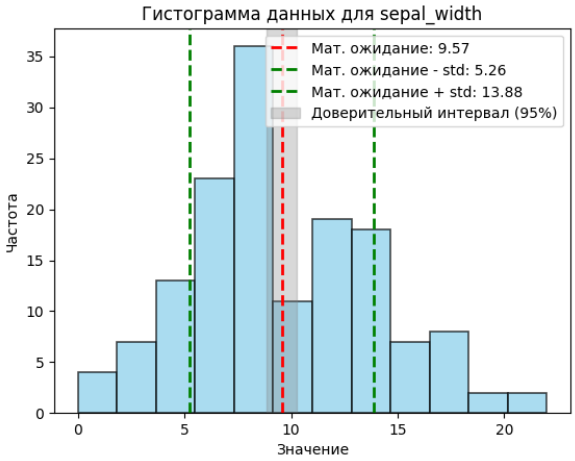

Рис. 2.4: Анализ sepal_width

Была построена матрица корреляций:

Видно, что в большинстве классов наблюдается слабая корреляция между признаками.

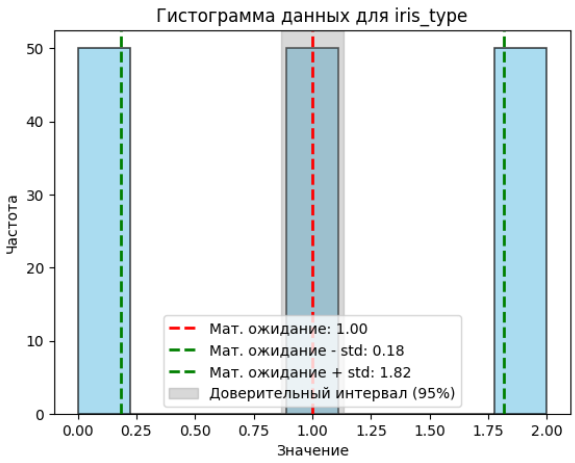

Рис. 2.5: Анализ iris_type

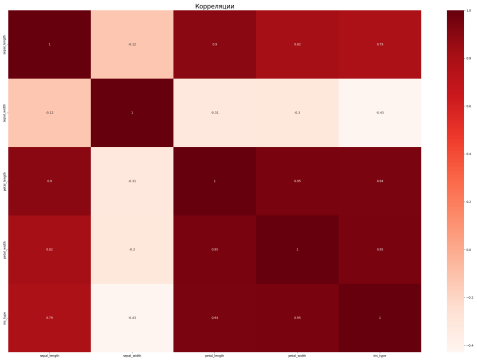

Рис. 2.6: Корреляционная матрица

2.4 Кластерный анализ данных

Для кластеризации были использованы 2 алгоритма и для каждого из них были проварьированы основные параметры:

— K-средних (параметры – количество кластеров, алгоритм выбора центроида);

— DBSCAN (параметры – окрестность соседства, минимальное количество в кластере, алгоритм расчёта расстояния).

В качестве метрики качества кластеризации использовалась разница разбиения на кластеры с помощью алгоритма и исходного деления на кластеры в датасете.

2.4.1 DBSCAN: окрестность соседства

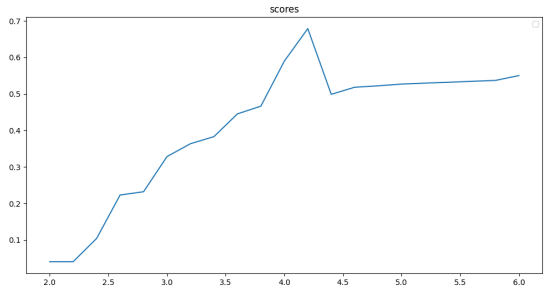

Рис. 2.7: DBSCAN: окрестность соседства

Видно, что лучшее значение 4.2.

2.4.2 DBSCAN: минимальное количество в кластере

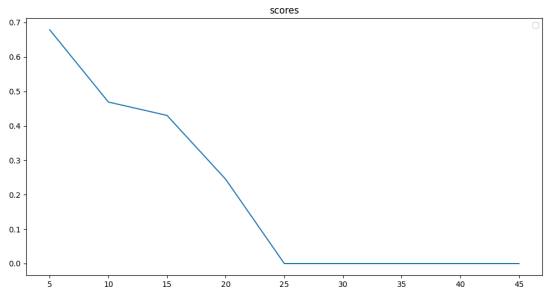

Рис. 2.8: DBSCAN: минимальное количество в кластере

Видно, что лучшее значение 5.

2.4.3 DBSCAN: алгоритм расчёта расстояния

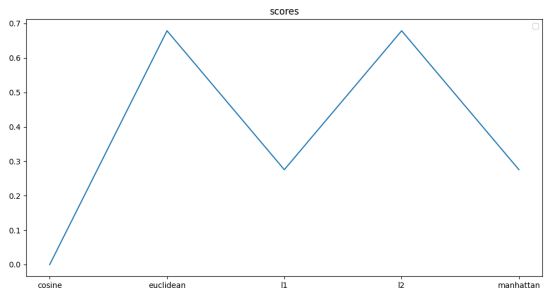

Рис. 2.9: DBSCAN: алгоритм расчёта расстояния

Видно, что лучший алгоритм – евклидово расстояние.

2.4.4 kmeans: количество кластеров

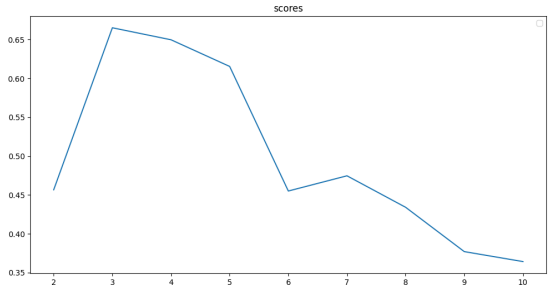

Рис. 2.10: kmeans: количество кластеров

Видно, что лучшее значение – 3. Для определения количества кластеров существует метод локтя. Проверим полученный результат с помощью него.

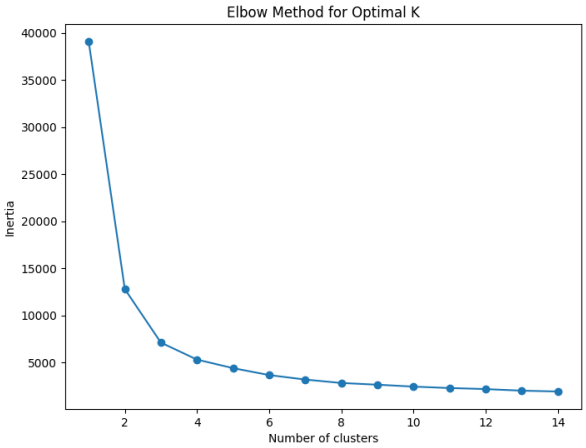

Рис. 2.11: Метод локтя

Видно, что для метода локтя оптимальным является 4-5 кластеров.

2.4.5 kmeans: алгоритм выбора центроида

Видно, что алгоритм выбора центроида не влияет на конечный результат кластеризации.

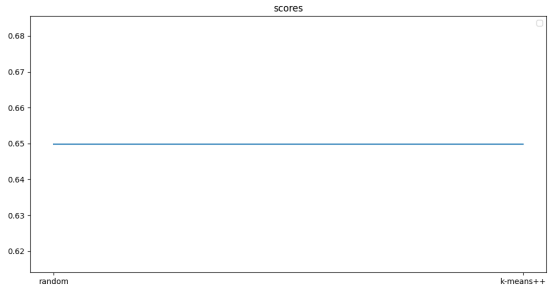

Рис. 2.12: kmeans: алгоритм выбора центроида

2.5 Оценка работы алгоритмов

С помощью определённых оптимальных параметров алгоритмов данные были кластеризованы и визуализированы с помощью разных алгоритмов уменьшения размерности данных, а также вычислены точности кластеризации:

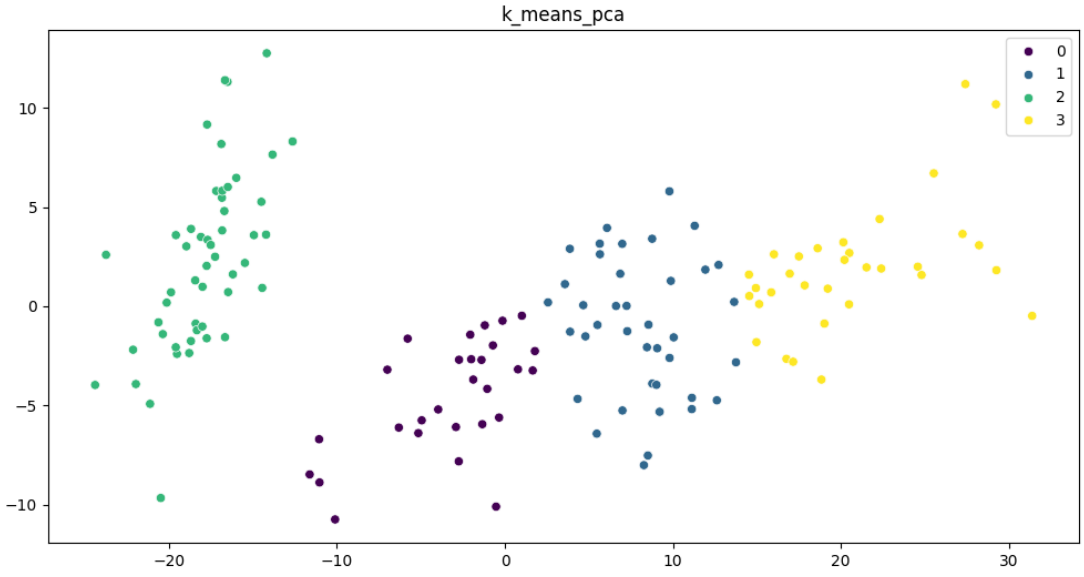

Рис. 2.13: kmeans, pca

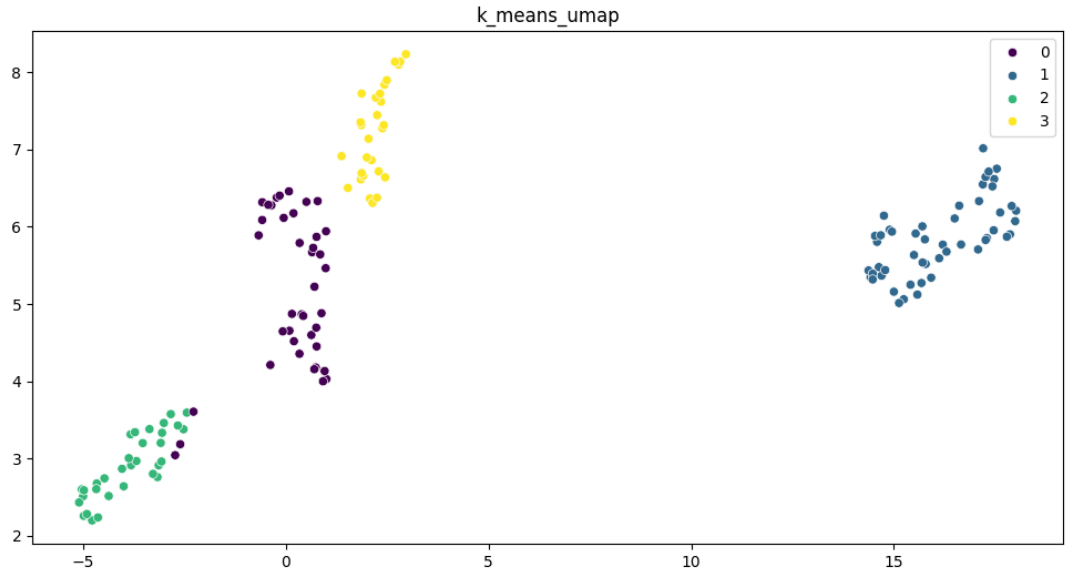

Рис. 2.14: kmeans, UMAP

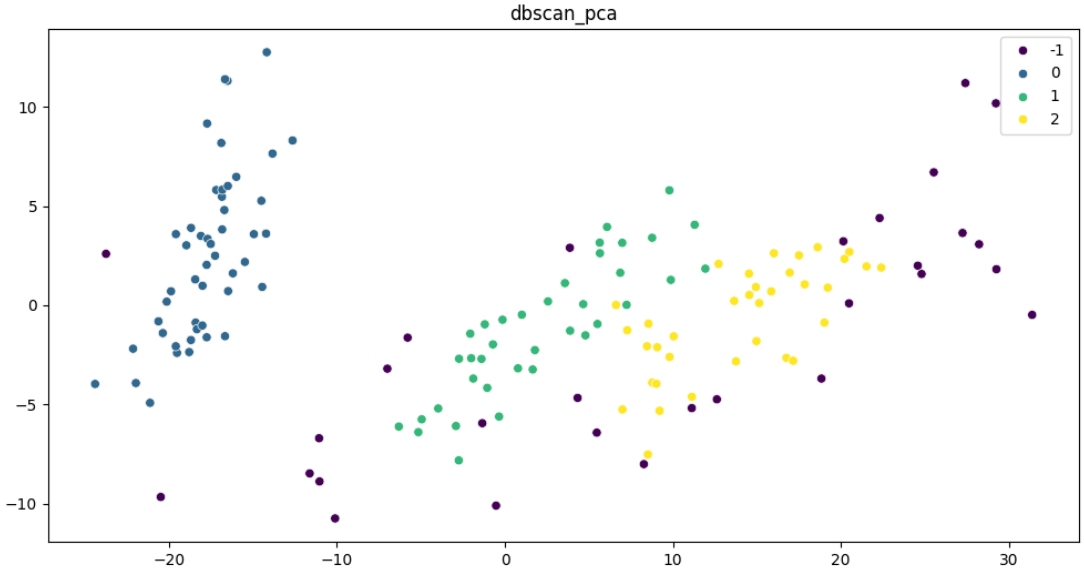

Рис. 2.15: DBSCAN, pca

Точности кластеризации(диапазон: [-0.5 - 1], где 0 – случайная кластеризация, 1 – точная):

— kmeans: 0.65.

— DBSCAN: 0.63.

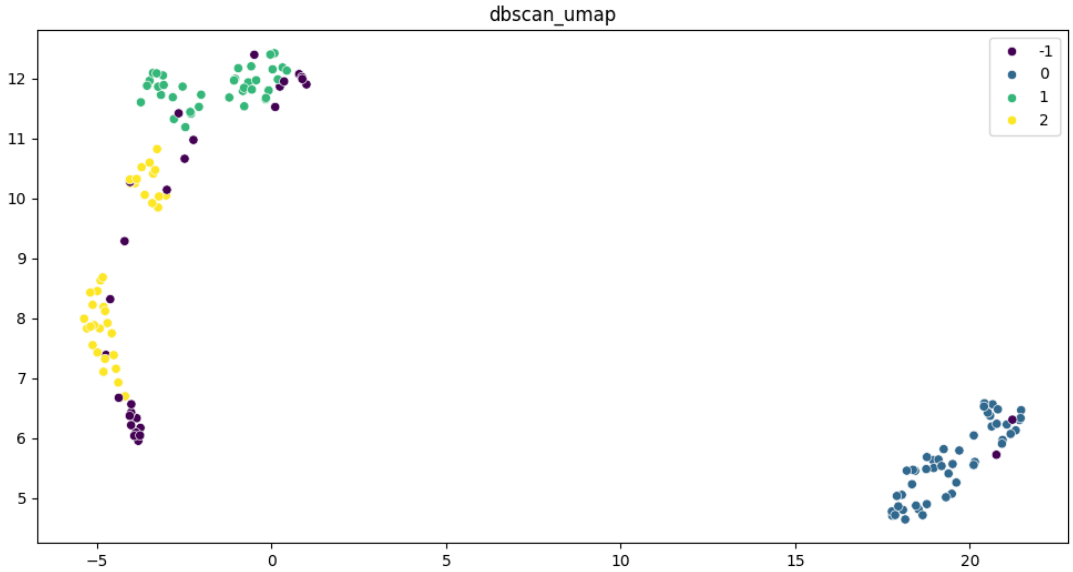

Рис. 2.16: DBSCAN, UMAP

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Scikit-learn: Machine learning in Python / F. Pedregosa [и др.]. — 2011.
2. Библиотека визуализации данных matplotlib [Электронный ресурс]. — Режим доступа: URL: https://matplotlib.org (дата обращения: 13.12.2023).