import seaborn as sns

import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler

from sklearn.impute import SimpleImputer

from sklearn.decomposition import PCA

from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN

from scipy.cluster.hierarchy import dendrogram, linkage

import matplotlib.pyplot as plt

import re

from nltk.corpus import stopwords

from nltk.stem import WordNetLemmatizer

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.datasets import load\_wine

from sklearn.datasets import load\_iris

from sklearn.metrics import adjusted\_rand\_score, adjusted\_mutual\_info\_score

from sklearn.datasets import fetch\_20newsgroups

from sklearn.preprocessing import MultiLabelBinarizer

import ast

from sklearn.datasets import make\_moons

Задача 1: Сегментация

Используйте встроенный в библиотеку seaborn датасет "penguins" для задачи кластеризации. Ваша задача - сегментировать пингвинов с использованием K-means.

Попробуйте иерархическую классификацию и DBSCAN на тех же данных и сравните результаты.

def task1\_1():

    penguins = sns.load\_dataset("penguins")

    print(penguins.head())

    print(penguins.isnull().sum())

    imputer = SimpleImputer(strategy="most\_frequent")

    penguins\_cleaned = pd.DataFrame(imputer.fit\_transform(penguins), columns=penguins.columns)

    # Выделение числовых признаков

    numeric\_features = ["bill\_length\_mm", "bill\_depth\_mm", "flipper\_length\_mm", "body\_mass\_g"]

    data = penguins\_cleaned[numeric\_features].astype(float)

    # Масштабирование данных

    scaler = StandardScaler()

    data\_scaled = scaler.fit\_transform(data)

    print(pd.DataFrame(data\_scaled, columns=numeric\_features).head())

    # K-Means кластеризация

    # Метод локтя

    inertia = []

    range\_n\_clusters = range(1, 10)

    for k in range\_n\_clusters:

        kmeans = KMeans(n\_clusters=k, random\_state=42)

        kmeans.fit(data\_scaled)

        inertia.append(kmeans.inertia\_)

    # График метода локтя

    plt.plot(range\_n\_clusters, inertia, marker='o')

    plt.xlabel("Number of Clusters")

    plt.ylabel("Inertia")

    plt.title("Elbow Method for Optimal Clusters")

    plt.show()

    # K-Means с 3 кластерами

    kmeans = KMeans(n\_clusters=3, random\_state=42)

    clusters\_kmeans = kmeans.fit\_predict(data\_scaled)

    penguins\_cleaned["Cluster\_KMeans"] = clusters\_kmeans

    linkage\_matrix = linkage(data\_scaled, method='ward')

    plt.figure(figsize=(10, 5))

    dendrogram(linkage\_matrix)

    plt.title("Dendrogram")

    plt.xlabel("Samples")

    plt.ylabel("Distance")

    plt.show()

    # Агломеративная кластеризация

    agg\_clustering = AgglomerativeClustering(n\_clusters=3)

    clusters\_agg = agg\_clustering.fit\_predict(data\_scaled)

    penguins\_cleaned["Cluster\_Agglomerative"] = clusters\_agg

    # DBSCAN кластеризация

    dbscan = DBSCAN(eps=0.5, min\_samples=5)

    clusters\_dbscan = dbscan.fit\_predict(data\_scaled)

    penguins\_cleaned["Cluster\_DBSCAN"] = clusters\_dbscan

    pca = PCA(n\_components=2)

    data\_pca = pca.fit\_transform(data\_scaled)

    # Визуализация кластеров для K-Means

    plt.scatter(data\_pca[:, 0], data\_pca[:, 1], c=clusters\_kmeans, cmap='viridis')

    plt.title("K-Means Clustering")

    plt.show()

    # Визуализация кластеров для агломеративной кластеризации

    plt.scatter(data\_pca[:, 0], data\_pca[:, 1], c=clusters\_agg, cmap='viridis')

    plt.title("Agglomerative Clustering")

    plt.show()

    # Визуализация кластеров для DBSCAN

    plt.scatter(data\_pca[:, 0], data\_pca[:, 1], c=clusters\_dbscan, cmap='viridis')

    plt.title("DBSCAN Clustering")

    plt.show()

    # Сравнение результатов кластеризации

    comparison\_kmeans = penguins\_cleaned.groupby("Cluster\_KMeans")[numeric\_features].mean()

    print("Средние значения признаков по кластерам (K-Means):")

    print(comparison\_kmeans)

    comparison\_agg = penguins\_cleaned.groupby("Cluster\_Agglomerative")[numeric\_features].mean()

    print("Средние значения признаков по кластерам (Agglomerative):")

    print(comparison\_agg)

    unique\_clusters\_dbscan = len(set(clusters\_dbscan)) - (1 if -1 in clusters\_dbscan else 0)

    print(f"Количество кластеров, найденных DBSCAN: {unique\_clusters\_dbscan}")

#task1\_1()

species island bill\_length\_mm bill\_depth\_mm flipper\_length\_mm \

0 Adelie Torgersen 39.1 18.7 181.0

1 Adelie Torgersen 39.5 17.4 186.0

2 Adelie Torgersen 40.3 18.0 195.0

3 Adelie Torgersen NaN NaN NaN

4 Adelie Torgersen 36.7 19.3 193.0

body\_mass\_g sex

0 3750.0 Male

1 3800.0 Female

2 3250.0 Female

3 NaN NaN

4 3450.0 Female

species 0

island 0

bill\_length\_mm 2

bill\_depth\_mm 2

flipper\_length\_mm 2

body\_mass\_g 2

sex 11

dtype: int64

bill\_length\_mm bill\_depth\_mm flipper\_length\_mm body\_mass\_g

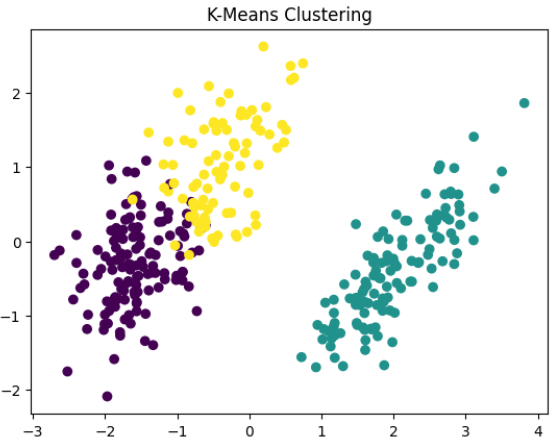
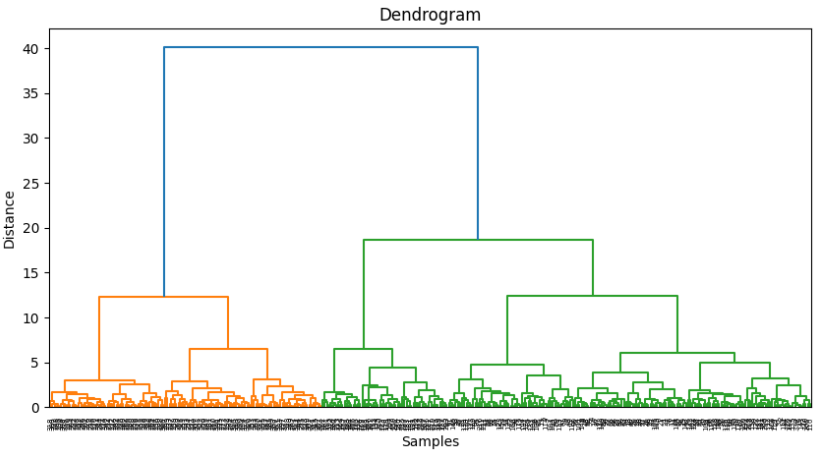
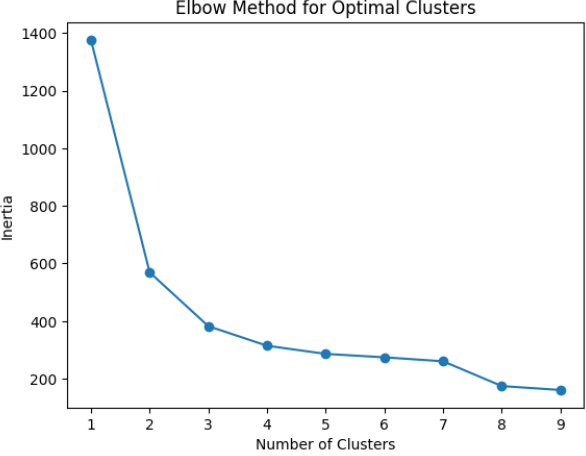
0 -0.883375 0.788176 -1.415471 -0.562452

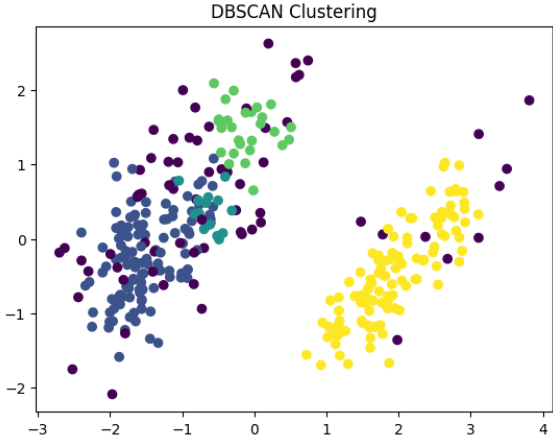
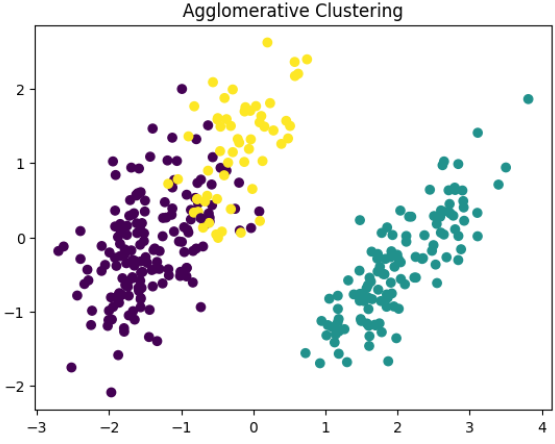
1 -0.809845 0.127001 -1.058960 -0.499877

2 -0.662785 0.432159 -0.417242 -1.188208

3 -0.515725 -0.076437 -0.773752 -0.499877

4 -1.324555 1.093334 -0.559846 -0.937906





Средние значения признаков по кластерам (K-Means):

bill\_length\_mm bill\_depth\_mm flipper\_length\_mm body\_mass\_g

Cluster\_KMeans

0 38.251493 18.09403 188.425373 3587.873134

1 47.504878 14.982114 217.186992 5076.01626

2 47.525287 18.762069 196.896552 3902.011494

Средние значения признаков по кластерам (Agglomerative):

bill\_length\_mm bill\_depth\_mm flipper\_length\_mm \

Cluster\_Agglomerative

0 39.17378 18.271951 189.902439

1 47.504878 14.982114 217.186992

2 49.752632 18.601754 197.105263

body\_mass\_g

Cluster\_Agglomerative

0 3677.286585

1 5076.01626

2 3810.087719

Количество кластеров, найденных DBSCAN: 4

Задача 2: Кластеризация текстовых данных (новостных заголовков)

Вам нужно скачать [News Aggregator Dataset](https://www.kaggle.com/uciml/news-aggregator-dataset) с Kaggle, предварительно обработать текст новостных заголовков и провести кластеризацию с использованием K-Means. Целью этой задачи является группировка похожих новостей в одном кластере.

def task1\_2():

    data = pd.read\_csv("D:\\Desktop\\7 sem\\МиАОД\\lab15-17\\uci-news-aggregator.csv")

    headlines = data["TITLE"]

    print(f"Всего заголовков: {len(headlines)}")

    # Предобработка текста

    stop\_words = set(stopwords.words("english"))

    lemmatizer = WordNetLemmatizer()

    def preprocess\_text(text):

        text = re.sub(r"[^a-zA-Z\s]", "", text)

        text = text.lower()

        words = text.split()

        words = [lemmatizer.lemmatize(word) for word in words if word not in stop\_words]  # Лемматизация и удаление стоп-слов

        return " ".join(words)

    headlines\_cleaned = headlines.apply(preprocess\_text)

    print(headlines\_cleaned.head())

    # Преобразование текста в числовой вид

    # Векторизация с помощью TF-IDF

    tfidf\_vectorizer = TfidfVectorizer(max\_features=1000)  # Оставляем топ-1000 признаков

    tfidf\_matrix = tfidf\_vectorizer.fit\_transform(headlines\_cleaned)

    # Преобразованный массив

    print(tfidf\_matrix.shape)  # Размерность: (число документов, 1000 признаков)

    # K-Means

    # Метод локтя

    inertia = []

    range\_n\_clusters = range(1, 10)

    for k in range\_n\_clusters:

        kmeans = KMeans(n\_clusters=k, random\_state=42)

        kmeans.fit(tfidf\_matrix)

        inertia.append(kmeans.inertia\_)

    # График для метода локтя

    plt.plot(range\_n\_clusters, inertia, marker='o')

    plt.xlabel("Number of Clusters")

    plt.ylabel("Inertia")

    plt.title("Elbow Method for Optimal Clusters")

    plt.show()

    # K-Means с 5 кластерами

    optimal\_clusters = 5

    kmeans = KMeans(n\_clusters=optimal\_clusters, random\_state=42)

    clusters = kmeans.fit\_predict(tfidf\_matrix)

    # Добавление результатов в DataFrame

    data["Cluster"] = clusters

    print(data.groupby("Cluster")["TITLE"].count())

    # Анализ кластеров

    # Вывод ключевых слов для каждого кластера

    terms = tfidf\_vectorizer.get\_feature\_names\_out()

    centroids = kmeans.cluster\_centers\_

    for i in range(optimal\_clusters):

        cluster\_terms = centroids[i].argsort()[-10:][::-1]  # Топ-10 ключевых слов

        print(f"Кластер {i}: {', '.join(terms[t] for t in cluster\_terms)}")

    # Примеры заголовков из каждого кластера

    for i in range(optimal\_clusters):

        print(f"\nКластер {i}:")

        print(data[data["Cluster"] == i]["TITLE"].head(5).values)

    # Визуализация кластеров

    # PCA для уменьшения размерности

    pca = PCA(n\_components=2)

    data\_pca = pca.fit\_transform(tfidf\_matrix.toarray())

    # Визуализация кластеров

    plt.scatter(data\_pca[:, 0], data\_pca[:, 1], c=clusters, cmap='viridis', s=5)

    plt.title("Кластеры новостных заголовков")

    plt.xlabel("PCA Component 1")

    plt.ylabel("PCA Component 2")

    plt.show()

#task1\_2()

Всего заголовков: 422419

0 fed official say weak data caused weather slow...

1 fed charles plosser see high bar change pace t...

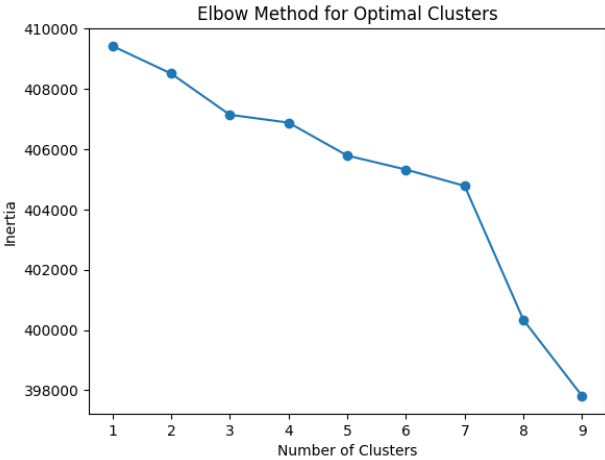
2 u open stock fall fed official hint accelerate...

3 fed risk falling behind curve charles plosser say

4 fed plosser nasty weather curbed job growth

Name: TITLE, dtype: object

(422419, 1000)



Cluster

0 1507

1 7383

2 1658

3 407134

4 4737

Name: TITLE, dtype: int64

Кластер 0: fda, drug, ecigarettes, new, device, first, warns, treatment, cancer, heart

Кластер 1: stock, market, higher, gain, earnings, lower, rise, data, today, european

Кластер 2: obama, president, climate, school, health, museum, pay, play, order, japan

Кластер 3: new, google, say, apple, video, get, star, report, show, may

Кластер 4: amazon, fire, smartphone, phone, prime, hbo, service, streaming, launch, deal

Кластер 0:

['FDA approves electric headband to prevent migraine'

'First device to prevent migraine headaches wins FDA approval'

"'Cefaly' headband-like device to prevent migraines gets FDA nod"

'FDA Approves Medical Device to Prevent Migraine Headaches [VIDEO]'

'US FDA approves first device to prevent migraine headaches']

Кластер 1:

['US open: Stocks fall after Fed official hints at accelerated tapering'

'TECH STOCKS: EBay And Icahn Keep Trading Punches'

"eBay's John Donahoe talks Icahn, conflicts, and $100 stock price (someday)"

'Stock market live blog: S&P 500 retreats from record after downbeat China data'

...

'Miley Cyrus Motorboats Dancer Amazon Ashley in Concert'

'Keep Amazon Prime at $79 for another year'

'Amazon Raises Fee for Prime Shipping'

'Will Amazon Prime Price Hike Drive Mass Exodus?']



Задача 3:

Примените алгоритм K-Means к датасету [Wine Recognition Data Set](https://archive.ics.uci.edu/ml/datasets/wine), чтобы выделить кластеры вин на основе их химического анализа.

def task1\_3():

    data = load\_wine()

    df = pd.DataFrame(data.data, columns=data.feature\_names)

    # Масштабирование данных

    scaler = StandardScaler()

    data\_scaled = scaler.fit\_transform(df)

    # Определение оптимального числа кластеров

    # Метод локтя

    inertia = []

    range\_n\_clusters = range(1, 10)

    for k in range\_n\_clusters:

        kmeans = KMeans(n\_clusters=k, random\_state=42)

        kmeans.fit(data\_scaled)

        inertia.append(kmeans.inertia\_)

    # График метода локтя

    plt.plot(range\_n\_clusters, inertia, marker='o')

    plt.xlabel("Number of Clusters")

    plt.ylabel("Inertia")

    plt.title("Elbow Method for Optimal Clusters")

    plt.show()

    # K-Means

    optimal\_clusters = 3

    kmeans = KMeans(n\_clusters=optimal\_clusters, random\_state=42)

    clusters = kmeans.fit\_predict(data\_scaled)

    # Добавляем кластеры в DataFrame

    df["Cluster"] = clusters

    # Вывод количества элементов в каждом кластере

    print(df["Cluster"].value\_counts())

    # Средние значения признаков по кластерам

    cluster\_centers = pd.DataFrame(kmeans.cluster\_centers\_, columns=data.feature\_names)

    print("Координаты центров кластеров:")

    print(cluster\_centers)

    # Группировка данных по кластерам

    cluster\_analysis = df.groupby("Cluster").mean()

    print("Средние значения признаков по кластерам:")

    print(cluster\_analysis)

    # PCA для визуализации

    pca = PCA(n\_components=2)

    data\_pca = pca.fit\_transform(data\_scaled)

    # Визуализация кластеров

    plt.scatter(data\_pca[:, 0], data\_pca[:, 1], c=clusters, cmap='viridis', s=50)

    plt.title("Кластеры вин (K-Means)")

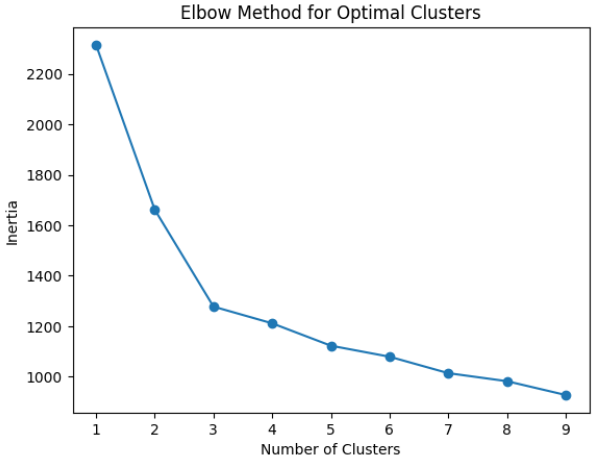
    plt.xlabel("PCA Component 1")

    plt.ylabel("PCA Component 2")

    plt.colorbar(label="Cluster")

    plt.show()

#task1\_3()



Cluster

0 65

2 62

1 51

Name: count, dtype: int64

Координаты центров кластеров:

alcohol malic\_acid ash alcalinity\_of\_ash magnesium \

0 -0.926072 -0.394042 -0.494517 0.170602 -0.491712

1 0.164907 0.871547 0.186898 0.524367 -0.075473

2 0.835232 -0.303810 0.364706 -0.610191 0.577587

total\_phenols flavanoids nonflavanoid\_phenols proanthocyanins \

0 -0.075983 0.020813 -0.033534 0.058266

1 -0.979330 -1.215248 0.726064 -0.779706

2 0.885237 0.977820 -0.562090 0.580287

color\_intensity hue od280/od315\_of\_diluted\_wines proline

0 -0.901914 0.461804 0.270764 -0.753846

1 0.941539 -1.164789 -1.292412 -0.407088

2 0.171063 0.473984 0.779247 1.125185

Средние значения признаков по кластерам:

alcohol malic\_acid ash alcalinity\_of\_ash magnesium \

Cluster

0 12.250923 1.897385 2.231231 20.063077 92.738462

1 13.134118 3.307255 2.417647 21.241176 98.666667

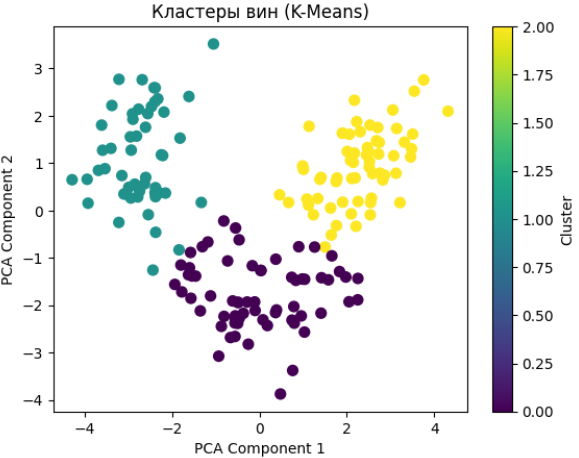
...

Cluster

0 2.973077 1.062708 2.803385 510.169231

1 7.234706 0.691961 1.696667 619.058824

2 5.453548 1.065484 3.163387 1100.225806



Задача 4:

Разделите датасет [Wholesale Customers Data](https://archive.ics.uci.edu/ml/datasets/wholesale+customers) на категории. В этом датасете содержатся данные о расходах клиентов оптового дистрибьютора на разные категории продуктов. Цель - сегментировать клиентов по их годовым расходам на разные категории продуктов.

def task1\_4():

    data = pd.read\_csv("Data/Wholesale customers data.csv")

    scaler = StandardScaler()

    data\_scaled = scaler.fit\_transform(data)

    inertia = []

    range\_n\_clusters = range(1, 10)

    for k in range\_n\_clusters:

        kmeans = KMeans(n\_clusters=k, random\_state=42)

        kmeans.fit(data\_scaled)

        inertia.append(kmeans.inertia\_)

    plt.plot(range\_n\_clusters, inertia, marker='o')

    plt.xlabel("Number of Clusters")

    plt.ylabel("Inertia")

    plt.title("Elbow Method for Optimal Clusters")

    plt.show()

    # K-Means

    optimal\_clusters = 3

    kmeans = KMeans(n\_clusters=optimal\_clusters, random\_state=42)

    clusters = kmeans.fit\_predict(data\_scaled)

    # Добавление кластеров в исходный DataFrame

    data["Cluster"] = clusters

    # Сколько объектов в каждом кластере

    print(data["Cluster"].value\_counts())

    # Средние значения расходов по кластерам

    cluster\_analysis = data.groupby("Cluster").mean()

    print("Средние значения расходов по кластерам:")

    print(cluster\_analysis)

    # PCA для уменьшения размерности

    pca = PCA(n\_components=2)

    data\_pca = pca.fit\_transform(data\_scaled)

    # Визуализация кластеров

    plt.scatter(data\_pca[:, 0], data\_pca[:, 1], c=clusters, cmap='viridis', s=50)

    plt.title("Кластеры клиентов (K-Means)")

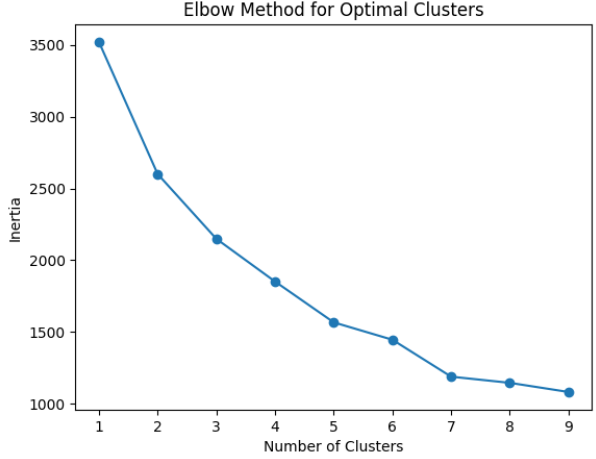
    plt.xlabel("PCA Component 1")

    plt.ylabel("PCA Component 2")

    plt.colorbar(label="Cluster")

    plt.show()

#task1\_4()



Cluster

1 297

0 130

2 13

Name: count, dtype: int64

Средние значения расходов по кластерам:

Channel Region Fresh Milk Grocery \

Cluster

0 2.000000 2.623077 8165.915385 8748.515385 13905.492308

1 1.003367 2.505051 13075.898990 3218.737374 3855.696970

2 1.846154 2.615385 25770.769231 35160.384615 41977.384615

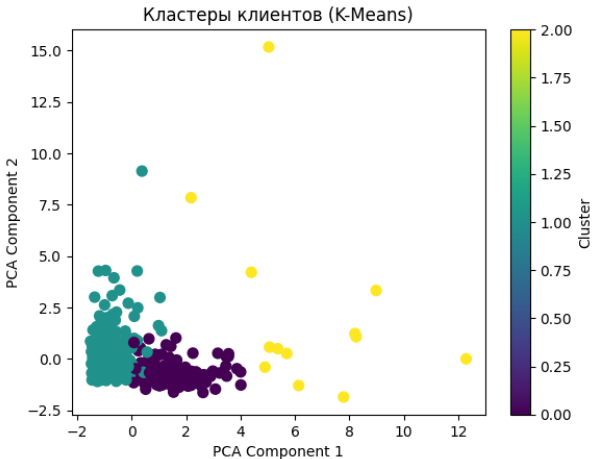
Frozen Detergents\_Paper Delicassen

Cluster

0 1441.630769 5991.146154 1535.007692

1 3620.400673 776.875421 1242.249158

2 6844.538462 19867.384615 7880.307692



Задача 1:

Примените иерархическую кластеризацию к набору данных [Iris](https://archive.ics.uci.edu/ml/datasets/Iris). Используйте дендрограмму для определения оптимального числа кластеров и сравните свои результаты с фактическими метками.

def task2\_1():

    iris = load\_iris()

    X = iris.data

    y = iris.target

    feature\_names = iris.feature\_names

    df = pd.DataFrame(X, columns=feature\_names)

    # Масштабирование данных

    scaler = StandardScaler()

    X\_scaled = scaler.fit\_transform(X)

    # Построение дендрограммы

    linked = linkage(X\_scaled, method='ward')

    plt.figure(figsize=(10, 7))

    dendrogram(linked, truncate\_mode='lastp', p=30)

    plt.title("Дендрограмма")

    plt.xlabel("Samples")

    plt.ylabel("Euclidean Distance")

    plt.show()

    # Применение агломеративной кластеризации

    optimal\_clusters = 3

    agg\_clustering = AgglomerativeClustering(n\_clusters=optimal\_clusters, metric='euclidean', linkage='ward')

    clusters = agg\_clustering.fit\_predict(X\_scaled)

    # Добавление кластеров в DataFrame

    df['Cluster'] = clusters

    print(df['Cluster'].value\_counts())

    # Оценка качества кластеризации

    ari = adjusted\_rand\_score(y, clusters)  # ARI (Adjusted Rand Index)

    ami = adjusted\_mutual\_info\_score(y, clusters)  # AMI (Adjusted Mutual Information)

    print(f"Adjusted Rand Index (ARI): {ari}")

    print(f"Adjusted Mutual Information (AMI): {ami}")

    # PCA для снижения размерности

    pca = PCA(n\_components=2)

    X\_pca = pca.fit\_transform(X\_scaled)

    # Визуализация кластеров

    plt.scatter(X\_pca[:, 0], X\_pca[:, 1], c=clusters, cmap='viridis', s=50)

    plt.title("Кластеры (Иерархическая кластеризация)")

    plt.xlabel("PCA Component 1")

    plt.ylabel("PCA Component 2")

    plt.colorbar(label="Cluster")

    plt.show()

    # Визуализация фактических меток

    plt.scatter(X\_pca[:, 0], X\_pca[:, 1], c=y, cmap='viridis', s=50)

    plt.title("Фактические метки")

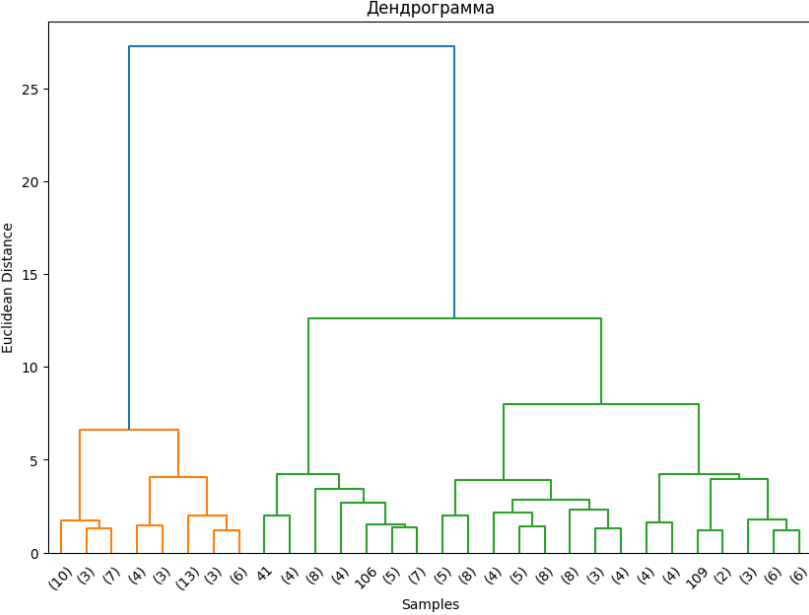
    plt.xlabel("PCA Component 1")

    plt.ylabel("PCA Component 2")

    plt.colorbar(label="Class")

    plt.show()

#task2\_1()



Cluster

0 71

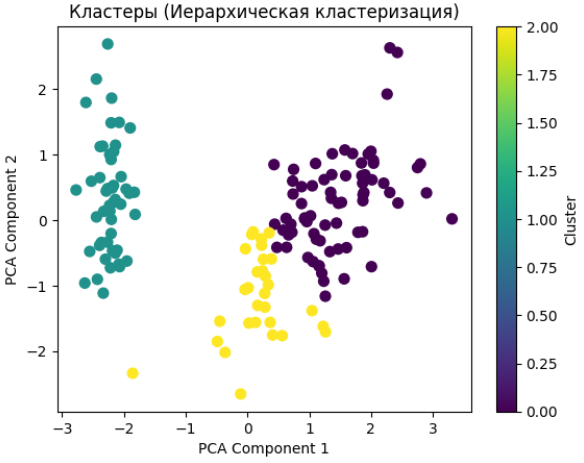
1 49

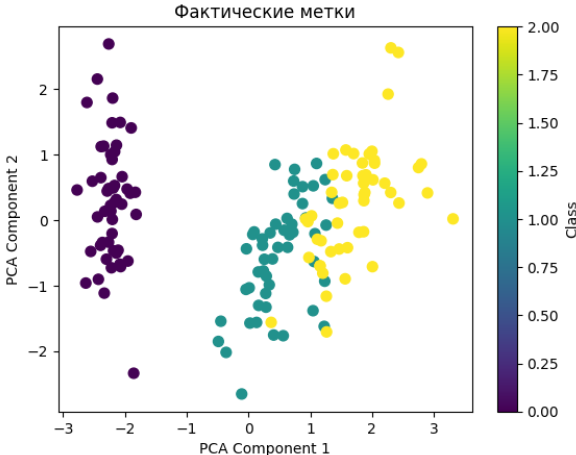
2 30

Name: count, dtype: int64

Adjusted Rand Index (ARI): 0.6153229932145449

Adjusted Mutual Information (AMI): 0.671286134807129





Задача 2:

Текстовый анализ. Используйте датасет [20 News Groups](http://qwone.com/~jason/20Newsgroups/), который включает в себя около 20 000 сообщений, сгруппированных примерно на 20 новостных групп. Ваша задача заключается в иерархической кластеризации данных с использованием любой техники векторизации текста.

def task2\_2():

    newsgroups = fetch\_20newsgroups(subset='all', remove=('headers', 'footers', 'quotes'))

    data = newsgroups.data

    target = newsgroups.target

    target\_names = newsgroups.target\_names

    print(f"Количество документов: {len(data)}")

    print(f"Количество категорий: {len(target\_names)}")

    print(f"Пример текста:\n{data[0]}")

    # Настройка векторизатора TF-IDF

    vectorizer = TfidfVectorizer(max\_features=2000, stop\_words='english')  # Ограничиваем число признаков для эффективности

    X\_tfidf = vectorizer.fit\_transform(data)

    print(f"Размерность векторизованных данных: {X\_tfidf.shape}")

    # Используем подвыборку для дендрограммы (например, 200 документов)

    sample\_size = 200

    X\_sample = X\_tfidf[:sample\_size].toarray()

    # Построение дендрограммы

    linked = linkage(X\_sample, method='ward')

    plt.figure(figsize=(10, 7))

    dendrogram(linked, truncate\_mode='lastp', p=20)

    plt.title("Дендрограмма (подвыборка)")

    plt.xlabel("Documents")

    plt.ylabel("Euclidean Distance")

    plt.show()

    # Применение агломеративной кластеризации

    optimal\_clusters = 20

    agg\_clustering = AgglomerativeClustering(

        n\_clusters=optimal\_clusters, metric='euclidean', linkage='ward'

    )

    clusters = agg\_clustering.fit\_predict(X\_tfidf.toarray())

    # Сравнение с исходными метками

    df = pd.DataFrame({"Cluster": clusters, "Category": target})

    print(df.groupby("Cluster")["Category"].value\_counts())

    # PCA для уменьшения размерности

    pca = PCA(n\_components=2)

    X\_pca = pca.fit\_transform(X\_tfidf.toarray())

    # Визуализация кластеров

    plt.scatter(X\_pca[:, 0], X\_pca[:, 1], c=clusters, cmap='viridis', s=10)

    plt.title("Кластеры (Иерархическая кластеризация)")

    plt.xlabel("PCA Component 1")

    plt.ylabel("PCA Component 2")

    plt.colorbar(label="Cluster")

    plt.show()

    # Оценка качества кластеризации

    ari = adjusted\_rand\_score(target, clusters)

    ami = adjusted\_mutual\_info\_score(target, clusters)

    print(f"Adjusted Rand Index (ARI): {ari}")

    print(f"Adjusted Mutual Information (AMI): {ami}")

#task2\_2()

Количество документов: 18846

Количество категорий: 20

Пример текста:

I am sure some bashers of Pens fans are pretty confused about the lack

of any kind of posts about the recent Pens massacre of the Devils. Actually,

I am bit puzzled too and a bit relieved. However, I am going to put an end

to non-PIttsburghers' relief with a bit of praise for the Pens. Man, they

are killing those Devils worse than I thought. Jagr just showed you why

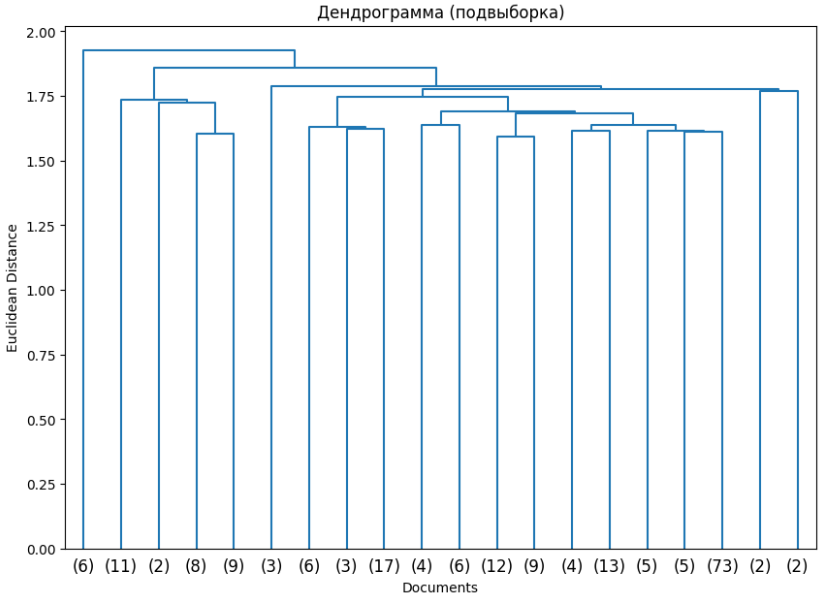
he is much better than his regular season stats. He is also a lot

fo fun to watch in the playoffs. Bowman should let JAgr have a lot of

fun in the next couple of games since the Pens are going to beat the pulp out of Jersey anyway. I was very disappointed not to see the Islanders lose the final

regular season game. PENS RULE!!!

Размерность векторизованных данных: (18846, 2000)



Cluster Category

0 5 606

1 324

2 260

3 159

6 145

...

19 13 38

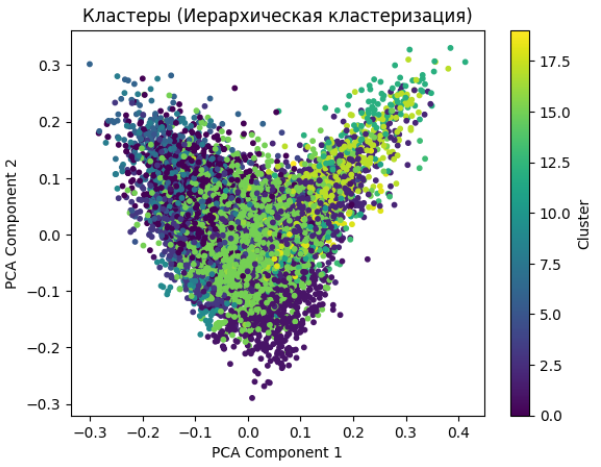
10 3

1 1

2 1

11 1

Name: count, Length: 255, dtype: int64



Adjusted Rand Index (ARI): 0.042146379533565134

Adjusted Mutual Information (AMI): 0.24339907108535352

Задача 3:

Получите данные о фильмах и их рейтинге (ссылка на данные: [Movies Dataset](https://www.kaggle.com/rounakbanik/movie-recommender-systems/data)). Ваша задача - с помощью иерархической кластеризации группировать фильмы по жанрам и средним рейтингам.

def task2\_3():

    movies\_path = "Data/movies\_metadata.csv"

    ratings\_path = "Data/ratings\_small.csv"

    movies = pd.read\_csv(movies\_path, low\_memory=False)

    ratings = pd.read\_csv(ratings\_path)

    # Извлечение жанров из столбца 'genres'

    def extract\_genres(genres\_str):

        try:

            genres = ast.literal\_eval(genres\_str)

            return [genre['name'] for genre in genres]

        except:

            return []

    movies['genres\_list'] = movies['genres'].apply(extract\_genres)

    # Приведение столбца 'id' в movies к числовому типу

    movies['id'] = pd.to\_numeric(movies['id'], errors='coerce')

    movies.dropna(subset=['id'], inplace=True)

    # Приведение типов для объединения

    movies['id'] = movies['id'].astype(int)

    ratings['movieId'] = ratings['movieId'].astype(int)

    # Расчет среднего рейтинга для каждого фильма

    average\_ratings = ratings.groupby('movieId')['rating'].mean().reset\_index()

    average\_ratings.rename(columns={'rating': 'average\_rating'}, inplace=True)

    # Объединение данных о фильмах с данными о рейтингах

    movies = movies.merge(average\_ratings, left\_on='id', right\_on='movieId', how='inner')

    # Оставляем только необходимые столбцы

    movies = movies[['title', 'genres\_list', 'average\_rating']]

    movies = movies[movies['genres\_list'].map(len) > 0]

    # Преобразование жанров в бинарный вид

    mlb = MultiLabelBinarizer()

    genres\_encoded = pd.DataFrame(mlb.fit\_transform(movies['genres\_list']), columns=mlb.classes\_)

    # Добавляем бинарные жанры к данным

    movies = pd.concat([movies, genres\_encoded], axis=1)

    # Оставляем только числовые столбцы для кластеризации

    features = ['average\_rating'] + list(mlb.classes\_)

    data = movies[features]

    scaler = StandardScaler()

    data\_scaled = scaler.fit\_transform(data)

    # Удаление строк с некорректными значениями

    valid\_indices = ~np.isnan(data\_scaled).any(axis=1) & ~np.isinf(data\_scaled).any(axis=1)

    movies = movies[valid\_indices].reset\_index(drop=True)

    data\_scaled = data\_scaled[valid\_indices]

    # Построение дендрограммы

    linked = linkage(data\_scaled, method='ward')

    plt.figure(figsize=(15, 7))

    dendrogram(linked, truncate\_mode='lastp', p=30)

    plt.title("Дендрограмма фильмов")

    plt.xlabel("Samples")

    plt.ylabel("Euclidean Distance")

    plt.show()

    # Применение агломеративной кластеризации

    optimal\_clusters = 5  # Число кластеров из дендрограммы

    agg\_clustering = AgglomerativeClustering(n\_clusters=optimal\_clusters, linkage='ward')

    clusters = agg\_clustering.fit\_predict(data\_scaled)

    # Добавляем кластеры в данные

    movies['Cluster'] = clusters

    # Смотрим количество фильмов в каждом кластере

    print("\nКоличество фильмов в каждом кластере:")

    print(movies['Cluster'].value\_counts())

    # Выбираем только числовые столбцы для расчёта среднего значения

    numerical\_columns = movies.select\_dtypes(include=['number']).columns

    cluster\_analysis = movies.groupby('Cluster')[numerical\_columns].mean()

    print("\nСредние значения характеристик для каждого кластера:")

    print(cluster\_analysis)

    # Частота жанров в каждом кластере

    genre\_cluster\_distribution = movies.groupby('Cluster')[mlb.classes\_].mean()

    print("\nЧастота жанров по кластерам:")

    print(genre\_cluster\_distribution)

    # PCA для уменьшения размерности до 2D

    pca = PCA(n\_components=2)

    data\_pca = pca.fit\_transform(data\_scaled)

    # Визуализация кластеров

    plt.figure(figsize=(10, 7))

    plt.scatter(data\_pca[:, 0], data\_pca[:, 1], c=clusters, cmap='viridis', s=10)

    plt.title("Кластеры фильмов (Иерархическая кластеризация)")

    plt.xlabel("PCA Component 1")

    plt.ylabel("PCA Component 2")

    plt.colorbar(label="Cluster")

    plt.show()

#task2\_3()

Пример данных о фильмах:

adult belongs\_to\_collection budget \

0 False {'id': 10194, 'name': 'Toy Story Collection', ... 30000000

1 False NaN 65000000

2 False {'id': 119050, 'name': 'Grumpy Old Men Collect... 0

3 False NaN 16000000

4 False {'id': 96871, 'name': 'Father of the Bride Col... 0

genres \

0 [{'id': 16, 'name': 'Animation'}, {'id': 35, '...

1 [{'id': 12, 'name': 'Adventure'}, {'id': 14, '...

2 [{'id': 10749, 'name': 'Romance'}, {'id': 35, ...

3 [{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...

4 [{'id': 35, 'name': 'Comedy'}]

homepage id imdb\_id original\_language \

0 <http://toystory.disney.com/toy-story> 862 tt0114709 en

1 NaN 8844 tt0113497 en

2 NaN 15602 tt0113228 en

3 NaN 31357 tt0114885 en

4 NaN 11862 tt0113041 en

original\_title \

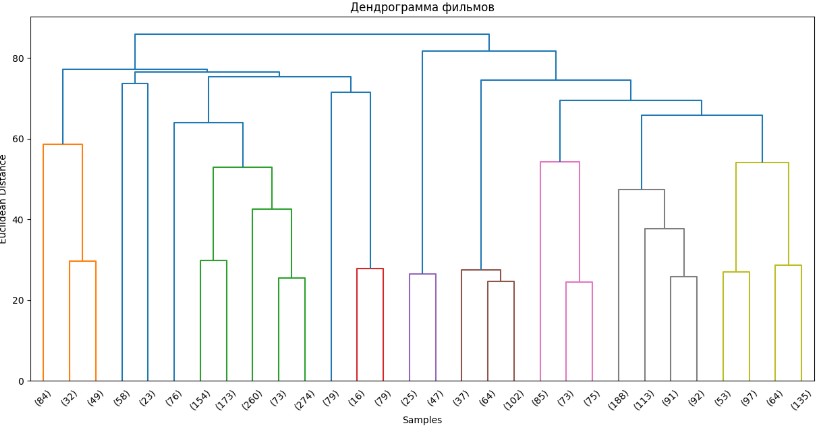
0 Toy Story

1 Jumanji

...

3 Casino [Drama, Crime] 3.555556

4 Sense and Sensibility [Drama, Romance] 5.000000



Количество фильмов в каждом кластере:

Cluster

1 1269

0 1184

3 165

4 81

2 72

Name: count, dtype: int64

Средние значения характеристик для каждого кластера:

average\_rating Action Adventure Animation Comedy Crime \

Cluster

0 3.302815 0.015203 0.008446 0.000000 0.430743 0.011824

1 3.362639 0.357762 0.223010 0.000000 0.208038 0.321513

2 3.217207 0.138889 0.333333 1.000000 0.361111 0.000000

3 3.312579 0.187879 0.133333 0.000000 0.109091 0.036364

4 3.426514 0.283951 0.197531 0.012346 0.185185 0.024691

Documentary Drama Family Fantasy ... Horror Music \

Cluster ...

0 0.088682 0.671453 0.005912 0.005068 ... 0.005068 0.081081

1 0.000000 0.425532 0.066982 0.148936 ... 0.206462 0.000000

2 0.000000 0.111111 0.625000 0.291667 ... 0.041667 0.055556

3 0.000000 0.896970 0.006061 0.006061 ... 0.006061 0.000000

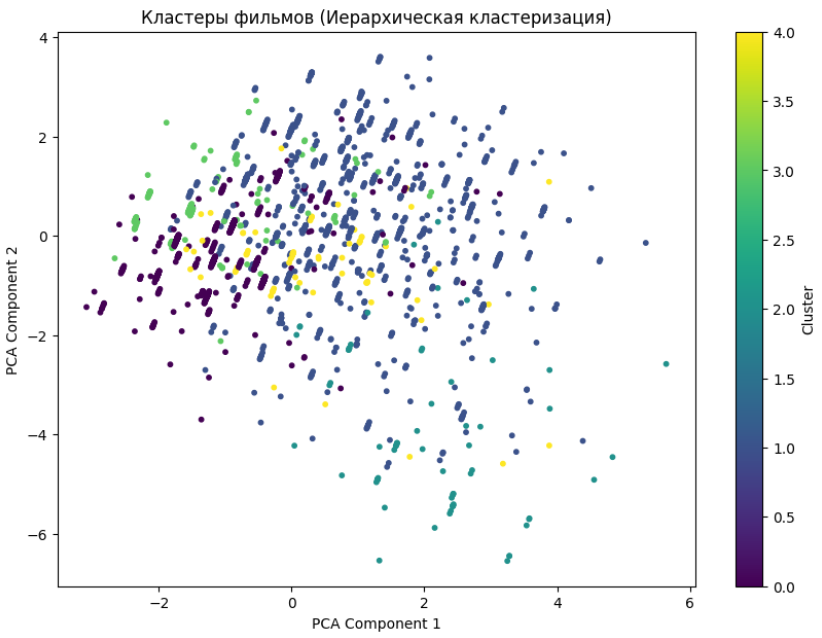
...

1 0.000000

2 0.000000

3 0.000000

4 0.740741



Задача 4:

Используйте набор данных [Seeds](https://archive.ics.uci.edu/ml/datasets/seeds) для кластеризации различных сортов пшеницы. Данный набор данных содержит измерения геометрических свойств семян семи различных видов пшеницы. Ваша задача провести иерархическую кластеризацию и визуализировать результаты с помощью дендрограммы.

def task2\_4():

    columns = [

        "Area", "Perimeter", "Compactness", "KernelLength", "KernelWidth",

        "AsymmetryCoefficient", "KernelGrooveLength", "Type"

    ]

    data = pd.read\_csv("Data/seeds\_dataset.txt", delim\_whitespace=True, names=columns)

    # Удаляем столбец "Type" (содержит метки классов)

    features = data.drop(columns=["Type"])

    scaler = StandardScaler()

    data\_scaled = scaler.fit\_transform(features)

    # Построение дендрограммы

    # Построение иерархической кластеризации

    linked = linkage(data\_scaled, method="ward")

    # Визуализация дендрограммы

    plt.figure(figsize=(15, 7))

    dendrogram(linked, truncate\_mode="lastp", p=30)

    plt.title("Дендрограмма (Seeds Dataset)")

    plt.xlabel("Samples")

    plt.ylabel("Euclidean Distance")

    plt.show()

    # Применение агломеративной кластеризации

    optimal\_clusters = 3  # Предположим, что оптимальное число кластеров равно 3

    agg\_clustering = AgglomerativeClustering(n\_clusters=optimal\_clusters, linkage="ward")

    clusters = agg\_clustering.fit\_predict(data\_scaled)

    # Добавляем кластеры в исходный DataFrame

    data["Cluster"] = clusters

    # Анализ кластеров

    print("\nКоличество объектов в каждом кластере:")

    print(data["Cluster"].value\_counts())

    # Средние значения признаков для каждого кластера

    cluster\_analysis = data.groupby("Cluster").mean()

    print("\nСредние значения признаков по кластерам:")

    print(cluster\_analysis)

    # Визуализация кластеров (с помощью PCA)

    # Уменьшение размерности до 2D для визуализации

    pca = PCA(n\_components=2)

    data\_pca = pca.fit\_transform(data\_scaled)

    # Визуализация кластеров

    plt.figure(figsize=(10, 7))

    plt.scatter(data\_pca[:, 0], data\_pca[:, 1], c=clusters, cmap="viridis", s=50)

    plt.title("Кластеры семян (Иерархическая кластеризация)")

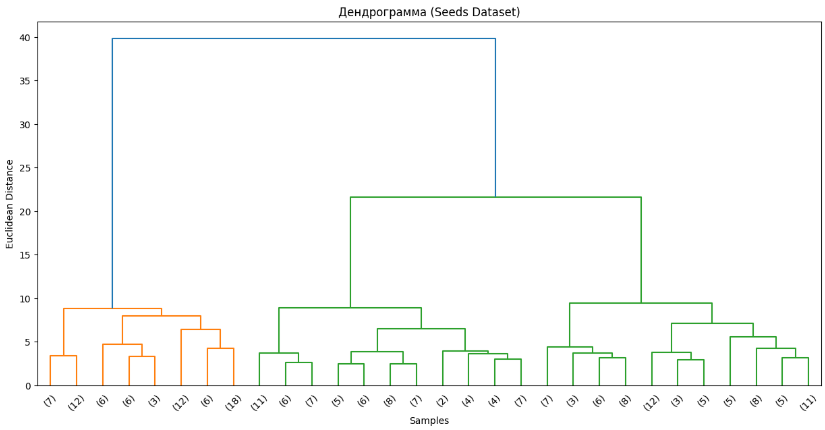
    plt.xlabel("PCA Component 1")

    plt.ylabel("PCA Component 2")

    plt.colorbar(label="Cluster")

    plt.show()

#task2\_4()



Количество объектов в каждом кластере:

Cluster

0 73

1 70

2 67

Name: count, dtype: int64

Средние значения признаков по кластерам:

Area Perimeter Compactness KernelLength KernelWidth \

Cluster

0 14.199041 14.233562 0.879190 5.478233 3.226452

1 18.371429 16.145429 0.884400 6.158171 3.684629

2 11.872388 13.257015 0.848072 5.238940 2.848537

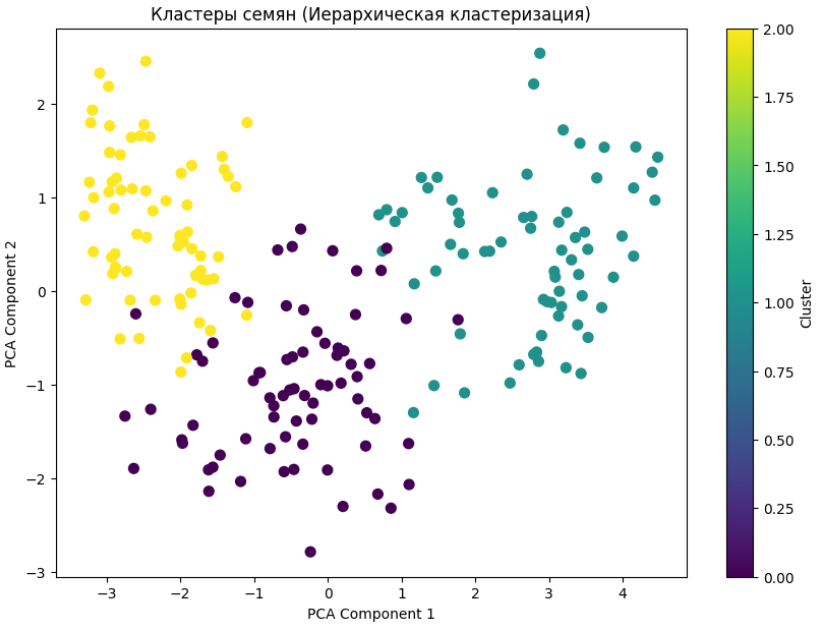
AsymmetryCoefficient KernelGrooveLength Type

Cluster

0 2.612181 5.086178 1.191781

1 3.639157 1 6.017371 1.942857

2 4.949433 5.122209 2.940299



Задача 1:

Используйте DBSCAN для кластеризации набора данных [Iris](https://archive.ics.uci.edu/ml/datasets/Iris). Сравните полученные результаты с изначальными метками. DBSCAN хорош для этой задачи, поскольку он может находить кластеры произвольной формы.

def task3\_1():

    iris = load\_iris()

    X = iris.data  # Признаки

    y = iris.target  # Метки классов

    scaler = StandardScaler()

    X\_scaled = scaler.fit\_transform(X)

    # Применение DBSCAN

    # Настройка гиперпараметров DBSCAN

    dbscan = DBSCAN(eps=0.5, min\_samples=5)

    clusters = dbscan.fit\_predict(X\_scaled)

    # Добавление кластеров в DataFrame

    data = pd.DataFrame(X, columns=iris.feature\_names)

    data['Cluster'] = clusters

    data['True\_Label'] = y

    # Сравнение результатов

    ari = adjusted\_rand\_score(y, clusters)

    ami = adjusted\_mutual\_info\_score(y, clusters)

    print(f"Adjusted Rand Index (ARI): {ari}")

    print(f"Adjusted Mutual Information (AMI): {ami}")

    # Визуализация результатов

    # Преобразование данных в двумерное пространство с помощью PCA

    pca = PCA(n\_components=2)

    X\_pca = pca.fit\_transform(X\_scaled)

    # Визуализация кластеров

    plt.figure(figsize=(10, 7))

    plt.scatter(X\_pca[:, 0], X\_pca[:, 1], c=clusters, cmap="viridis", s=50)

    plt.title("Кластеры (DBSCAN)")

    plt.xlabel("PCA Component 1")

    plt.ylabel("PCA Component 2")

    plt.colorbar(label="Cluster")

    plt.show()

    # Визуализация истинных меток классов

    plt.figure(figsize=(10, 7))

    plt.scatter(X\_pca[:, 0], X\_pca[:, 1], c=y, cmap="viridis", s=50)

    plt.title("Истинные метки классов")

    plt.xlabel("PCA Component 1")

    plt.ylabel("PCA Component 2")

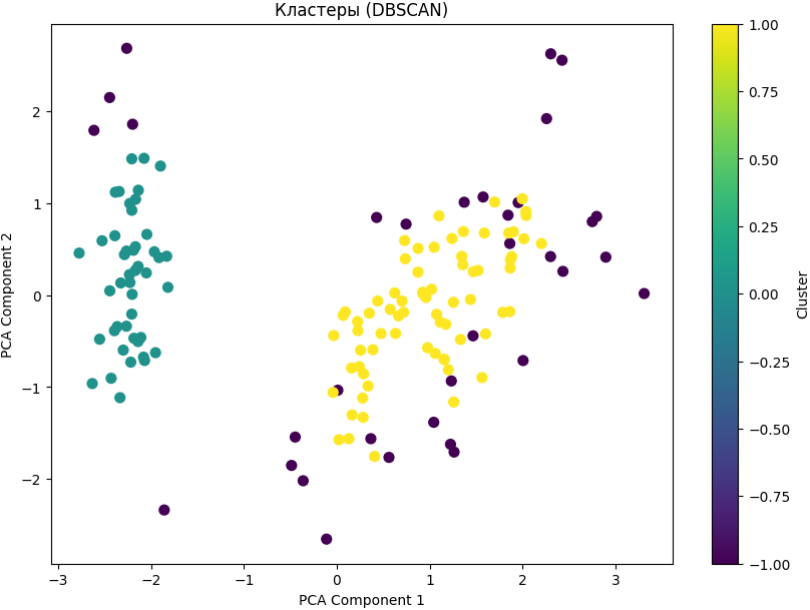
    plt.colorbar(label="True Label")

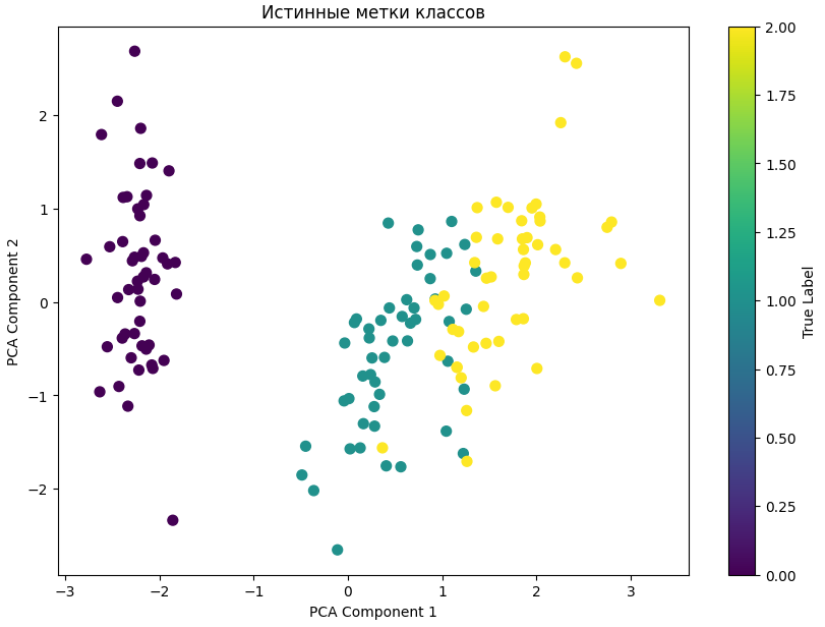
    plt.show()

#task3\_1()

Adjusted Rand Index (ARI): 0.4420986685885924

Adjusted Mutual Information (AMI): 0.5051666404374139





Задача 2:

Примените DBSCAN для кластеризации датасета [Moons](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make\_moons.html) из библиотеки sklearn. Этот набор данных идеально подходит для демонстрации преимуществ DBSCAN, потому что он содержит два визуально различимых "месяца", и каждый из них должен стать отдельным кластером.

def task3\_2():

    X, y = make\_moons(n\_samples=300, noise=0.05, random\_state=42)

    data = pd.DataFrame(X, columns=["Feature 1", "Feature 2"])

    data["True Label"] = y

    plt.figure(figsize=(10, 7))

    plt.scatter(X[:, 0], X[:, 1], c=y, cmap="viridis", s=50)

    plt.title("Исходные данные (Moons)")

    plt.xlabel("Feature 1")

    plt.ylabel("Feature 2")

    plt.show()

    # Масштабирование данных

    scaler = StandardScaler()

    X\_scaled = scaler.fit\_transform(X)

    # Применение DBSCAN

    dbscan = DBSCAN(eps=0.2, min\_samples=5)

    clusters = dbscan.fit\_predict(X\_scaled)

    # Добавление кластеров в DataFrame

    data["Cluster"] = clusters

    # Проверка распределения точек по кластерам

    print("\nРаспределение точек по кластерам:")

    print(data["Cluster"].value\_counts())

    # Визуализация кластеров, найденных DBSCAN

    plt.figure(figsize=(10, 7))

    plt.scatter(X[:, 0], X[:, 1], c=clusters, cmap="viridis", s=50)

    plt.title("Кластеры, выделенные DBSCAN")

    plt.xlabel("Feature 1")

    plt.ylabel("Feature 2")

    plt.colorbar(label="Cluster")

    plt.show()

    # Анализ шумовых точек

    noise\_points = data[data["Cluster"] == -1]

    print("\nКоличество шумовых точек:")

    print(len(noise\_points))

    # Визуализация шумовых точек отдельно

    plt.figure(figsize=(10, 7))

    plt.scatter(X[:, 0], X[:, 1], c=clusters, cmap="viridis", s=50, alpha=0.5)

    plt.scatter(

        noise\_points["Feature 1"], noise\_points["Feature 2"],

        c="red", label="Шумовые точки", edgecolors="black"

    )

    plt.title("Шумовые точки (DBSCAN)")

    plt.xlabel("Feature 1")

    plt.ylabel("Feature 2")

    plt.legend()

    plt.show()

Распределение точек по кластерам:

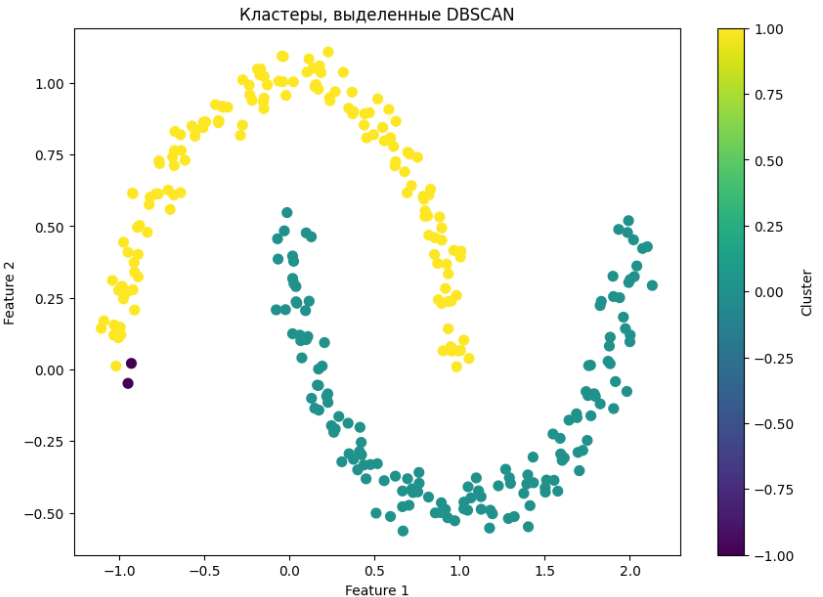
Cluster

0 150

1 148

-1 2

Name: count, dtype: int64



Количество шумовых точек:

2

