Практическая работа 6

Тема: Кластеризация данных

Цель: Овладеть методами кластерного анализа с использованием различных алгоритмов кластеризации, таких как k-means, иерархическая кластеризация и DBSCAN. Научиться визуализировать результаты и оценивать качество кластеризации с помощью коэффициента силуэта и метода локтя.

ОГЛАВЛЕНИЕ

ЗАДАНИЕ 5

ШАГ 1: ПОДБОР ДАННЫХ 6

1.1. ПОИСК И ВЫБОР НАБОРА ДАННЫХ 6

1.2. ПЕРВИЧНАЯ ОБРАБОТКА ДАННЫХ 7

1.2.1. ОЧИСТКА ДАННЫХ..... 7

ПРИМЕР ОБРАБОТКИ ДАННЫХ В PYTHON 7

1.3. НОРМАЛИЗАЦИЯ ДАННЫХ 8

СПОСОБЫ НОРМАЛИЗАЦИИ..... 8

ПРИМЕР НОРМАЛИЗАЦИИ ДАННЫХ В PYTHON..... 8

1.4. ВЫБОР ПРИЗНАКОВ ДЛЯ КЛАСТЕРИЗАЦИИ 9

ПРИМЕР ВЫБОРА ПРИЗНАКОВ И ОБРАБОТКИ ВЫБРОСОВ 9

ШАГ 2: РЕАЛИЗАЦИЯ АЛГОРИТМОВ КЛАСТЕРИЗАЦИИ..... 10

2.1. АЛГОРИТМ K-MEANS 10

ШАГИ РАБОТЫ АЛГОРИТМА K-MEANS..... 10

КОД РЕАЛИЗАЦИИ K-MEANS В PYTHON 10

2.2. ИЕРАРХИЧЕСКАЯ КЛАСТЕРИЗАЦИЯ..... 11

ОПИСАНИЕ АЛГОРИТМА 11

ДВА ОСНОВНЫХ ПОДХОДА..... 11

ШАГИ АГЛОМЕРАТИВНОЙ КЛАСТЕРИЗАЦИИ 11

КОД РЕАЛИЗАЦИИ ИЕРАРХИЧЕСКОЙ КЛАСТЕРИЗАЦИИ В PYTHON.. 12

2.3. АЛГОРИТМ DBSCAN 12

ОПИСАНИЕ АЛГОРИТМА 12

ОСНОВНЫЕ ПАРАМЕТРЫ DBSCAN 12
ШАГИ РАБОТЫ DBSCAN 13
КОД РЕАЛИЗАЦИИ DBSCAN В PYTHON 13
ШАГ 3: ОПРЕДЕЛЕНИЕ КОЛИЧЕСТВА КЛАСТЕРОВ 14
3.1. МЕТОД ЛОКТЯ 14
ОПИСАНИЕ МЕТОДА 14
ШАГИ ДЛЯ МЕТОДА ЛОКТЯ: 14
КОД ДЛЯ МЕТОДА ЛОКТЯ В PYTHON 15
3.2. КОЭФФИЦИЕНТ СИЛУЭТА 15
ОПИСАНИЕ МЕТОДА 15
ШАГИ ДЛЯ ИСПОЛЬЗОВАНИЯ КОЭФФИЦИЕНТА СИЛУЭТА 16
КОД ДЛЯ РАСЧЁТА КОЭФФИЦИЕНТА СИЛУЭТА В PYTHON 16
3.3. СОВМЕЩЕНИЕ МЕТОДОВ ДЛЯ ОПТИМАЛЬНОГО ВЫБОРА КОЛИЧЕСТВА КЛАСТЕРОВ 16
ШАГ 4: ВИЗУАЛИЗАЦИЯ ИЕРАРХИЧЕСКОЙ КЛАСТЕРИЗАЦИИ 18
4.1. ЧТО ТАКОЕ ДЕНДРОГРАММА? 18
ЭЛЕМЕНТЫ ДЕНДРОГРАММЫ: 18
4.2. ПОСТРОЕНИЕ ДЕНДРОГРАММЫ 18
КОД ДЛЯ ПОСТРОЕНИЯ ДЕНДРОГРАММЫ В PYTHON 19
4.3. ВАРИАНТЫ МЕТОДОВ ОБЪЕДИНЕНИЯ КЛАСТЕРОВ (LINKAGE) 19
ПРИМЕР ИСПОЛЬЗОВАНИЯ ДРУГОГО МЕТОДА (COMPLETE LINKAGE) 20
4.4. АНАЛИЗ ДЕНДРОГРАММЫ 20
ПРИМЕР АНАЛИЗА 21
ШАГ 5: ПОСТРОЕНИЕ ГРАФИКОВ 22

5.1. ПОСТРОЕНИЕ ГРАФИКОВ ДЛЯ K-MEANS 22
2D ВИЗУАЛИЗАЦИЯ КЛАСТЕРОВ..... 22
КОД ДЛЯ ВИЗУАЛИЗАЦИИ КЛАСТЕРОВ K-MEANS В PYTHON 22
5.2. ПОСТРОЕНИЕ ГРАФИКОВ ДЛЯ ИЕРАРХИЧЕСКОЙ КЛАСТЕРИЗАЦИИ 23
ПРИМЕР ВИЗУАЛИЗАЦИИ РЕЗУЛЬТАТОВ ИЕРАРХИЧЕСКОЙ КЛАСТЕРИЗАЦИИ 23
5.3. ПОСТРОЕНИЕ ГРАФИКОВ ДЛЯ DBSCAN 24
КОД ДЛЯ ВИЗУАЛИЗАЦИИ РЕЗУЛЬТАТОВ DBSCAN В PYTHON 24
5.4. ДОПОЛНИТЕЛЬНЫЕ ГРАФИКИ ДЛЯ ОЦЕНКИ КЛАСТЕРОВ 25
ПРИМЕР ГРАФИКА СИЛУЭТА ДЛЯ K-MEANS..... 25
ШАГ 6: СРАВНЕНИЕ РЕЗУЛЬТАТОВ 26
6.1. СРАВНЕНИЕ ПО КОЭФФИЦИЕНТУ СИЛУЭТА 26
КОД ДЛЯ РАСЧЁТА СРЕДНЕГО КОЭФФИЦИЕНТА СИЛУЭТА ДЛЯ РАЗНЫХ АЛГОРИТМОВ 26
6.2. СРАВНЕНИЕ ПО ВИЗУАЛИЗАЦИИ КЛАСТЕРОВ 27
ПРИМЕР ВОПРОСОВ ДЛЯ АНАЛИЗА 27
6.3. СРАВНЕНИЕ НА ОСНОВЕ УСТОЙЧИВОСТИ К ПАРАМЕТРАМ 28
ПРИМЕР ТЕСТИРОВАНИЯ УСТОЙЧИВОСТИ 28
6.4. ВЫВОДЫ НА ОСНОВЕ СРАВНЕНИЯ..... 29
ПРИМЕР ВЫВОДОВ 29

ЗАДАНИЕ

1. Подбор данных.

Найти и подготовить набор данных для решения задачи кластеризации. Данные не должны повторяться в группе. Если необходимо, выполнить предобработку данных: очистку, нормализацию признаков и устранение пропущенных значений.

2. Реализация алгоритмов кластеризации.
    a. k-means;
    b. Иерархическая кластеризация;
    c. DBSCAN.
3. Определение количества кластеров.
    a. Метода локтя;
    b. Коэффициента силуэта.
4. Визуализация иерархической кластеризации.

Построить дендрограмму для иерархической кластеризации, чтобы проанализировать связи между объектами.

5. Построение графиков.

Построить графики результатов для всех применённых алгоритмов кластеризации. Визуально оценить качество кластеризации.

6. Сравнение результатов.

Провести сравнение результатов кластеризации между методами. Сравните силуэтные коэффициенты и другие метрики для оценки качества кластеров.

ШАГ 1: ПОДБОР ДАННЫХ

1.1. Поиск и выбор набора данных

Для успешного выполнения задачи кластеризации необходимо подобрать качественный набор данных. Важно учитывать следующие критерии при выборе данных:

- *Соответствие цели кластеризации:* Данные должны быть релевантны задаче и содержать признаки, по которым можно выделить группы (кластеры);
- *Объем данных:* Достаточный объем данных для того, чтобы кластеризация имела смысл. Если данных слишком мало, результат кластеризации может быть неинформативным;
- *Множество признаков:* Наличие нескольких переменных (признаков) для того, чтобы кластеризация могла выявить разнообразные закономерности и сегменты.

Примеры популярных наборов данных:

- *Iris dataset:* Один из самых известных наборов данных для кластеризации. Содержит информацию о длине и ширине лепестков и чашелистиков 150 образцов ириса;
- *Wine dataset:* Содержит химические показатели различных видов вина и может быть использован для кластеризации по признакам качества.

Источники для поиска данных:

- Kaggle (kaggle.com);
- UCI Machine Learning Repository;
- Открытые данные государственных порталов.

1.2. Первичная обработка данных

После того как данные выбраны, необходимо провести их предварительную обработку, чтобы они были готовы для кластеризации.

1.2.1. Очистка данных

В процессе очистки важно обратить внимание на:

- Повторяющиеся данные: Проверить, есть ли дубликаты строк в наборе данных. Например, с помощью метода drop_duplicates() в pandas можно легко удалить дубликаты;
- Пропущенные значения: Если в данных есть пропуски (например, отсутствующие значения признаков), нужно решить, как их обработать. Возможные варианты:
    - Удалить строки с пропущенными значениями (если таких строк немного);
    - Заполнить пропуски средним или медианным значением признака для числовых данных, или самым частым значением для категориальных признаков;
    - Использовать методы предсказания недостающих значений (например, с помощью модели регрессии или KNN).

Пример обработки данных в Python

Листинг 1 — Пример обработки данных

```
import pandas as pd
data = pd.read_csv('dataset.csv')
data = data.drop_duplicates()
data.fillna(data.mean(), inplace=True)
```

1.3. Нормализация данных

Большинство алгоритмов кластеризации чувствительны к масштабу признаков, особенно такие, как K-means. Признаки, измеренные в различных единицах, могут оказывать разное влияние на результаты кластеризации. Например, если один признак измеряется в километрах, а другой — в метрах, то более крупные по значению признаки могут доминировать в процессе кластеризации.

Способы нормализации

1. Минимум-Максимум (Min-Max): Преобразование значений в интервал от 0 до 1. Используется для того, чтобы значения признаков не выходили за определённый диапазон.
2. Z-нормализация (StandardScaler): Приведение значений к среднему значению 0 и стандартному отклонению 1. Подходит для данных с нормальным распределением, где важна относительная позиция значений.

Формула:

$$X_{\text{норм}} = \frac{X - \mu}{\sigma}$$

где μ — среднее значение, σ — стандартное отклонение.

Пример нормализации данных в Python

Листинг 2 — Пример нормализации данных

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
normalized_data = scaler.fit_transform(data)
```

1.4. Выбор признаков для кластеризации

Если в данных слишком много признаков, это может усложнить процесс кластеризации и визуализации результатов. Чтобы избежать “проклятия размерности”, можно выполнить:

- Выбор признаков: Оставить только те признаки, которые наиболее важны для решения задачи. Например, это может быть сделано на основе анализа корреляции или с помощью методов уменьшения размерности, таких как PCA.
- Анализ выбросов: Некоторые наблюдения могут быть выбросами, которые могут повлиять на результат. Для таких данных полезно применять методы обнаружения выбросов (например, на основе межквартильного размаха).

Пример выбора признаков и обработки выбросов

Листинг 3 — Пример выбора признаков и обработки выбросов

```
correlation_matrix = data.corr()
print(correlation_matrix)
Q1 = data.quantile(0.25)
Q3 = data.quantile(0.75)
IQR = Q3 - Q1
filtered_data = data[~((data < (Q1 - 1.5 * IQR)) | (data > (Q3 +
1.5 * IQR))).any(axis=1)]
```

ШАГ 2: РЕАЛИЗАЦИЯ АЛГОРИТМОВ КЛАСТЕРИЗАЦИИ

2.1. Алгоритм K-means

K-means — это один из самых популярных и простых в реализации алгоритмов кластеризации. Его суть заключается в том, что он делит данные на K групп (кластеров), минимизируя внутрикластерные различия (сумму квадратов расстояний от каждого объекта до ближайшего центроида).

Шаги работы алгоритма K-means

1. Задать количество кластеров K.
2. Инициализировать K случайных центроидов.
3. Назначить каждый объект данных к ближайшему центроиду (на основе расстояния).
4. Пересчитать положение центроидов как среднее значение всех объектов, принадлежащих кластеру.
5. Повторять шаги 3 и 4 до тех пор, пока положение центроидов не перестанет изменяться (или пока не будет достигнут предел итераций).

Код реализации K-means в Python

Листинг 4 — Реализация алгоритма K-means

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3)
kmeans.fit(data)
```

```
labels = kmeans.labels_
centroids = kmeans.cluster_centers_
```

2.2. Иерархическая кластеризация

Описание алгоритма

Иерархическая кластеризация — это метод, который не требует заранее задавать количество кластеров. Он работает по принципу создания “дерева кластеров” (дендрограммы), начиная с каждого объекта как отдельного кластера и постепенно объединяя кластеры на основании их сходства.

Два основных подхода

- Агломеративная кластеризация: Начинаем с того, что каждый объект — это отдельный кластер, и постепенно объединяем наиболее близкие кластеры;
- Дивизионная кластеризация: Начинаем с одного большого кластера, который постепенно делим на более мелкие кластеры.

Шаги агломеративной кластеризации

1. Рассчитать расстояния между всеми парами объектов (евклидово или манхэттенское расстояние).
2. Найти две ближайшие точки (кластера) и объединить их в один кластер.
3. Повторять объединение кластеров до тех пор, пока не останется

один общий кластер.

Код реализации иерархической кластеризации в Python

Листинг 5 — Код реализации иерархической кластеризации

```
from scipy.cluster.hierarchy import dendrogram, linkage
import matplotlib.pyplot as plt
Z = linkage(data, method='average')
plt.figure(figsize=(10, 7))
dendrogram(Z)
plt.show()
```

2.3. Алгоритм DBSCAN

Описание алгоритма

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) — это алгоритм кластеризации, основанный на плотности объектов. Он может обнаруживать кластеры любой формы и выделять шумовые данные (то есть объекты, которые не принадлежат никакому кластеру). В отличие от K-means, DBSCAN не требует задавать количество кластеров заранее.

Основные параметры DBSCAN

- eps: Радиус, внутри которого должны находиться объекты, чтобы считаться частью одного кластера;
- min_samples: Минимальное количество объектов в пределах радиуса eps, необходимых для того, чтобы образовать кластер.

Шаги работы DBSCAN

1. Для каждой точки данных определяется число соседей в радиусе eps.
2. Если число соседей больше или равно min_samples, то точка становится “ядром кластера”.
3. Точки, которые находятся в пределах радиуса от ядра, присоединяются к кластеру.
4. Если точка не имеет достаточного количества соседей, она помечается как шум.

Код реализации DBSCAN в Python

Листинг 6 — Код реализации DBSCAN

```
from sklearn.cluster import DBSCAN
dbscan = DBSCAN(eps=0.5, min_samples=5)
dbscan.fit(data)
labels = dbscan.labels_
```

ШАГ 3: ОПРЕДЕЛЕНИЕ КОЛИЧЕСТВА КЛАСТЕРОВ

После реализации алгоритмов кластеризации необходимо определить оптимальное количество кластеров. Для этого на практике часто используют несколько методов, таких как метод локтя и коэффициент силуэта. Эти методы помогают оценить, насколько хорошо данные делятся на кластеры и найти оптимальное число кластеров для K-means и других алгоритмов.

3.1. Метод локтя

Описание метода

Метод локтя используется для нахождения оптимального количества кластеров в алгоритме K-means. Идея состоит в том, чтобы построить график зависимости суммы квадратов ошибок (инерции) от количества кластеров. По мере увеличения числа кластеров инерция уменьшается, так как кластеры становятся всё более точными. Однако после определённого количества кластеров уменьшение инерции замедляется — на графике это выглядит как “локоть”. Этот “локоть” и является оптимальным количеством кластеров.

Шаги для метода локтя:

1. Реализовать K-means для разных значений K (обычно от 1 до 10 или больше).
2. Рассчитать сумму квадратов ошибок (инерцию) для каждого значения K.

3. Построить график инерции и определить точку, где происходит резкое изменение наклона кривой (локоть).

Код для метода локтя в Python

Листинг 7 — Код метода локтя

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
inertia = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(data)
    inertia.append(kmeans.inertia_)
plt.figure(figsize=(8, 6))
plt.plot(range(1, 11), inertia, marker='o')
plt.title('Метод локтя')
plt.xlabel('Количество кластеров')
plt.ylabel('Инерция')
plt.show()
```

3.2. Коэффициент силуэта

Описание метода

Коэффициент силуэта оценивает качество кластеризации, измеряя, насколько близко объекты внутри одного кластера и насколько далеко они от объектов в других кластерах. Значение коэффициента варьируется от -1 до 1:

- Значение, близкое к 1, указывает на то, что объекты хорошо сгруппированы внутри кластера и хорошо отделены от других кластеров;
- Значение, близкое к 0, говорит о том, что объекты находятся на границе между кластерами;
- Отрицательные значения означают, что объекты могут быть неверно

распределены по кластерам.

Шаги для использования коэффициента силуэта

1. Для каждого объекта вычислить коэффициент силуэта.
2. Рассчитать среднее значение коэффициента для всех объектов.
3. Выбрать количество кластеров, при котором средний коэффициент силуэта максимален.

Код для расчёта коэффициента силуэта в Python

Листинг 8 — Код для расчета коэффициента силуэта

```
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
silhouette_scores = []
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(data)
    labels = kmeans.labels_
    silhouette_scores.append(silhouette_score(data, labels))
plt.figure(figsize=(8, 6))
plt.plot(range(2, 11), silhouette_scores, marker='o')
plt.title('Коэффициент силуэта')
plt.xlabel('Количество кластеров')
plt.ylabel('Средний коэффициент силуэта')
plt.show()
```

3.3. Совмещение методов для оптимального выбора количества кластеров

На практике метод локтя и коэффициент силуэта часто используются совместно:

- Метод локтя позволяет увидеть общее поведение инерции по мере

увеличения числа кластеров и оценить, где дальнейшее увеличение K не даёт значительного улучшения;

- Коэффициент силуэта помогает точнее оценить, насколько хорошо разделены кластеры и насколько эффективно они сгруппированы.

ШАГ 4: ВИЗУАЛИЗАЦИЯ ИЕРАРХИЧЕСКОЙ КЛАСТЕРИЗАЦИИ

Иерархическая кластеризация имеет одно из ключевых преимуществ — возможность визуализации процесса объединения данных в кластеры через дендрограмму. Дендрограмма представляет собой дерево, на котором отображаются иерархические связи между объектами и кластерами.

4.1. Что такое дендрограмма?

Дендрограмма — это графическое представление иерархической кластеризации, которое показывает, как кластеры объединяются или разъединяются по мере изменения уровня агрегации. Каждая точка на дендрограмме соответствует объекту данных, а линии соединяют объекты и кластеры на каждом уровне агрегации.

Элементы дендрограммы:

- Ветви (линии): показывают связь между объектами и кластерами;
- Высота соединения: определяет расстояние или различие между кластерами. Чем выше соединение на графике, тем больше различие между объектами или кластерами.

4.2. Построение дендрограммы

Построение дендрограммы — это основной шаг для визуализации

результатов иерархической кластеризации. Используя дендрограмму, студенты могут не только увидеть, как объекты объединяются в кластеры, но и выбрать оптимальное количество кластеров, разрезав дерево на определённом уровне.

Код для построения дендрограммы в Python

Листинг 9 — Код для построения дендрограммы

```
from scipy.cluster.hierarchy import dendrogram, linkage
import matplotlib.pyplot as plt
Z = linkage(data, method='average')
plt.figure(figsize=(10, 7))
dendrogram(Z)
plt.title('Дендрограмма иерархической кластеризации')
plt.xlabel('Объекты')
plt.ylabel('Расстояние')
plt.show()
```

Значение осей дендрограммы:

- X-ось: каждый объект данных;
- Y-ось: расстояние между объединяемыми кластерами (чем выше соединение на дендрограмме, тем более удалены друг от друга объекты);
- Как выбрать количество кластеров: Чтобы выбрать количество кластеров, нужно провести горизонтальную линию через дендрограмму. Количество кластеров будет равно количеству пересечений этой линии с ветвями дендрограммы.

4.3. Варианты методов объединения кластеров (linkage)

В иерархической кластеризации существует несколько подходов к объединению кластеров. В коде выше мы использовали метод “средних

связей” (average linkage), но есть и другие методы:

1. Single linkage (ближайший сосед): объединение кластеров происходит на основании минимального расстояния между двумя объектами разных кластеров.
2. Complete linkage (дальний сосед): объединение кластеров происходит на основании максимального расстояния между объектами разных кластеров.
3. Average linkage (средние связи): объединение происходит на основании среднего расстояния между объектами разных кластеров.
4. Ward’s method (метод Уорда): объединение кластеров минимизирует сумму квадратов отклонений внутри кластеров.

Каждый метод даёт разные результаты и может быть выбран в зависимости от характера данных.

Пример использования другого метода (complete linkage)

Листинг 10 — Использование complete linkage

```
Z = linkage(data, method='complete')
plt.figure(figsize=(10, 7))
dendrogram(Z)
plt.title('Дендрограмма (Complete Linkage)')
plt.show()
```

4.4. Анализ дендрограммы

1. *Оценить расстояния между кластерами*: чем больше расстояние между кластерами, тем более различны они между собой.
2. *Выбрать количество кластеров*: чем ниже точка разреза на дендрограмме, тем больше будет количество кластеров.

Пример анализа

Если на дендрограмме наблюдается резкий скачок по оси Y, это может указывать на естественное разбиение данных на кластеры. Если несколько ветвей объединяются на высоком уровне, это означает, что данные на этих уровнях менее схожи и могут быть объединены только при больших значениях расстояний.

ШАГ 5: ПОСТРОЕНИЕ ГРАФИКОВ

После реализации кластеризации важно визуально оценить результаты. Визуализация — ключевой инструмент, который помогает понять, как алгоритмы разделяют данные и насколько чётко выделяются кластеры.

5.1. Построение графиков для K-means

2D Визуализация кластеров

Для визуализации результатов K-means кластеризации можно построить графики распределения данных в пространстве признаков, где объекты будут раскрашены в зависимости от их принадлежности к определённому кластеру.

Если данные многомерные (содержат более двух признаков), можно использовать метод уменьшения размерности, например PCA (Principal Component Analysis), чтобы отобразить их на плоскости.

Код для визуализации кластеров K-means в Python

Листинг 11 — Визуализация кластеров K-means

```
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
pca = PCA(n_components=2)
pca_data = pca.fit_transform(data)
kmeans = KMeans(n_clusters=3)
kmeans.fit(data)
labels = kmeans.labels_
plt.figure(figsize=(8, 6))
plt.scatter(pca_data[:, 0], pca_data[:, 1], c=labels,
cmap='viridis', marker='o')
plt.title('Результаты K-means кластеризации')
```

```
plt.xlabel('Первая главная компонента')
plt.ylabel('Вторая главная компонента')
plt.show()
```

Пояснение:

- На графике точки данных раскрашены по кластерам. Результаты можно интерпретировать следующим образом: объекты одного цвета принадлежат одному кластеру;
- Применение метода PCA помогает упростить визуализацию многомерных данных, отображая их в двух измерениях.

5.2. Построение графиков для иерархической кластеризации

Для иерархической кластеризации график дендрограммы (построенный на предыдущем шаге) также можно рассматривать как способ визуализации кластеров. Однако, если нужно отобразить результаты разбиения данных на конкретное количество кластеров, можно использовать график, подобный тому, который применяли для K-means.

Пример визуализации результатов иерархической кластеризации

Листинг 12 — Визуализация результатов иерархической кластеризации

```
from scipy.cluster.hierarchy import fcluster
clusters = fcluster(Z, 3, criterion='maxclust')
plt.figure(figsize=(8, 6))
plt.scatter(pca_data[:, 0], pca_data[:, 1], c=clusters,
cmap='plasma', marker='o')
plt.title('Результаты иерархической кластеризации')
plt.xlabel('Первая главная компонента')
plt.ylabel('Вторая главная компонента')
plt.show()
```

Пояснение:

- В этом примере использован метод fcluster, который позволяет выделить конкретное количество кластеров из дендрограммы;
- Аналогично K-means, точки данных раскрашены в зависимости от их кластера, что позволяет визуально оценить, насколько хорошо разделились группы.

5.3. Построение графиков для DBSCAN

В отличие от K-means, алгоритм DBSCAN может выделять шумовые данные (объекты, которые не принадлежат ни одному кластеру). При построении графиков для DBSCAN важно отдельно визуализировать шум и кластеры.

Код для визуализации результатов DBSCAN в Python

Листинг 13 — Визуализация результатов DBSCAN

```
from sklearn.cluster import DBSCAN
dbscan = DBSCAN(eps=0.5, min_samples=5)
dbscan.fit(data)
labels = dbscan.labels_
plt.figure(figsize=(8, 6))
plt.scatter(pca_data[:, 0], pca_data[:, 1], c=labels,
cmap='rainbow', marker='o')
plt.title('Результаты DBSCAN кластеризации')
plt.xlabel('Первая главная компонента')
plt.ylabel('Вторая главная компонента')
plt.show()
```

Пояснение:

- В этом случае точки данных также окрашены в зависимости от кластера, но шумовые точки (те, что не принадлежат ни одному кластеру) могут иметь отдельный цвет (обычно чёрный или -1 по

метке);

- DBSCAN хорошо работает с кластерами разной формы, и на графике это может проявиться в виде нерегулярных групп точек.

5.4. Дополнительные графики для оценки кластеров

Также можно построить дополнительные графики для анализа качества кластеризации:

1. График зависимости инерции от числа кластеров (метод локтя): используется для оценки оптимального числа кластеров в K-means.
2. График зависимости коэффициента силуэта от числа кластеров: помогает оценить, насколько хорошо данные разделены на кластеры.

Пример графика силуэта для K-means

Листинг 14 — Пример графика силуэта для K-means

```
from sklearn.metrics import silhouette_samples, silhouette_score
import numpy as np
silhouette_vals = silhouette_samples(data, labels)
plt.figure(figsize=(8, 6))
plt.barh(range(len(silhouette_vals)), silhouette_vals)
plt.title('График силуэта для кластеризации K-means')
plt.xlabel('Коэффициент силуэта')
plt.ylabel('Объекты')
plt.show()
```

Пояснение:

- График силуэта показывает, насколько хорошо каждый объект вписывается в свой кластер;
- Если большинство значений коэффициента силуэта близки к 1, это значит, что кластеры хорошо разделены.

ШАГ 6: СРАВНЕНИЕ РЕЗУЛЬТАТОВ

После того как были выполнены все алгоритмы кластеризации, важно провести сравнительный анализ их результатов. Сравнение поможет студентам понять, какой алгоритм лучше справляется с задачей на данном наборе данных и какие кластеры выделяются наиболее чётко. В этом шаге мы будем использовать количественные и визуальные методы для оценки качества кластеров.

6.1. Сравнение по коэффициенту силуэта

Коэффициент силуэта помогает оценить, насколько хорошо данные сгруппированы в кластеры. Для каждого алгоритма нужно рассчитать среднее значение коэффициента силуэта и сравнить результаты.

Код для расчёта среднего коэффициента силуэта для разных алгоритмов

Листинг 15 — Расчет среднего коэффициента силуэта для разных алгоритмов

```
from sklearn.metrics import silhouette_score
from sklearn.cluster import KMeans, DBSCAN

# Для K-means
kmeans = KMeans(n_clusters=3)
kmeans.fit(data)
kmeans_labels = kmeans.labels_
kmeans_silhouette = silhouette_score(data, kmeans_labels)

# Для иерархической кластеризации
from scipy.cluster.hierarchy import fcluster
hierarchical_labels = fcluster(Z, 3, criterion='maxclust')
hierarchical_silhouette = silhouette_score(data,
hierarchical_labels)
```

```
# Для DBSCAN
dbscan = DBSCAN(eps=0.5, min_samples=5)
dbscan.fit(data)
dbscan_labels = dbscan.labels_
dbscan_silhouette = silhouette_score(data, dbscan_labels)

# Вывод средних коэффициентов силуэта для каждого алгоритма
print(f"K-means silhouette: {kmeans_silhouette}")
print(f"Hierarchical silhouette: {hierarchical_silhouette}")
print(f"DBSCAN silhouette: {dbscan_silhouette}")
```

Пояснение:

- Алгоритм с наибольшим средним значением коэффициента силуэта считается наиболее успешным в разбиении данных на кластеры. Это говорит о том, что объекты внутри каждого кластера более компактны и лучше отделены от других кластеров;
- Можно увидеть, что разные алгоритмы могут по-разному справляться с кластеризацией в зависимости от структуры данных.

6.2. Сравнение по визуализации кластеров

Кроме количественных метрик, важно визуально оценить результаты кластеризации. Студенты уже построили графики для каждого алгоритма на предыдущих шагах, и теперь можно сравнить эти графики, чтобы увидеть:

- Четкость разделения кластеров;
- Форма кластеров;
- Шумовые данные.

Пример вопросов для анализа

1. Как распределены объекты на графиках K-means и DBSCAN?
2. Какие различия в результатах кластеризации дают разные

алгоритмы?

3. Есть ли шумовые данные, которые один алгоритм выделяет, а другой нет?

6.3. Сравнение на основе устойчивости к параметрам

K-means и иерархическая кластеризация чувствительны к выбору количества кластеров. Если студент выбрал слишком большое или слишком маленькое количество кластеров, это может повлиять на результат. Для K-means это можно наблюдать на графике локтя.

DBSCAN, напротив, более чувствителен к параметрам eps и min_samples. Если eps слишком мал, то алгоритм может выделить слишком много кластеров или считать большинство точек шумом. Если eps слишком велик, то кластеры могут объединяться в один.

Пример тестирования устойчивости

Листинг 16 — Тестирование устойчивости

```
dbscan_1 = DBSCAN(eps=0.3, min_samples=5)
dbscan_1.fit(data)
dbscan_silhouette_1 = silhouette_score(data, dbscan_1.labels_)
dbscan_2 = DBSCAN(eps=0.7, min_samples=5)
dbscan_2.fit(data)
dbscan_silhouette_2 = silhouette_score(data, dbscan_2.labels_)
print(f"Silhouette for DBSCAN with eps=0.3:
{dbscan_silhouette_1}")
print(f"Silhouette for DBSCAN with eps=0.7:
{dbscan_silhouette_2}")
```

Пояснение:

- Можно проанализировать, как изменяются результаты алгоритмов при изменении их параметров, и оценить, какой из алгоритмов более устойчив к этим изменениям;

- Это обучает внимательному подбору гиперпараметров алгоритмов для достижения наилучших результатов.

6.4. Выводы на основе сравнения

Студенты должны сделать выводы на основе:

- Средних коэффициентов силуэта;
- Визуальных различий в графиках кластеров;
- Устойчивости алгоритмов к параметрам.

Пример выводов

1. Если алгоритм K-means показал наилучшие результаты по силуэту и на графиках, он является предпочтительным для данной задачи.
2. Если иерархическая кластеризация выделила чёткие кластеры, но силуэт оказался ниже, это может указывать на то, что метод лучше подходит для сложных данных, но требует ручного анализа.
3. DBSCAN может хорошо справляться с шумом, однако его чувствительность к параметрам может привести к неоднозначным результатам.