**ФГБОУ ВО   
Уфимский университет науки и технологий**

**Кафедра ВМиК**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 100 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 90 |  |  |  |  |  |  |  |  |  |  |  |  |
| 80 |  |  |  |  |  |  |  |  |  |  |  |  |
| 70 |  |  |  |  |  |  |  |  |  |  |  |  |
| 60 |  |  |  |  |  |  |  |  |  |  |  |  |
| 50 |  |  |  |  |  |  |  |  |  |  |  |  |
| 40 |  |  |  |  |  |  |  |  |  |  |  |  |
| 30 |  |  |  |  |  |  |  |  |  |  |  |  |
| 20 |  |  |  |  |  |  |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

Оценка качества методов кластеризации

**ОТЧЕТ**

**по лабораторной работе**

**по** Основам машинного обучения

(*наименование дисциплины*)

|  |
| --- |
| Лабораторная работа 3 |
| (обозначение документа) |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Группа |  |  | Фамилия, И., О. | Подпись | Дата | Оценка |
| МО-325Б |  |
|  |  |
| Студент | | | Шарыгин М.С.,  Агафонов Р.В.,  Лепоринский Г.А. |  |  |  |
| Преподаватель | | | Миронов К.В. |  |  |  |
| Принял | | |  |  |  |  |

**Уфа 2025 г****.**

Содержание

[1 Цель работы 3](#_Toc194710022)

[2 Практическая часть 4](#_Toc194710023)

[2.1 Подготовка датасета 4](#_Toc194710024)

[2.2 Подготовка таблиц и массивов 5](#_Toc194710025)

[2.3 Дополнительные функции 5](#_Toc194710026)

[2.4 Метод k-средних / «kMeans» 7](#_Toc194710027)

[2.5 Метод сдвига среднего значения / «MeanShift» 8](#_Toc194710028)

[2.6 Метод агломеративной кластеризации / «AgglomerativeClustering» 9](#_Toc194710029)

[2.7 Метод пространственной кластеризации для приложений с шумами / «DBSCAN» 10](#_Toc194710030)

[2.8 Сравнение результатов 10](#_Toc194710031)

[3 Вывод 12](#_Toc194710032)

[4 Приложение 13](#_Toc194710033)

[4.1 Код программы 13](#_Toc194710034)

# Цель работы

В ходе лабораторной работы необходимо исследовать следующие методы кластеризации при разных размерностях и различных гиперпараметрах:

* Метод k-средних / kMeans;
* Метод сдвига среднего значения / MeanShift;
* Метод агломеративной кластеризации / AgglomerativeClustering;
* Метод пространственной кластеризации для приложений с шумами / DBSCAN.

# Практическая часть

## Подготовка датасета

Для начала импортируем и нормализуем датасет, а также заполним таблицу собственных векторов и значений так, как показано на рисунке 2.1[[1]](#footnote-1).

Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.1 – Подготовка датасета

## Подготовка таблиц и массивов

Затем необходимо модулем «PrettyTable()» создать таблицы для хранения результатов выполнения каждого метода, в которые с помощью «field\_names» добавим колонки; также для каждого метода и лучших результатов нужно создать массивы, которые будут использованы для хранения лучших результатов (рисунок 2.2).

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.2 – Создание таблиц и массивов

## Дополнительные функции

Для начала реализуем функцию «search\_for\_best\_results», которая будет принимать в себя следующие параметры:

* «table» – таблица, в которую будут записаны полученные на данном шаге результаты;
* «best» – массив, хранящий лучший результат до текущего шага;
* «dimension» – размерность пространства на текущем шаге;
* «ari» – индекс Рэнда, который вычисляет меру сходства между известными и полученными классами, рассматривая все пары образцов и подсчитывая пары, которые отнесены к одним и тем же или разным кластерам;
* «accuracy» – полученная точность при заданных параметрах;
* «period» – время выполнения классификации;
* «parameter\_1» – первый гиперпараметр;
* «parameter\_2» – второй гиперпараметр.

Она [функция] будет сравнивать результаты текущего шага с лучшими и если текущие окажутся лучше, то они будут сохранены на их месте [месте лучших] и добавлены в соответствующую таблицу своего метода, как показано на рисунке 2.3.

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.3 – Функция «search\_for\_best\_results»

Теперь создадим функцию «definition\_of\_accuracy», которая будет подсчитывать точность путем деления верных ответов на количество объектов (рисунок 2.4).

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.4 – Функция «definition\_of\_accuracy»

Наконец реализуем функцию «table\_best\_formation», добавляющую лучшие результаты в соответствующую таблицу. Функция приведена на рисунке 2.5.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.5 – Функция «table\_best\_formation»

## Метод k-средних / «kMeans»

Сначала запустим цикл «for», который будет задавать размерность пространства от одного до всех параметров. После этого применим метод «PCA» для сокращения размерности пространства – это выполняются однажды для всех методов сразу (рисунок 2.6).

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.6 – Сокращение размерности и разделение выборки

Теперь реализуем метод k-средних **–** это алгоритм кластеризации, который разделяет данные на «k» кластеров, минимизируя суммарное квадратичное расстояние точек до центроидов[[2]](#footnote-2) их кластеров. На каждом шаге он пересчитывает центроиды и перераспределяет точки, пока они [центроиды] не стабилизируются. Метод прост в реализации, но требует заранее задавать число кластеров, которое и является основным гиперпараметром. На рисунке 2.7 представлена реализация метода, а на рисунке 2.8 – наилучшие результаты.

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Графическое программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.7 – Метод k-средних

Изображение выглядит как снимок экрана, текст

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.8 – Лучшие результаты

## Метод сдвига среднего значения / «MeanShift»

Метод сдвига среднего значения – алгоритм кластеризации на основе плотности, который автоматически определяет число кластеров, итеративно сдвигая точки в направлении наибольшего увеличения плотности данных. Он использует основной гиперпараметр «bandwidth» (радиус окрестности) для оценки плотности и не требует предварительного задания количества кластеров. Алгоритм эффективен для данных сложной формы, но требует тщательного подбора радиуса окрестности, а также вычислительно затратен для больших наборов данных. Реализация метода и его результаты представлены на рисунках 2.9 и 2.10 соответственно.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.9 – Метод сдвига среднего значения

Изображение выглядит как снимок экрана, текст, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.10 – Лучшие результаты

## Метод агломеративной кластеризации / «AgglomerativeClustering»

Метод агломеративной кластеризации – иерархический алгоритм кластеризации "снизу вверх", который последовательно объединяет ближайшие пары кластеров, пока не останется заданное число кластеров – основной гиперпараметр. Метод хорошо работает с данными сложной структуры, но требует вычислительных ресурсов для больших наборов данных и задания количества кластеров (рисунки 2.11 и 2.12).

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Графическое программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.11 – Метод агломеративной кластеризации

Изображение выглядит как снимок экрана, текст

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.12 – Лучшие результаты

## Метод пространственной кластеризации для приложений с шумами / «DBSCAN»

Метод пространственной кластеризации для приложений с шумами – это алгоритм кластеризации на основе плотности, который группирует точки, находящиеся в зоне заданного радиуса «eps», при условии минимального числа соседей «min\_samples». Он автоматически определяет число кластеров и хорошо работает с данными произвольной формы, помечая выбросы как шум. Однако его эффективность сильно зависит от правильного выбора гиперпараметров «eps» и «min\_samples», а также от масштабирования данных. Реализация и результаты метода приведены на рисунках 2.13 и 2.14 соответственно.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.13 – Метод пространственной кластеризации для приложений с шумами

Изображение выглядит как снимок экрана, текст, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.14 – Лучшие результаты

## Сравнение результатов

После выполнения всех методов добавим информацию о лучших результатах каждого метода в сводную таблицу (каждая строка в этой таблице – это наилучший результат по точности и времени выполнения при различных размерностях и гиперпараметрах), как показано на рисунках 2.21 и 2.22.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.21 – Добавлений строк с лучшими результатами в сводную таблицу

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.22 – Таблица лучших результатов

Большинством методов были получены примерно одинаковые точности (0.002 – 0.004), исключение составил «DBSCAN», точность которого составила ~ 0.026. При этом время выполнения всех методов в секундах составило 0.001 – 0.002, и только «AgglomerativeClustering» был выполнен за 0.6 секунд.

Таким образом, можно отметить, что из-за особенности выборки данных в данном датасете наилучшим методом стал «DBSCAN».

# Вывод

В ходе лабораторной работы мы исследовали методы кластеризации при разной размерности и различных гиперпараметрах.

# Приложение

## Код программы

from ucimlrepo import fetch\_ucirepo  
from sklearn import preprocessing  
from sklearn.decomposition import PCA  
from sklearn.cluster import KMeans**,** MeanShift**,** AgglomerativeClustering**,** DBSCAN  
from sklearn.metrics import adjusted\_rand\_score  
from prettytable import PrettyTable  
import numpy as np  
import time  
  
  
  
# Импорт датасета  
wholesale\_customers = fetch\_ucirepo(id=**292**)  
x = wholesale\_customers.data.features  
y = wholesale\_customers.data.targets  
  
  
  
# Нормализация датасета  
x\_train = np.array(x)  
x\_scaled = preprocessing.StandardScaler().fit\_transform(x\_train)  
y = np.array(y).ravel()  
  
  
  
# Заполнение таблицы собственных векторов и значений  
pca = PCA()  
X = pca.fit\_transform(x\_scaled)  
eigenvectors = pca.fit\_transform(X)  
eigenvalues = pca.explained\_variance\_  
  
  
  
# Создание таблиц для представления результатов  
table\_kmeans = PrettyTable()  
table\_ms = PrettyTable()  
table\_ac = PrettyTable()  
table\_dbscan = PrettyTable()  
table\_best = PrettyTable()  
  
# Добавление колонок в таблицы  
table\_kmeans.field\_names = ["Размерность"**,** "ARI"**,** "Точность"**,** "Время выполнения"**,** "n\_clusters"**,** "-" ]  
table\_ms.field\_names = ["Размерность"**,** "ARI"**,** "Точность"**,** "Время выполнения"**,** "bandwidth" **,** "-" ]  
table\_ac.field\_names = ["Размерность"**,** "ARI"**,** "Точность"**,** "Время выполнения"**,** "n\_clusters"**,** "-" ]  
table\_dbscan.field\_names = ["Размерность"**,** "ARI"**,** "Точность"**,** "Время выполнения"**,** "eps" **,** 'min\_samples']  
table\_best.field\_names = ["Размерность"**,** "ARI"**,** "Точность"**,** "Время выполнения"**,** "Параметр 1"**,** "Параметр 2" ]  
  
# Создание массивов для поиска лучших результатов  
best\_kmeans = [**0, 0, 0, 0,** ""**,** ""]  
best\_ms = [**0, 0, 0, 0,** ""**,** ""]  
best\_ac = [**0, 0, 0, 0,** ""**,** ""]  
best\_dbscan = [**0, 0, 0, 0,** ""**,** ""]  
  
  
  
def search\_for\_best\_results (table**,** best**,** dimension**,** ari**,** accuracy**,** period**,** parameter\_1**,** parameter\_2):  
 if (best[**1**] < ari) or (best[**1**] == ari and best[**3**] > period):  
 best[**0**] = dimension  
 best[**1**] = ari  
 best[**2**] = accuracy  
 best[**3**] = period  
 best[**4**] = parameter\_1  
 best[**5**] = parameter\_2  
  
 table.add\_row([dimension**,** ari**,** accuracy**,** period**,** parameter\_1**,** parameter\_2])  
  
  
  
def definition\_of\_accuracy(true**,** predict):  
 count = **0** for i in range(len(true)):  
 if true[i] == predict[i]:  
 count += **1** return count / len(true)  
  
  
  
def table\_best\_formation(table**,** best):  
 table.add\_row([best[**0**]**,** best[**1**]**,** best[**2**]**,** best[**3**]**,** best[**4**]**,** best[**5**]])  
  
  
  
# Цикл по всем размерностям  
for component in range(**1,** len(eigenvalues) + **1**):  
 pca = PCA(n\_components=component)  
 X = pca.fit\_transform(x\_scaled)  
  
  
  
 # Метод k-средних / kMeans  
 for k in range(**2, 10**):  
 kmeans = KMeans(n\_clusters=k)  
  
 start\_time = time.time()  
 kmeans.fit(X)  
 end\_time = time.time()  
  
 ARI = adjusted\_rand\_score(y**,** kmeans.labels\_)  
 acc = definition\_of\_accuracy(y**,** kmeans.labels\_)  
  
 search\_for\_best\_results(table\_kmeans**,** best\_kmeans**,** component**,** ARI**,** acc**,** end\_time - start\_time**,** k**,** "")  
  
  
  
 # Метод сдвига среднего значения / Mean shift  
 for b in range(**2, 10**):  
 ms = MeanShift(bandwidth=b)  
  
 start\_time = time.time()  
 ms.fit(X)  
 end\_time = time.time()  
  
 ARI = adjusted\_rand\_score(y**,** ms.labels\_)  
 acc = definition\_of\_accuracy(y**,** ms.labels\_)  
  
 search\_for\_best\_results(table\_ms**,** best\_ms**,** component**,** ARI**,** acc**,** end\_time - start\_time**,** b**,** "")  
  
  
  
 # Метод агломеративной кластеризации / AgglomerativeClustering  
 for k in range(**2, 10**):  
 ac = AgglomerativeClustering(n\_clusters=k)  
  
 start\_time = time.time()  
 ac.fit(X)  
 end\_time = time.time()  
  
 ARI = adjusted\_rand\_score(y**,** ac.labels\_)  
 acc = definition\_of\_accuracy(y**,** ac.labels\_)  
  
 search\_for\_best\_results(table\_ac**,** best\_ac**,** component**,** ARI**,** acc**,** end\_time - start\_time**,** k**,** "")  
  
  
  
 # Метод пространственной кластеризации для приложений с шумами / DBSCAN  
 for min\_samples in range(**3, 10**):  
 for eps in range(**1, 50**):  
 e = eps / **10** dbscan = DBSCAN(eps=e**,** min\_samples=min\_samples)  
  
 start\_time = time.time()  
 dbscan.fit(X)  
 end\_time = time.time()  
  
 ARI = adjusted\_rand\_score(y**,** dbscan.labels\_)  
 acc = definition\_of\_accuracy(y**,** dbscan.labels\_)  
  
 search\_for\_best\_results(table\_dbscan**,** best\_dbscan**,** component**,** ARI**,** acc**,** end\_time - start\_time**,** e**,** min\_samples)  
  
  
  
table\_best\_formation(table\_best**,** best\_kmeans)  
table\_best\_formation(table\_best**,** best\_ms)  
table\_best\_formation(table\_best**,** best\_ac)  
table\_best\_formation(table\_best**,** best\_dbscan)  
  
  
  
print("Метод k-средних / kMeans")  
print(table\_kmeans)  
  
print("\nМетод сдвига среднего значения / Mean shift")  
print(table\_ms)  
  
print("\nМетод агломеративной кластеризации / AgglomerativeClustering")  
print(table\_ac)  
  
print("\nМетод пространственной кластеризации для приложений с шумами / DBSCAN")  
print(table\_dbscan)  
  
print("\nЛучшие результаты")  
print(table\_best)

1. Импорт, нормализация и заполнение были подробно разобраны в лабораторной работе 1. [↑](#footnote-ref-1)
2. Центроид – средняя точка кластера [↑](#footnote-ref-2)