**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Российский экономический университет имени Г.В. Плеханова»**

**Высшая школа кибертехнологий, математики и статистики**

**Кафедра Прикладной информатики и информационной безопасности**

**О Т Ч Е Т**

**по выполнению практического задания 13-17**

**Направление 09.03.03 Прикладная информатика**

**Профиль: Инжиниринг предприятий и информационных систем**

**Уровень: Бакалавриат**

**Дисциплина:** **Структуры данных и алгоритмы**

Выполнил: студент гр. 15.27Д-ПИ05/25б

1 курс

Козачок Мария Геннадьевна

Москва

2025

**Машинное обучение**

1. **K-Means Clustering**
2. Краткое описание области определения алгоритма

**K-Means**- алгоритм **кластеризации** в машинном обучении, который разделяет набор данных на k групп (кластеров) по принципу схожести объектов.

**Основные применения:**

- Сегментация данных на группы

- Анализ и визуализация данных

- Сжатие информации

- Предобработка данных для других алгоритмов

Начало



Вывести: 1)Назначения кластеров 2)Финальные центроиды

Создать новые центроиды как среднее точек в кластере

Присвоить каждый объект к ближайшему центроиду

Инициализировать К центроидов случайным образом

Ввод данных Х и параметра К

Центроиды изменились?

Конец

1. Пошаговый анализ работы алгоритма

### **Шаг 1 Инициализация:** Выбираем число k (сколько кластеров нужно), случайно выбираем k точек из данных как начальные центры

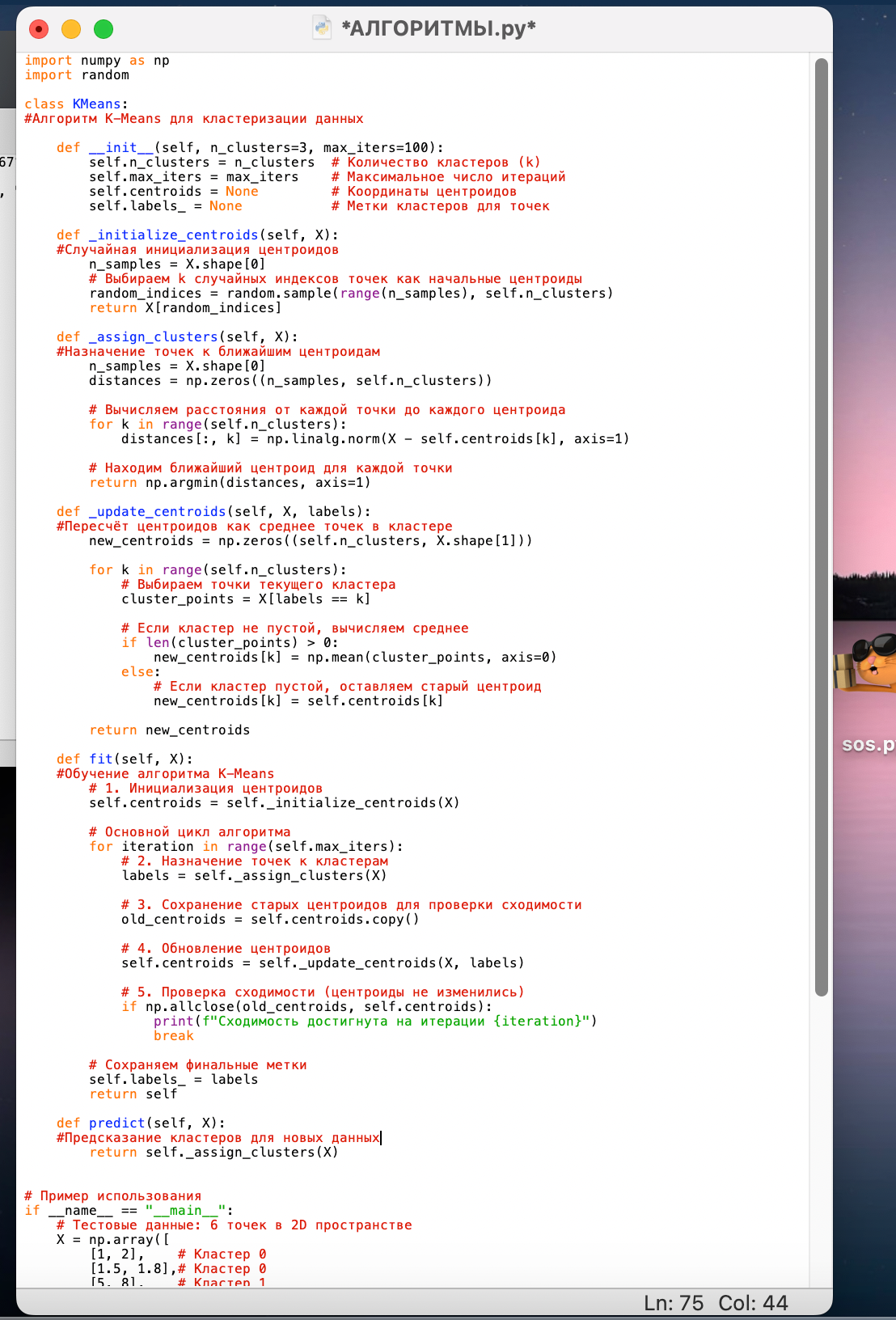
### **Шаг 2 Назначение кластеров.** Для каждой точки: Вычисляем расстояние до всех k центроидов, назначаем точке кластер с ближайшим центроидом

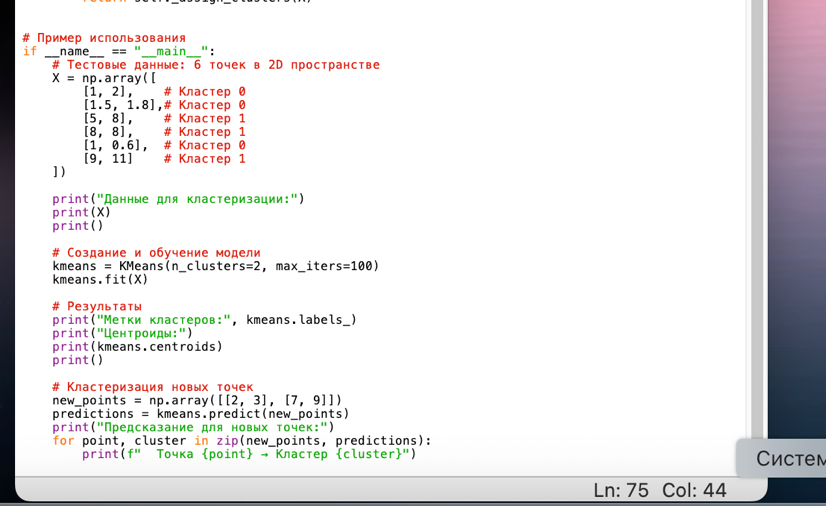
### **Результат:** Все точки разделены на k групп

### **Шаг 3 Обновление центроидов.** Для каждого кластера: Находим среднюю точку всех его объектов, делаем эту точку новым центроидом **Результат:** Центры кластеров сдвигаются

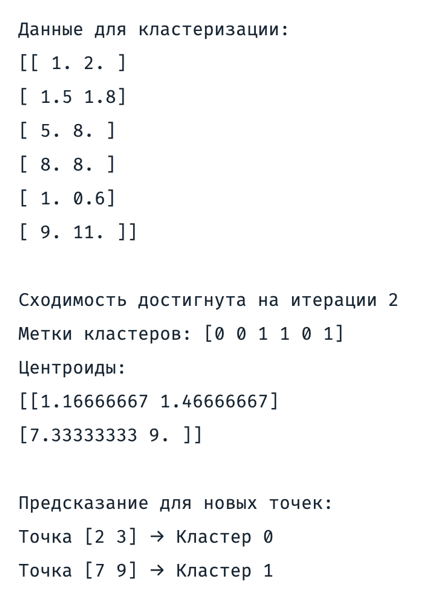
### **Шаг 4 Проверка сходимости:** Если центроиды изменились → вернуться к Шагу 2, Если не изменились → алгоритм завершён

**Критерий остановки:** Центроиды стабилизировались

1. Программный код алгоритма (Python)



**Вывод:**



1. Оценка временной сложности

Сложность: O(n × k × d × i), где:

- n — количество точек данных

- k — количество кластеров

- d — размерность данных (количество признаков)

- i — количество итераций до сходимости

Объяснение:

1. Назначение кластеров (O(n × k × d)):  
   Для каждой из n точек вычисляем расстояние до каждого из k центроидов в d-мерном пространстве.
2. Обновление центроидов (O(n × d)):  
   Для каждого кластера вычисляем среднее значение всех его точек.
3. Повторение i раз:  
   Шаги 1-2 повторяются i раз до сходимости (обычно i ≤ 100).
4. **DBSCAN**
5. Краткое описание области определения алгоритма

**DBSCAN** используется в **машинном обучении** для кластеризации данных на основе плотности. Способен находить кластеры произвольной формы и отделять выбросы (шум).

начало

Для каждой точки

Уже посещена

да

конец

нет

Найти соседей в радиусе eps

Соседей >=MinPts?

Нет да

Создать новый кластер

Пометить как шум

Расширить кластер

Добавить достижимые точки

1. - Выбрать произвольную не посещённую точку.

- Найти все точки в радиусе eps.

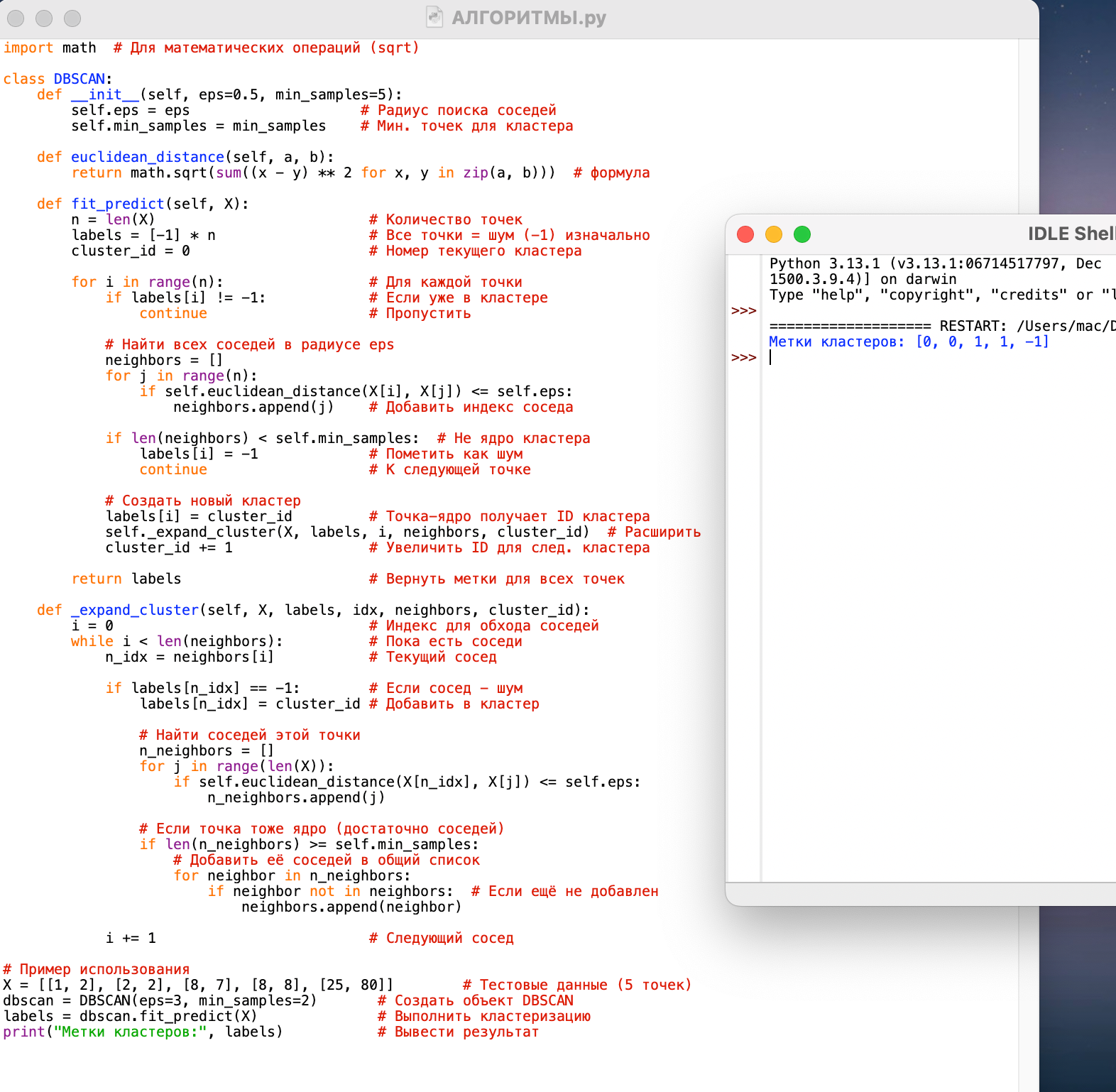
- Если точек меньше MinPts - пометить как шум.

- Иначе создать новый кластер и добавить все точки из eps-окрестности.

- Рекурсивно расширять кластер, добавляя точки, у которых тоже достаточно соседей.

- Повторять, пока все точки не будут обработаны.

4.



5.

O(n²) для каждой из n точек мы вычисляем расстояния до всех остальных n точек → n × n = n² операций.

**С оптимизацией (KD-дерево/индексы):** O(n log n)  
Индекс ускоряет поиск соседей с O(n) до O(log n) на точку → n × log n

1. **Affinity Propagation**
2. Краткое описание области определения алгоритма

**Affinity Propagation**— алгоритм **кластеризации** (обучения без учителя) в машинном обучении, который автоматически определяет количество кластеров на основе **обмена сообщениями** междуточками данных.

В отличие от K-Means, не требует задания количества кластеров заранее.

Начало



Сходимость достигнута?

Цикл итераций

Определить exemplars

Назначить точки к exemplars

Обновить доступность A

Обновить ответственность R

Инициализировать матрицы R и A нулями

Вычислить матрицу сходства S

Конец, вернуть клатеры

1. Пошаговый анализ работы алгоритма

**Шаг 1: Инициализация**

- Вычисление матрицы сходства S(i,j) (обычно отрицательный квадрат расстояния)

- Установка предпочтений на диагонали (каждая точка "хочет" быть exemplar)

**Шаг 2: Обмен сообщениями (итеративный процесс)**

**- Ответственность R(i,k):**

- Сообщение от точки i к точке k: "Насколько ты хорош как мой exemplar?"

- R(i,k) = S(i,k) - max{a(i,k') + S(i,k')} для всех k' ≠ k

**- Доступность A(i,k):**

- Сообщение от точки k к точке i: "Насколько я доступен как твой exemplar?"

- A(i,k) = min{0, R(k,k) + Σ max{0, R(i',k)} для всех i' ∉ {i,k}}

**Шаг 3: Определение exemplars**

- Точка k становится exemplar, если R(k,k) + A(k,k) > 0

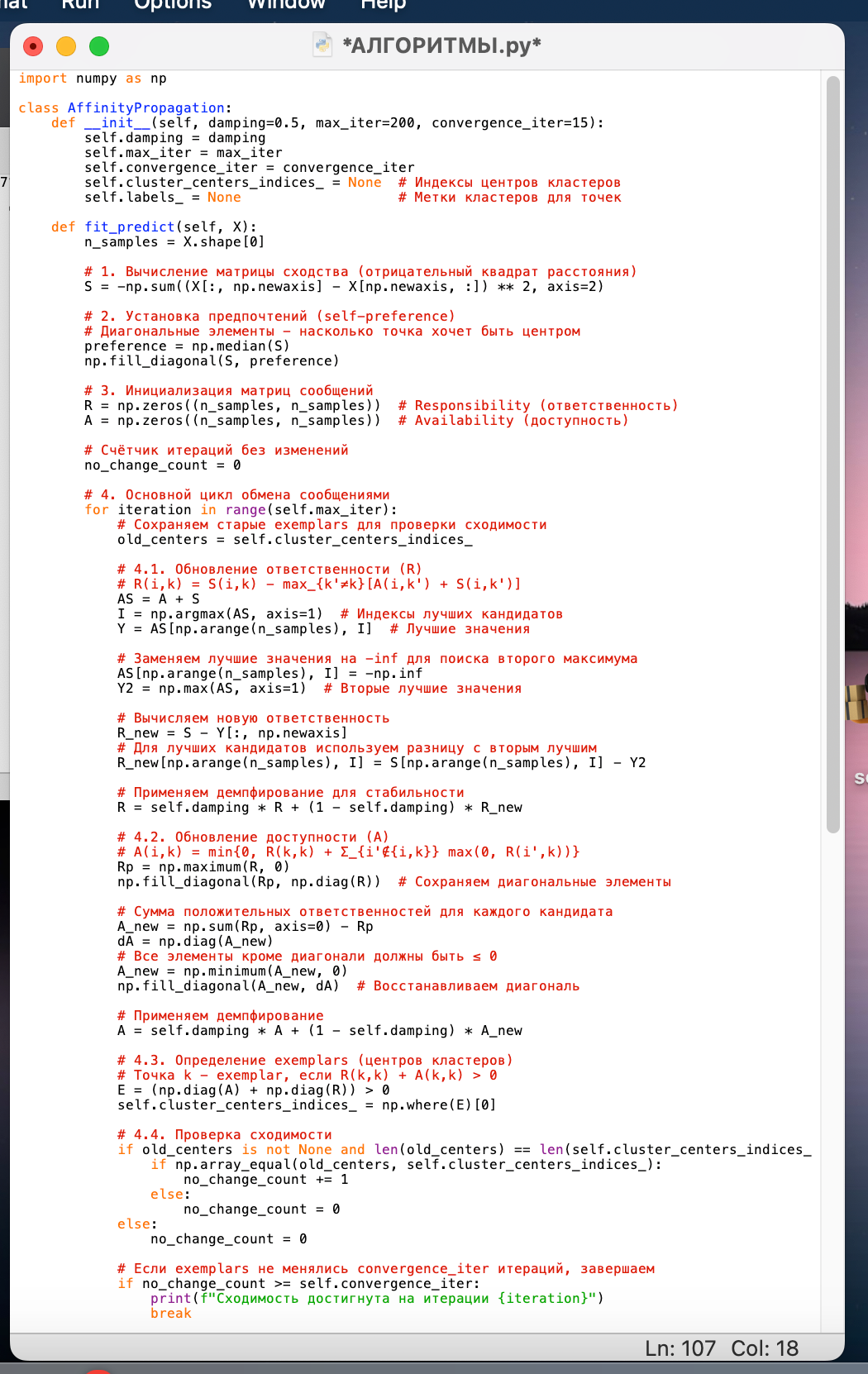
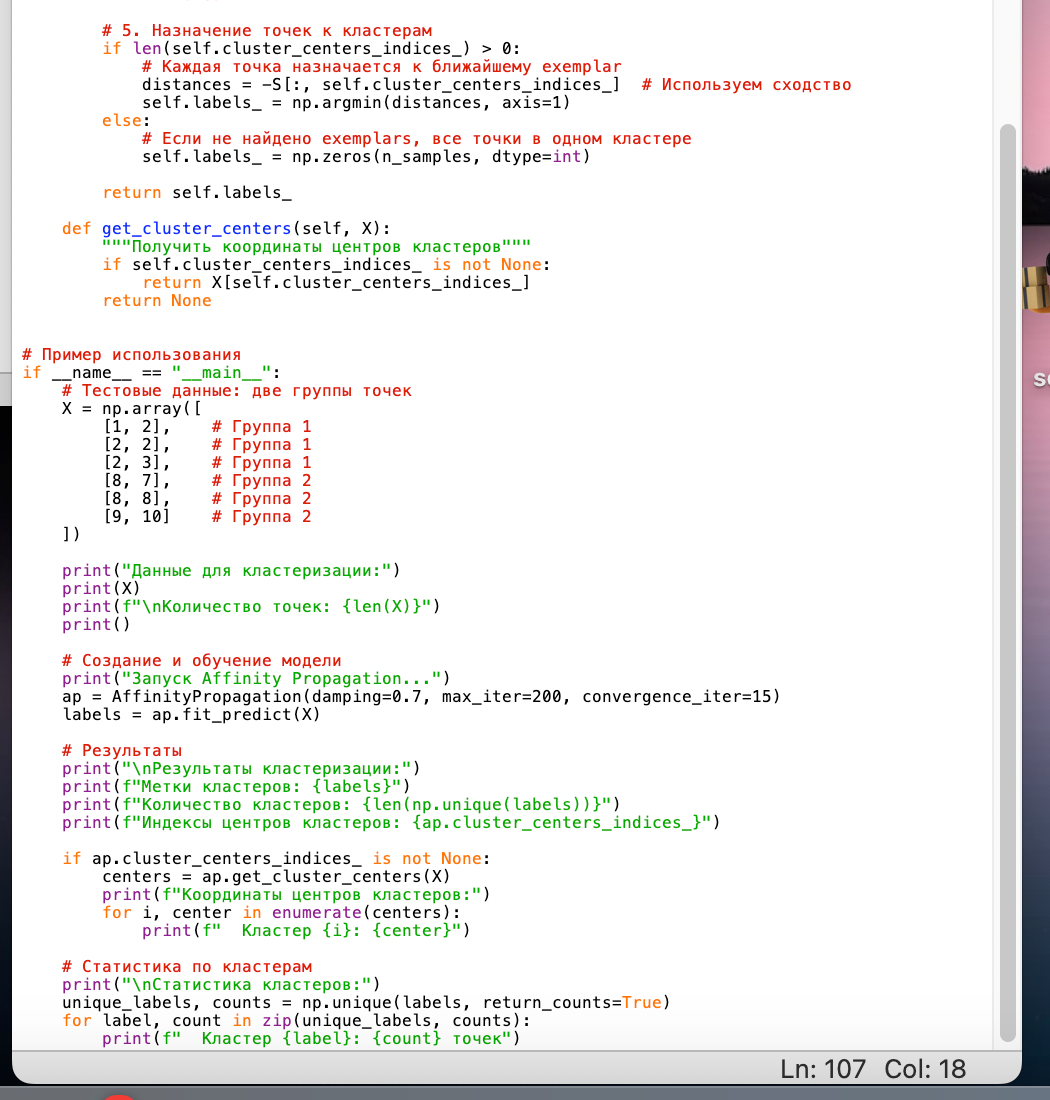
- Exemplars — это точки-представители кластеров

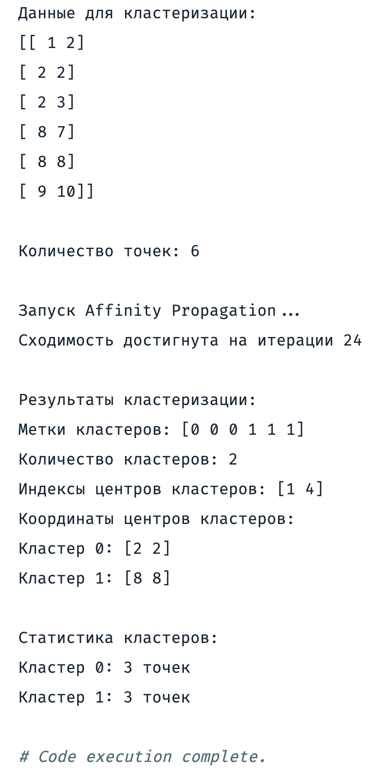
**Шаг 4: Назначение кластеров**

- Каждая точка назначается к ближайшему exemplar

- Метод завершается, когда exemplars перестают меняться

1. Программный код алгоритма (Python)



вывод:

1. Оценка временной сложности

**Сложность: O(n² × t),** где: **n** — количество точек данных, **t** — количество итераций до сходимости

**Объяснение:**

**- Вычисление матрицы сходства: O(n² × d)**

- Для каждой пары точек (n × n) вычисляется расстояние в d-мерном пространстве

- Это делается один раз в начале

**- Каждая итерация алгоритма: O(n²)**

- Обновление матрицы ответственности R: O(n²)

- Обновление матрицы доступности A: O(n²)

- Обе операции выполняются для всех пар точек

**- Общая сложность: O(n² × t)**

- t итераций × O(n²) на итерацию

- Обычно t = 50-200 итераций

**- Память: O(n²)**

- Хранение матриц S, R, A размера n × n

4) **K-Nearest Neighbors (KNN)**

1. Краткое описание области определения алгоритма

**K-Nearest Neighbors (KNN)** - алгоритм машинного обучения, используемый для задач **классификации** и**регрессии**. Основная идея - определение принадлежности нового объекта к классу на основе меток k ближайших к нему объектов в пространстве признаков.

Начало

2.

Конец

Вывести численное значение

Вывести метку класса

Вычислить среднее значение меток К соседей

Выбрать класс большинства

Подсчитать частоту классов среди соседей К

Тип задачи?

Выбрать К ближайших соседей с минимальными расстояниями

Отсортировать расстояние по возрастанию

Вычислить расстояние от X\_new до все объектов X\_train

Ввод параметра К

Ввод обучающей выборки X\_train, Y\_train

Ввод нового объекта X\_new

классификация

регрессия

3. Пошаговый анализ работы алгоритма

Шаг 1: Вычисление расстояний

- Для нового объекта вычисляются расстояния до каждого объекта обучающей выборки

- Используются метрики: Евклидово или Манхэттенское расстояние

- Результат: список расстояний [d₁, d₂, ..., dₙ]

Шаг 2: Выбор k соседей

- Расстояния сортируются по возрастанию

- Выбираются k объектов с наименьшими расстояниями

- Получаются метки этих k соседей

- Результат: список меток k ближайших соседей

Шаг 3: Принятие решения

А) Классификация:

- Подсчитывается частота каждого класса среди k соседей

- Выбирается класс с наибольшей частотой (мажоритарное голосование)

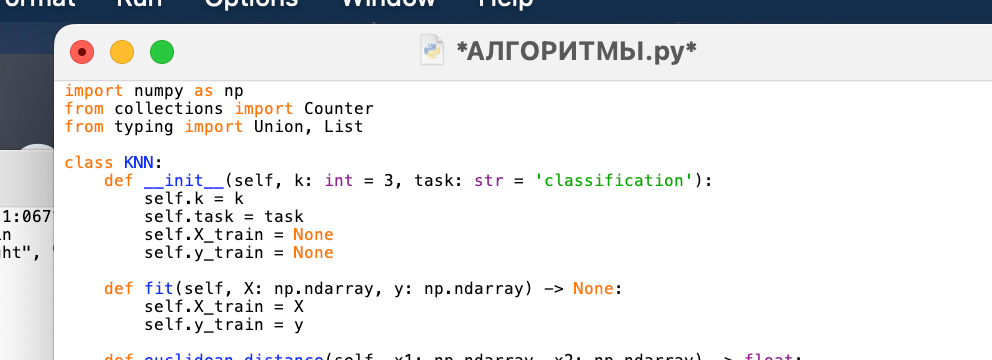
- Результат: метка класса

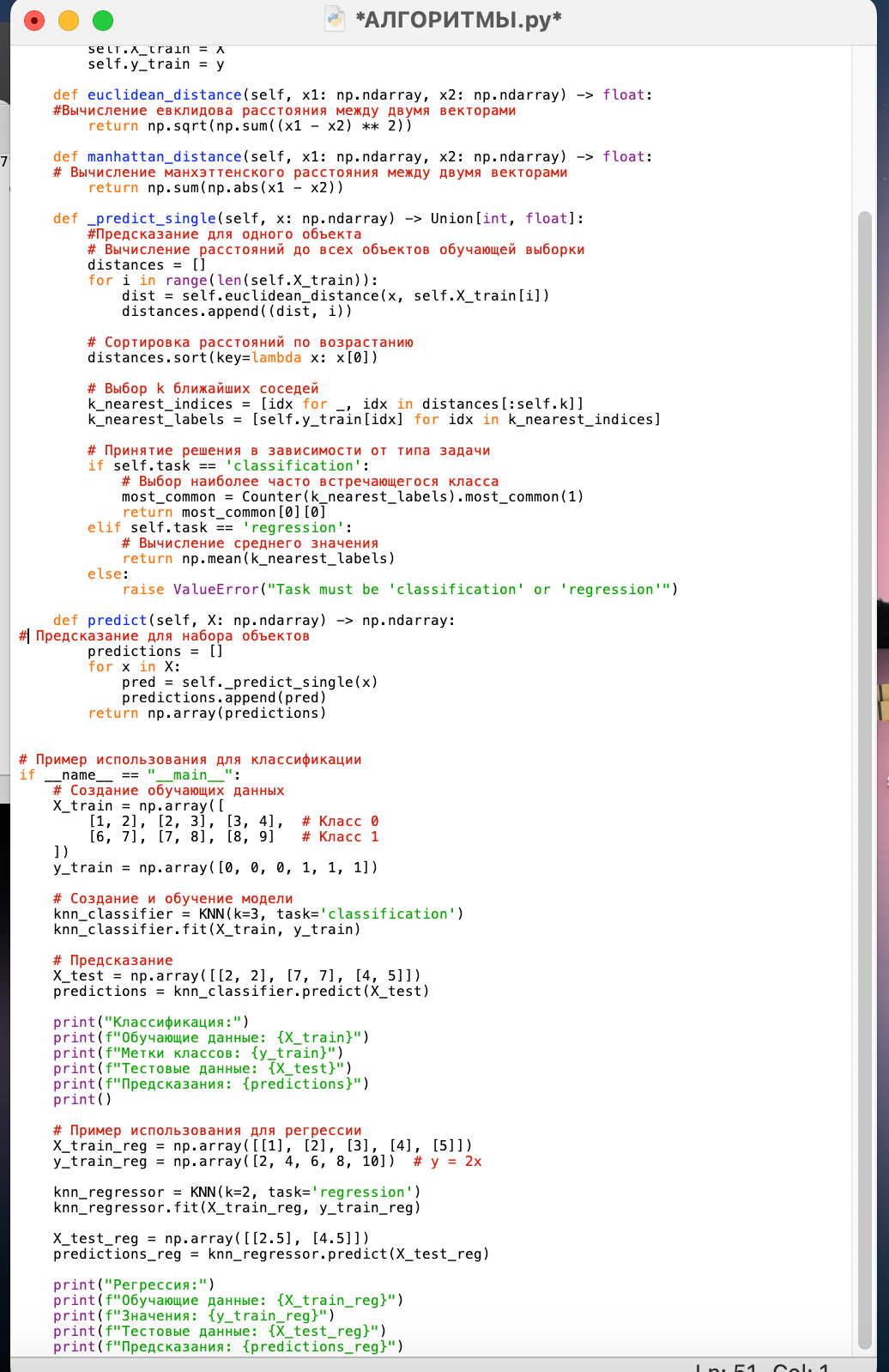
Б) Регрессия:

- Вычисляется среднее арифметическое значений меток k соседей

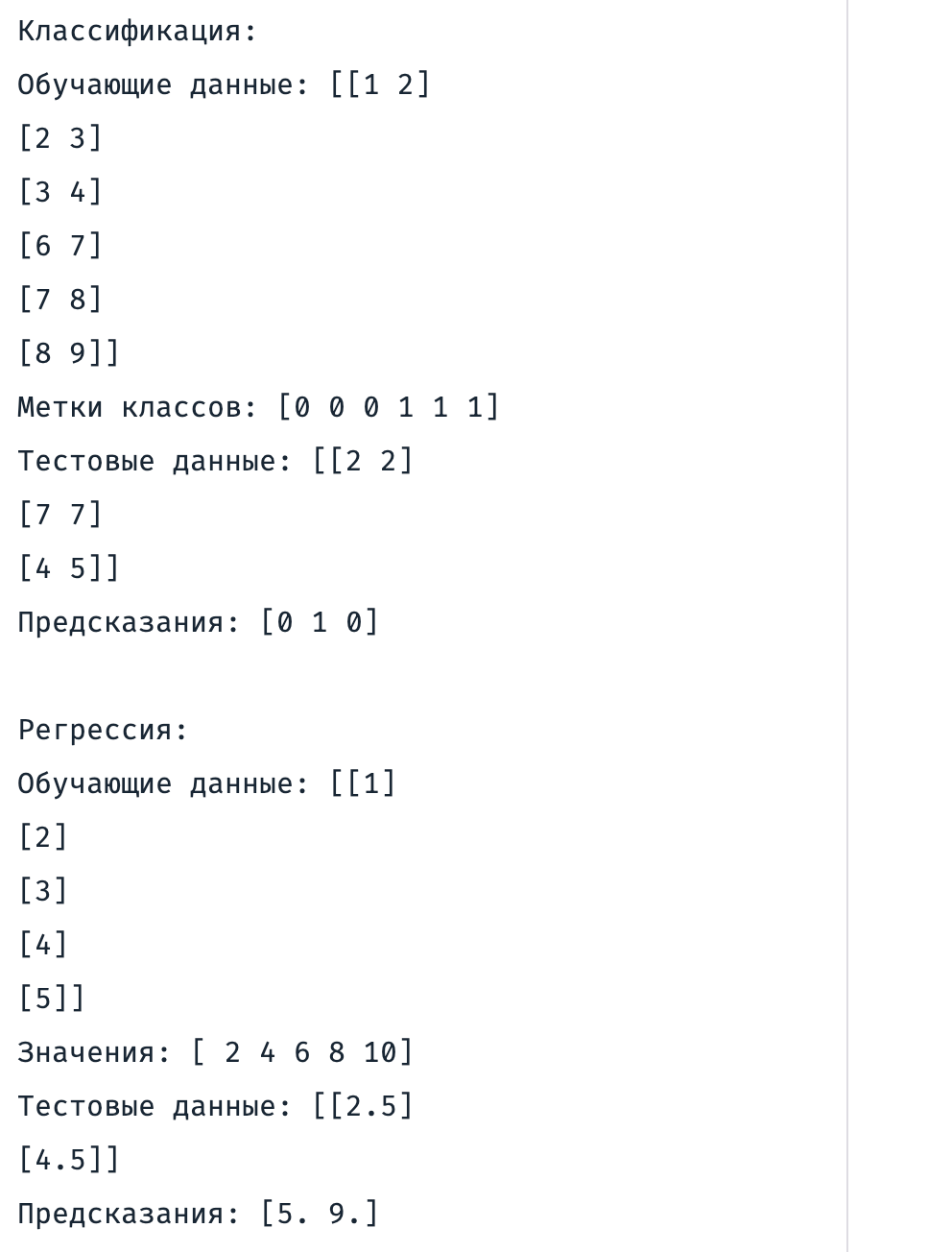
- Результат: численное значение

4. Программный код алгоритма (Python)





Вывод:



5.Оценка временной сложности

**- Фаза обучения (fit):**

**- Сложность:** O(1)

**- Объяснение:** Алгоритм просто сохраняет данные в памяти без выполнения сложных вычислений

**- Фаза предсказания для одного объекта:**

**- Вычисление расстояний:** O(n × d)

- n: количество объектов в обучающей выборке

- d: количество признаков (размерность)

**- Сортировка расстояний**: O(n log n)

- Используется эффективный алгоритм сортировки

**- Выбор k соседей:** O(k)

- Выбор первых k элементов после сортировки

**- Принятие решения:** O(k)

- Подсчет голосов для классификации или вычисление среднего для регрессии

**Итоговая сложность для одного объекта:** O(n × d + n log n + k)  
Упрощенно: **O(n × d + n log n)**

**- Фаза предсказания для m объектов (predict):**

**- Сложность:** O(m × (n × d + n log n))

**- Объяснение:** Для каждого из m тестовых объектов выполняются все вычисления заново