МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Дальневосточный федеральный университет»
(ДВФУ)

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ (ШКОЛА)

Департамент математического и компьютерного моделирования

Название работы

Применение кластерного анализа для группировки пациентов на основе схожих генетических мутаций, влияющих на развитие аденокарциномы лёгких

КУРСОВАЯ РАБОТА
по образовательной программе подготовки бакалавров
по направлению 01.03.02 «Прикладная математика и информатика»
профиль «Математические и компьютерные технологии»

Студент группы № Б9122-01.03.02мкт
_____ Пелагеев Д. И.
(подпись)
«_____» _____ 2025г.

Руководитель к.ф.-м. н. Пак Т.В.
(должность, ученое звание)
_____
_____
(подпись) (ФИО)
«_____»_____2025г.

Работа защищена
с оценкой _____

Регистрационный номер _____
«____»_____2025г.

г. Владивосток
2025

Содержание

Аннотация

Данная курсовая работа посвящена исследованию возможностей кластерного анализа для молекулярной стратификации пациентов с лёгочной аденокарциномой (LUAD) на основе данных проекта TCGA.

Ключивые слова

Кластерный анализ, K-Means, метод Уорда, гауссова смесь, лёгочная аденокарцинома (LUAD), TCGA, PCA, мутационный профиль, copy-number alteration (CNA), Kaplan–Meier, лог-ранк-тест, устойчивость кластеров (bootstrap ARI), неподконтрольное обучение, биоинформатический анализ, RNA-seq, TPM-нормализация, мутационное бремя (TMB), TP53, KRAS, TTN, MAD-фильтр, метод главных компонент, силуэтный коэффициент, индекс Рэнда (ARI), бутстрэп-валидация, молекулярная стратификация, персонализированная онкология, геномика опухолей, copy-number вариации, мультиомные данные, cBioPortal, BigQuery TCGA, визуализация PC1-PC2, скри-плот, cumulative variance.

Введение

Рак лёгких — одно из наиболее распространённых и летальных онкологических заболеваний. По оценкам Всемирной организации здравоохранения, в 2022 году в мире было зарегистрировано около 2,21 млн новых случаев, а число смертей превысило 1,8 млн [1]. Лёгочная аденокарцинома (LUAD) является самым частым гистологическим подтипом — на неё приходится более 40% всех опухолей лёгких[2]. Заболевание характеризуется значительной молекулярной гетерогенностью: у существенной доли пациентов не обнаруживаются «классические» драйверные мутации (EGFR, ALK, KRAS и др.), что осложняет выбор терапии и указывает на сложность генетических механизмов опухоли[3].

Кластерный анализ давно применяется для стратификации онкологических заболеваний и позволяет выделять биологически значимые подтипы. Разделение пациентов по генетическим особенностям опухоли даёт возможность выявлять группы с разными уязвимостями и подбирать персонализированные схемы лечения.

Кластеризация — метод машинного обучения без учителя, целью которого является формирование внутренних однородных и внешне гетерогенных групп объектов. В биоинформатике кластерные алгоритмы используют для группировки генов, белков и образцов тканей по сходному профилю экспрессии или мутаций. В клинической геномике такие подходы позволяют объединять пациентов с похожими молекулярными характеристиками опухоли, что, в свою очередь, способствует уточнению прогноза и оптимизации терапии.

Цель данной работы – рассмотреть методы кластеризации (K-Means, Модель гауссовой смеси, DBSCAN, Метод Уорда), продемонстрировать их применение на примере генетических данных (набор LUAD из проекта TCGA) и провести оценку полученных результатов.

Для достижения поставленной цели был определен следующий план работы:

а) Найти источник для получения датасета, связанного с раком легких, а именно LUAD

б) Провести предварительную обработку данных:
   — Очистка данных от выбросов и пропущенных значений
   — Нормализация численных показателей
   — Кодирование категориальных переменных
   — Отбор значимых признаков с помощью методов снижения размерности
в) Применить разные методы кластеризации:
   — K-Means
   — Метод Уорда
   — Модель гауссовой смеси
г) Получить результаты исследования (выявить кластеры)
д) Провести статистическую валидацию результатов:
   — Оценка силуэтного коэффициента
   — Проверка устойчивости кластеров
   — Сравнение результатов различных алгоритмов кластеризации
   — Валидация на тестовой выборке

1 Теоретические основы кластерного анализа

1.1 Понятие кластеризации

Кластерный анализ (кластеризация) — задача группировки множества объектов так, чтобы элементы внутри одного кластера были максимально похожи, а между кластерами — максимально различны. Поскольку классовые метки отсутствуют, метод относится к обучению *без учителя*. В биоинформатике кластеризацию применяют для группировки генов, белков, а также пациентов по молекулярным профилям, выявляя скрытые подтипы опухолей [4].

1.2 Алгоритмы кластеризации

1.2.1 Метод k-средних

Разбиение на k кластеров осуществляется путём минимизации суммы квадратов расстояний до центров:

$$V = \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2, \tag{1.1}$$

где S_i — множество объектов i-го кластера, μ_i — его центр масс [5]. Алгоритм итеративно чередует этапы «присвоение $\rightarrow$ пересчёт центров» до сходимости.

1.2.2 Метод Уорда

В иерархическом подходе Уорда прирост внутрикластерной дисперсии при объединении кластеров S_i и S_j равен

$$\Delta E = \sum_{x \in S_{ij}} \|x - \mu_{ij}\|^2 - \Big(\sum_{x \in S_i} \|x - \mu_i\|^2 + \sum_{x \in S_j} \|x - \mu_j\|^2 \Big), \tag{1.2}$$

где μ_{ij} — центр объединённого кластера. На каждом шаге сливаются кластеры с минимальным ΔE.

1.2.3 Гауссова смесь (GMM)

Распределение данных моделируется суммой K нормальных компонент:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x \mid \mu_k, \Sigma_k), \qquad \sum_{k=1}^{K} \pi_k = 1. \tag{1.3}$$

Параметры $\{\pi_k, \mu_k, \Sigma_k\}$ оцениваются EM-алгоритмом [6]:

E-шаг

$$\gamma_{ik} = \frac{\pi_k \mathcal{N}(x_i | \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_i | \mu_j, \Sigma_j)}; \tag{1.4}$$

M-шаг

$$\mu_k = \frac{\sum_i \gamma_{ik} x_i}{\sum_i \gamma_{ik}}, \quad \Sigma_k = \frac{\sum_i \gamma_{ik} (x_i - \mu_k)(x_i - \mu_k)^\top}{\sum_i \gamma_{ik}}, \quad \pi_k = \frac{1}{N} \sum_i \gamma_{ik}. \tag{1.5}$$

1.3 Метод главных компонент (PCA)

PCA линейно преобразует данные в пространство, где новая ось v соответствует

$$Cv = \lambda v, \qquad C = \frac{1}{n-1} X_{\text{centered}}^\top X_{\text{centered}}, \tag{1.6}$$

λ — собственное значение ковариационной матрицы, отражающее дисперсию вдоль v. Процент объяснённой дисперсии k компонент:

$$\text{EV}(k) = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{p} \lambda_i}. \tag{1.7}$$

Компоненты выбирают по порогу EV или «каменистой осыпи» (scree-plot).

1.4 Статистическая валидация кластеров

1.4.1 Силуэтный коэффициент

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \tag{1.8}$$

где $a(i)$ — средняя дистанция до собственного кластера, $b(i)$ — до ближайшего чужого. Диапазон $s(i) \in [-1,1]$; усреднённый s служит критерием качества и выбора k [7]. Практическая интерпретация:

— $s > 0.70$ — сильное разделение;

— $0.50 \leq s < 0.70$ — удовлетворительное;

— $0.25 \leq s < 0.50$ — слабое;

— $s < 0.25$ — разделения нет.

1.4.2 Устойчивость кластеров (bootstrap)

Устойчивость кластеров оценивает надежность результатов кластеризации, проверяя, сохраняются ли кластеры при изменении данных или параметров алгоритма. Высокая устойчивость означает, что кластеры отражают реальную структуру данных, а не случайные особенности выборки.

Существуют следующие методы оценки устойчивости кластеров:

— **Бутстрэппинг:** Создаются подвыборки данных, на которых выполняется кластеризация, а затем сравниваются результаты с помощью метрик, таких как коэффициент Жаккара или скорректированный индекс Рэнда.

— **Изменение параметров**: Проверяется, как изменения параметров алгоритма (например, числа кластеров K) влияют на результаты.

— **Повторные запуски**: Для алгоритмов, чувствительных к инициализации (например, K-means), проверяется стабильность при разных начальных условиях.

В нашем случае будет использовать повторная кластеризация на B бутстрэп-подвыборках позволяет оценить стабильность с помощью скорректированного индекса Рэнда (ARI):

$$\text{ARI} = \frac{\text{RI} - \text{E[RI]}}{\max(\text{RI}) - \text{E[RI]}},$$

где RI — индекс Рэнда. Средний ARI$>$ 0.8 свидетельствует о высокой воспроизводимости.

2 Реализация кластеризации

2.1 Поиск датасета

2.1.1 Выбор источника

В соответствии с планом исследования, изложенным во введении, первым этапом работы стал поиск подходящего набора данных. Изначальные попытки были сосредоточены на российских источниках, включая официальные порталы онкологических центров. Однако анализ показал, что доступ к релевантным данным ограничен: большинство наборов либо недоступны для публичного использования, либо содержат фрагментарную информацию.

В связи с этим фокус поиска сместился на международные ресурсы. Среди рассмотренных платформ выделились два источника:

— cBioPortal[8] — открытая платформа с предобработанными наборами данных по геномным профилям опухолей, включая информацию о мутациях, экспрессии генов и клинических исходах;

— GDC Data Portal[9] — база Национального института рака США (NCI), содержащая "сырые"данные многолетних исследований (геномные, эпигеномные, транскриптомные).

Для задачи кластеризации ключевым фактором стала степень предобработки данных. Наборы cBioPortal содержат аннотированные мутации и структурированные клинические метаданные, что минимизирует необходимость дополнительной очистки. В отличие от них, данные GDC требуют трудоемкой первичной обработки: конвертации форматов, фильтрации артефактов секвенирования и интеграции разрозненных типов данных. Дополнительным аргументом стал ограниченный доступ к некоторым наборам GDC, требующий подачи формального запроса.

2.1.2 Выбор датасета

При поиске подходящего датасета выяснилось, что существуют различные наборы данных, содержащие схожую информацию, но предназначенные для разных целей.

На выбор были представлены следующие датасеты:

— MSK (Memorial Sloan Kettering Cancer Center) – Это ведущий NCI онкологический центр и клиническая лаборатория, где разработан и применяется таргетный NGS – панельный тест MSK – IMPACT для одновременного секвенирования опухоль/норма по $\approx 410 - 468$ генам[10].

— TCGA (The Cancer Genome Atlas) – Масштабная совместная инициатива NCI и NHGRI по мультиомному профилированию (экзом, транскриптом, CNV, метилирование, miRNA и пр.) более 11,000 образцов 33 типов опухолей.

— CPTAC (Clinical Proteomic Tumor Analysis Consortium) – Программа NCI, дополняющая данные TCGA глубоким протеомно – геномным анализом. Исследуются сотни опухолей по нескольким типам (протеомика, фосфопротеомика, ацетиломика), чтобы связать геномные аберрации с протеиновыми фенотипами

Из описания видно, что из трёх датасетов нам подходят TCGA и CPTAC, так как они содержат данные о мутациях в генах. В итоге был выбран TCGA[11], поскольку, в отличие от CPTAC, он иммет не полную карту мутаций

2.1.3 Выбор файлов из датасета

Хоть TCGA и имеет нужные нам фалы, но там так же присутствуют те данные, которые мы не будем использовать.

Полный список файлов:

`data_clinical_patient.txt` — Содержит атрибуты пациентов.

`data_clinical_sample.txt` — Содержит атрибуты образцов.

`data_cna_hg38.seg` — Содержит данные о соматических изменениях числа копий (CNA), а именно отношение числа копий из опухолевых образцов минус отношение из соответствующих нормальных образцов.

`data_cna.txt` — Содержит предполагаемые изменения числа копий из GISTIC 2.0, со значениями, указывающими на различные уровни изменений.

`data_mrna_seq_fpkm_zscores_ref_all_samples.txt` — Содержит z-оценки FPKM экспрессии мРНК из данных RNA Seq.

`data_mrna_seq_fpkm.txt` — Содержит значения экспрессии мРНК в FPKM из данных RNA Seq.

`data_mrna_seq_read_counts_zscores_ref_all_samples.txt` — Содержит z-оценки количества прочтений экспрессии мРНК из данных RNA Seq.

`data_mrna_seq_read_counts.txt` — Содержит количества прочтений экспрессии мРНК из данных RNA Seq.

`data_mrna_seq_tpm_zscores_ref_all_samples.txt` — Содержит z-оценки TPM экспрессии мРНК из данных RNA Seq.

`data_mrna_seq_tpm.txt` — Содержит значения экспрессии мРНК в TPM из данных RNA Seq.

`data_mutations.txt` — Содержит данные о мутациях.

`data_timeline_sample_acquisition.txt` — Содержит данные о временной шкале получения образцов.

`data_timeline_status.txt` — Содержит данные о временной шкале статуса.

Из всех эти файлов, нам первоначально пригодится `data_clinical_patient.txt`, так как он содержит метаданные пациента (возраст, пол, стадия, выживаемость). Еще нам нужны: `data_mutations.txt`,`data_clinical_sample.txt`. Мутации нам нужны, так как это основная тема работы, сэмплы помогут на группировать датасеты. Далее для улучшения результатов нужно было добавить файлы: `data_cna.txt`, `data_mrna_seq_tpm.txt`. CNA – поможет лучше кластеризовать, так как с помощью CNA можно увидеть в каком месте мутировал ген. TPM - может показать как две опухоли с одинаковым набором мутаций могут радикально отличаться по активности путей. Express-слой покажет, «работает» ли драйвер-мутация и какие сигнальные цепи активированы.[12]

2.2 Программная среда и используемые библиотеки

Все расчёты выполнялись в среде `Python 3.11.4`. Ключевые библиотеки и их версии приведены в табл. 2.1.

Таблица 2.1 — Используемые библиотеки Python

Библиотека	Версия
pandas	2.2.2
numpy	2.0.2
scikit-learn	1.6.1
lifelines	0.30.0
matplotlib	3.10.1

2.3 Предварительная обработка данных

2.3.1 Очистка данных

После того, как мы стандартными способами загрузили данные в нашу среду разработки. Нам нужно теперь их обработать, стандартные методы обработки включают в себя очистку данных от пропусков, но так как пропусков в датасетах большое множество, то очистка пропусков будет происходить по мере необходимости в коде. Следом нам нужно огрганизовать совместимость данных из разных файлов (см. Приложение А.1). TCGA использует строгий формат идентификаторов, где суффикс "01A"обозначает первичную опухоль [13].

Чтобы еще повысить качество данных. Нужно ичсключить "Silent-мутации из датасета, так как обычно они не влияют на фенотип опухоли [14]. Фильтр "PASS"гарантирует, что варианты прошли контроль качества секвенатора (см. Приложение А.2). [13]

2.3.2 Нормализация TMP

Так как мы используем RNA-seq данные, то для лучших результатов стоит их нормализовать (см. Приложение А.3). Для нормализации

будем использовать логарифмирование.[15]

$$X_{\text{norm}} = \log_2(X + 1) \tag{2.1}$$

где X - TPM-значения, а прибавление 1 исключает неопределённость $\log 0$.

2.3.3 Бинаризация мутаций

Теперь нам нужно преобразовать разреженные категориальные данные в матрицу наблюдение×признак. Метод pivot_table, чтобы создать бинарные векторы (1 = мутация есть в гене) и заполнить пропуски нулями. В данных еще присутствуют редкие мутации, из-за которых матрица становится разреженной. Чтобы это решить применим частотный фильтр в 5% (см. Приложение А.4).[16]

2.3.4 Отбор признаков

— **MAD для экспрессии:** Отбираем 32636 генов, чтобы сделать на анализ более устойчивы к выбросам (см. Приложение А.5). Выбор 32636 генов эмпирически сохраняет 80% информативных генов для рака лёгкого.

— **Фильтрация CNA:** При фильтрации CNA была решено установить порог в 5% для событий CNA (см. Приложение А.6). Если эти собития встречаются редко, то они считаются случайными.

2.3.5 Слияние модальностей

Так как гены могут иметь CNA, экспрессию или мутацию, то нужно добавить к каждым генам их суфиксы. В конце объеденяем все в одну матрицу признаков и транспонируем ее, преобразовав гены в столбцы, а образцы — в строки (см. Приложение А.7).

2.3.6 Нормирование всех признаков

Признаки, которые имеют нулевую дисперсию, мы убираем, так как они не влияют на результат и могут вызывать ошибки. И следующем

шагом стандартизируем с помощью Z-score каждый признак. Это нужно, чтобы привести все признаки к единой шкале (см. Приложение А.8).

$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j} \tag{2.2}$$

где μ_j — среднее, σ_j — стандартное отклонение признака j.

2.4 PCA

Для сокращения размерности данных при сохранении заданного уровня информации (дисперсии) применяем адаптивный PCA-подход. Целевой уровень дисперсии установлен в 80% для баланса между упрощением данных и сохранением биологически значимых паттернов, в этом мы убедимся, когда взглянем на графики объясненной и кумулятивной дисперсии. Пользуемся мы автоматическим подбором компонент PCA с рандомизированный алгоритм SVD и фиксируем seed для воспроизводимости (см. Приложение Б.1).

С помощью бинарного поиска определяется минимальное количество главных компонент, кумулятивная дисперсия которых превышает целевой порог:

$$k = \operatorname{argmin} \left(\sum_{i=1}^{k} \lambda_i \geqslant 0.8 \cdot \sum \lambda \right) + 1 \tag{2.3}$$

где λ_i - собственные значения ковариационной матрицы.

В результате получаем, что 119 компонентов дают $\geqslant$ 80% дисперсии

2.5 Разбиение выборки

2.5.1 Определение оптимального числа главных компонент

Не забываем разбить нашу выборку на тренировочкную и тестовую, чтобы проверить наши результаты кластеризации на независимых данных (см. Приложение Г.1). Под тренировочную выборку отдаем 80%, а под тестовую выборку отдаем оставшиеся 20%.

2.5.2 Визуализация дисперсий PCA

Сохраняем объяснённую и кумулятивную дисперсию для будующей визуализации (см. Приложение Б.2).

Генерируем графики, чтобы визуально оценить правильность и необходимость увеличения компонент в PCA (см. Приложение Б.3).

Рисунок В.1, демонстрирует нам, что первая главная компонента (PC1) объясняет 25% дисперсии, что указывает на наличие сильного паттерна в данных. Далее можно наблюдать, что существует значительное снижение объяснённой дисперсии после PC1 (до <5% к 50-й компоненте). Из-за этого можно сказать, что компоненты после 50-й вносят минимальный вклад и могут быть отброшены как шум. Но прежде чем делать такие вывод, нужно рассмотреть рисунок В.2. На нем можно наблюдать, что для достижения охвата 90% общей дисперсии требуется чуть меньше 250 компонент. При чем начиная с 110 компонент добавляется всего 10% дисперсии, что свидетельствует о вкладе малозначимых факторов (шум, редкие вариации). Дабы сохранить точность и предотвратить переобучение, остановимся на 80% объясненной дисперсии.

2.6 Кластеризация

2.6.1 Функциональные компоненты

Создадим функцию `eval_and_store`, которая будет сохранять метки кластеров и силуэтный коэффициент для последующего сравнения. И Создадим функцию `fit_predict`, которая будет вызывать три алгоритма кластеризации: K-means, Метод Уорда и модель гауссовой смеси (см. Приложение Г.2). При использовании функции обучения (fit_predict), мы выберем диапазон кластеров $k = 2..9$. Этот диапазон выбран эмпирически.

2.6.2 Визуализация найденных кластеров

После такого как мы найдем наилучший метод кластеризации и применим его, то нам нужно увидеть как выглядит наше разделение кластеров. Что бы это увидеть мы примени генерацию графиков (см. Прило-

жение Е.1). Благодаря этому мы получим рисунок(см. Приложение Д.1). На нем видно, что мы имеем два четка разделенных кластера.

2.7 Молекулярная аннотация кластеров

Одной из целей работы было понять, какими именно генами различаются выделенные кластеры. Для этого по каждому гену рассчитана доля образцов внутри кластера, содержащих "не-молчаливую"мутацию (см. Приложение Е.2). Далее для отчёта сохранены таблицы с десятью наиболее частыми генами каждой группы.

Табл. 2.2–2.3 содержат 10 наиболее частых генов в кластерах 0 и 1 соответственно. Заметно, что для обоих кластеров характерны мутации одинаковых генов (*TP53*, *TTN*, *MUC16* и тд), однако кластер 1 демонстрирует систематически более высокую частоту. В общем случае это нам говорит о том, что найденное кластерное разбиение отражает прежде всего общий объём геномных повреждений, а не наличие уникального драйвера.

Таблица 2.2 — Топ-10 генов, кластер 0

Ген (Entrez)	Частота, %
TP53 (7157)	37.1
TTN (7273)	31.4
MUC16 (94025)	30.5
CSMD3 (114788)	27.6
RYR2 (6262)	26.7
KRAS (3845)	25.7
LRP1B (53353)	24.8
ZFHX4 (79776)	20.9
XIRP2 (129446)	20.0
USH2A (7399)	18.6

Таблица 2.3 — Топ-10 генов, кластер 1

Ген (Entrez)	Частота, %
TP53 (7157)	59.1
TTN (7273)	58.0
CSMD3 (114788)	47.0
MUC16 (94025)	46.3
RYR2 (6262)	40.6
USH2A (7399)	38.8
ZFHX4 (79776)	37.4
LRP1B (53353)	37.0
XIRP2 (129446)	29.9
KRAS (3845)	29.5

3 Статистическая валидация

3.1 Сравнение результатов различных алгоритмов кластеризации с помощью силуэтного коэффициента

Для объективного выбора модели выполнен перебор трёх алгоритмов кластеризации — k-means, Ward, Gaussian Mixture — при числе кластеров $k = 2 \dots 9$. На каждом сочетании вычисляли средний силуэт $s \in [-1; 1]$ на обучающей выборке (80 % данных). Максимальное *silhouette* указывает на оптимальную конфигурацию (см. Приложение Г.3).

На тренировочной выборке максимальный силуэт $silhouette = 0.275$ достигнут при **k-means,** $k = 2$. Те же параметры применены к тестовой подвыборке.

Посмотрим как поведет себя модель на тестовой выборке. Применим лучший алгоритма к тестовым данным с последующим объединением меток. Результирующие кластеры сохраняются в CSV для интеграции с другими анализами (см. Приложение Г.4).

Силуэт на тестовых данных равен (0.249). Это близоко к реузльтату на обучающей выборке, что подтверждает отсутствие переобучения.

Теперь создаем график, чтобы наглядно увидеть сравнение методов кластеризации с помощью силуэтных коэффициентов при разном разбиение на кластеры в данной задачи (см. Приложение Е.3).

На рисунке (см. Приложение З.3) видно, что k-means при $k = 2$ превосходит альтернативы, а качество резко падает при $k > 3$.

3.2 Проверка устойчивости кластеров

Надёжность разбиения оценили бутстрэп-подходом ($B = 100$ перезамещённых выборок). Для каждой реплики выполняли повторную кластеризацию лучшим алгоритмом и сравнивали метки с оригинальными (на тех же объектах) с помощью скорректированного индекса Рэнда (ARI) (см. Приложение Ж.1).

Среднее значение ARI составило **0,97** (интерквартильный диапазон 0,96–1,00), что свидетельствует о высокой воспроизводимости кла-

стеров: при случайном перевыборе 80 % образцов метки практически не изменяются (см. Приложение З.1).

Таким образом, двумерное разделение, найденное k-means, является не только наилучшим по силуэт-критерию, но и стабильным при вариациях исходных данных.

3.3 Анализ общей выживаемости

Еще нам стоит проверить, различается ли общая выживаемость (OS) между двумя кластерами, полученными по молекулярным данным (см. Приложение Ж.2). Метод Kaplan–Meier позволяет непараметрически оценить функцию выживаемости $S(t)$, а критерий лог-ранга — статистически сравнить кривые [17].

Для каждого пациента из файла `data_clinical_patient.txt` использованы переменные:

— `OS_MONTHS` — время наблюдения в месяцах;

— `OS_STATUS` — событие (=1, если пациент умер; =0 — если пациент жив);

— метка `cluster`, присвоенная на этапе кластеризации (см. Рисунок Д.1).

В итоге мы получаем рисунок (см. Приложение З.2):

Таблица 3.1 — Сводка Kaplan–Meier-анализа(Лог-ранк $p = 0.61$)

Кластер	n	Медиана OS, мес%
0	245	50.2
1	248	49.0

Эти данные можно интерпретировать так: кривые выживаемости практически совпадают, а медианы OS составляют 50.2 и 49.0 месяца. Лог-ранк $p = 0.61 > 0.05$ свидетельствует об отсутствии статистически значимых различий между кластерами. Следовательно, обнаруженное молекулярное деление отражает различия в генетическом профиле, но не влияет на прогноз общей выживаемости в данной когорте.

Заключение

Настоящее исследование продемонстрировало, что даже при использовании относительно простого вычислительного конвейера (лог-нормализация TPM $\rightarrow$ объединение слоёв мутаций, CNA и экспрессии $\rightarrow$ PCA $\rightarrow$ кластеризация) можно получить воспроизводимое молекулярное разбиение когорты пациентов с лёгочной аденокарциномой TCGA-LUAD.

После многоступенчатой очистки и отбора признаков в пространстве 119 главных компонент алгоритм K-means уверенно выделил два кластера (силуэт 0,25–0,28), причём структура оказалась исключительно устойчивой: при бутстрэп-пересэмпливании скорректированный индекс Рэнда в среднем достигал 0,97. Молекулярная аннотация показала, что различия между группами определяются, главным образом, общим мутационным бременем: во втором кластере чаще встречаются мутации «гигант-генов» *TP53*, *TTN*, *MUC16*, тогда как частота *KRAS* остаётся сопоставимой. При этом различий в общей выживаемости не выявлено (медиана 50,2 мес против 49,0 мес, лог-ранк $p = 0{,}61$), что подчёркивает: обнаруженное разделение отражает скорее фоновую мутационную нагрузку, чем наличие уникального драйвера опухолевого роста.

Результат имеет двойную природу. С одной стороны, мы получаем чётко воспроизводимую стратификацию, которая может оказаться полезной при анализе альтернативных конечных точек — скорости прогрессирования, ответа на иммуно-терапию или комбинаций таргетных препаратов. С другой стороны, отсутствие различий по OS свидетельствует о необходимости более глубокого фенотипического слоя — иммунных сигнатур, метилирования, протеомики CPTAC — и валидации на независимых, в том числе метастатических, выборках (MSK-IMPACT, GENIE).

Тем не менее работа ясно показывает: корректная предобработка данных и строгая статистическая валидация превращают даже базовую связку *PCA + K-Means* в надёжный инструмент молекулярной классификации, который может служить отправной точкой для дальнейших

многофакторных прогностических моделей и персонализированного подбора схем лечения для пациентов с LUAD.

Перспективы дальнейших исследований:

— Оценить прогностическую роль кластерной метки в многофакторной Cox-модели (кластер + стадия + возраст + тип терапии).

— Исследовать возможность более мелкого деления (k = 3 - 4), которое может выявить подгруппу EGFR-/ALK-положительных опухолей с особыми терапевтическими опциями.

— Применить консенсусную кластеризацию, дабы получить более точные результаты или выявить новые кластеры.

Список использованных источников

1. Cancer. URL:https://www.who.int/news-room/fact-sheets/detail/cancer.

2. *Shibiao Wan Yiping Fan, Shengli Li.* Bioinformatics and Machine Learning for Cancer Biology. — 2022. URL:https://mdpi-res.com/bookfiles/book/5918/Bioinformatics_and_Machine_Learning_for_Cancer_Biology.pdf?v=1747098316.

3. *Ferdinandos Skoulidis, John V Heymach.* Co-occurring genomic alterations in non-small-cell lung cancer biology and therapy. — 2019. URL:https://pubmed.ncbi.nlm.nih.gov/31406302/.

4. *Jain, A.K.* Algorithms for Clustering Data / A.K. Jain, R.C. Dubes. — Prentice Hall Inc., Englewood Cliffs, New Jersey, 1988.

5. *J, MacQUEEN.* Some methods for classification and analysis of multivariate observations. — 1967. URL:https://www.cs.cmu.edu/~bhiksha/courses/mlsp.fall2010/class14/macqueen.pdf.

6. *P., Dempster A.* Maximum likelihood from incomplete data via the EM algorithm. — 1977. URL:https://www.ece.iastate.edu/~namrata/EE527_Spring08/Dempster77.pdf.

7. *Rousseeuw, Peter J.* Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. — 1987. URL:https://www.sciencedirect.com/science/article/pii/0377042787901257#aep-abstract-id3.

8. cBioPortal for Cancer Genomics. URL:https://www.cbioportal.org.

9. GDC Data Portal. URL:https://portal.gdc.cancer.gov.

10. *Dr. Berger, Ahmet Zehir.* MSK-IMPACT. — 2017. URL:https://www.mskcc.org/news-releases/msk-impact-first-tumor-profiling-multiplex-panel-authorized-fda-set

11. TCGA-LUAD. URL:https://www.cbioportal.org/study/summary?id=luad_tcga_gdc.

12. *Weinstein, John N.* Comprehensive molecular profiling of lung adenocarcinoma. — 2014. URL:https://www.nature.com/articles/nature13385.

13. *Zhang, Zhenyu.* Uniform genomic data analysis in the NCI Genomic Data Commons. — 2021. URL:https://www.nature.com/articles/s41467-021-21254-9.

14. *Chie Kikutake, Mikita Suyama.* Possible involvement of silent mutations in cancer pathogenesis and evolution. — 2023. URL:https://www.nature.com/articles/s41598-023-34452-w.

15. *Kourkoumpetis, Themistoklis.* Expression of Pattern Recognition Receptor Pathway Genes and Colorectal Neoplasms. — 2018. URL:https://www.researchgate.net/publication/324880091_Sa1670_-_Expression_of_Pattern_Recognition_Receptor_Pathway_Genes_and_Colorectal_Neoplasms.

16. *Mendiratta, Gaurav.* Cancer gene mutation frequencies for the U.S. population. — 2021. URL:https://www.nature.com/articles/s41467-021-26213-y.

17. *Kaplan, E. L.* Nonparametric Estimation from Incomplete Observations. — 2013. URL:https://web.stanford.edu/~lutian/coursepdf/KMpaper.pdf.

Приложение А Обработка данных

Листинг А.1 — Фильтрация TCGA-образцов

```
    def tcga_sample(s):
        return isinstance(s,str) and
            re.fullmatch(r"TCGA-[A-Z0-9]{2}-[A-Z0-9{4}-01A",s)

    common_ids = sorted({c for c in expr.columns
                         if tcga_sample(c)} &
                        {c for c in cna.columns
                         if tcga_sample(c)} &
                        {s for s in samp["SAMPLE_ID"]
                         if tcga_sample(s)})
    expr, cna = expr[common_ids], cna[common_ids]
```

Листинг А.2 — Фильтрация мутаций

```
    mut = mut[mut["Variant_Classification"] != "Silent"]
    mut = mut[mut["FILTER"].isin({"PASS", ".", "", np.nan})]
```

Листинг А.3 — Нормализация экспрессии

```
    expr_log2 = np.log2(expr + 1)
```

Листинг А.4 — Бинаризация матрицы мутаций

```
    mut = mut[mut["Tumor_Sample_Barcode"].isin(common_ids)]
    mut["val"]=1
    mut_bin = mut.pivot_table(index="Entrez_Gene_Id",
        columns="Tumor_Sample_Barcode", values="val",
        aggfunc="max", fill_value=0)
    mut_sel = mut_bin.loc[mut_bin.mean(axis=1)>=0.05]
```

Листинг А.5 — Фильтрация кспрессия

```
    expr_sel = expr_log2.loc[expr_log2.apply(mad,
        axis=1).nlargest(32636)]
```

Листинг А.6 — Фильтрация CNA

```
    cna_sel = cna.loc[(cna!=0).mean(axis=1)>=0.05]
```

Листинг А.7 — Слияние

```
    expr_sel.index = expr_sel.index.astype(str)+"_expr"
    cna_sel.index = cna_sel.index.astype(str)+"_cna"
    mut_sel.index = mut_sel.index.astype(str)+"_mut"

    X = pd.concat([expr_sel, cna_sel, mut_sel]).T
```

Листинг А.8 — Нормирование

```
    X = X.loc[:, X.var()>0]

    X_z = ((X - X.mean())/X.std()).fillna(0)
```

Приложение Б PCA

Листинг Б.1 — Адаптивный PCA

```
pca_full = PCA(svd_solver="randomized",
    random_state=0).fit(X_z)

optimal_n =
    (np.searchsorted(np.cumsum(pca_full.explained_variance_ratio_),
    0.8) + 1)

pca = PCA(n_components=optimal_n, svd_solver="randomized",
    random_state=0)
pcs = pca.fit_transform(X_z)
```

Листинг Б.2 — Экспорт PCA статистики

```
pca_df = pd.DataFrame(pcs, index=X_z.index,
    columns=[f"PC{i+1}" for i in range(optimal_n)])

pca_df_plt = pd.DataFrame({
    "PC": [f"PC{i+1}" for i in range(optimal_n)],
    "explained": pca.explained_variance_ratio_,
    "cumulative": np.cumsum(pca.explained_variance_ratio_),
})
```

Листинг Б.3 — Визуализация PCA дисперсий

```
plt.figure()
plt.plot(pca_df_plt["PC"].str.replace("PC",
    "").astype(int), pca_df_plt["explained"], marker="o")
plt.xlabel("Principal Component")
plt.ylabel("Explained Variance Ratio")
plt.title("Scree Plot")
plt.grid(True)
plt.tight_layout()


plt.figure()
plt.plot(pca_df_plt["PC"].str.replace("PC",
    "").astype(int), pca_df_plt["cumulative"], marker="o")
plt.xlabel("Number of Components")
```

```
13      plt.ylabel("Cumulative Explained Variance")
14      plt.title("Cumulative Variance Explained")
15      plt.grid(True)
16      plt.tight_layout()
17      plt.savefig(OUT_DIR / "pca_cumulative.png")
```

Приложение В Графики PCA

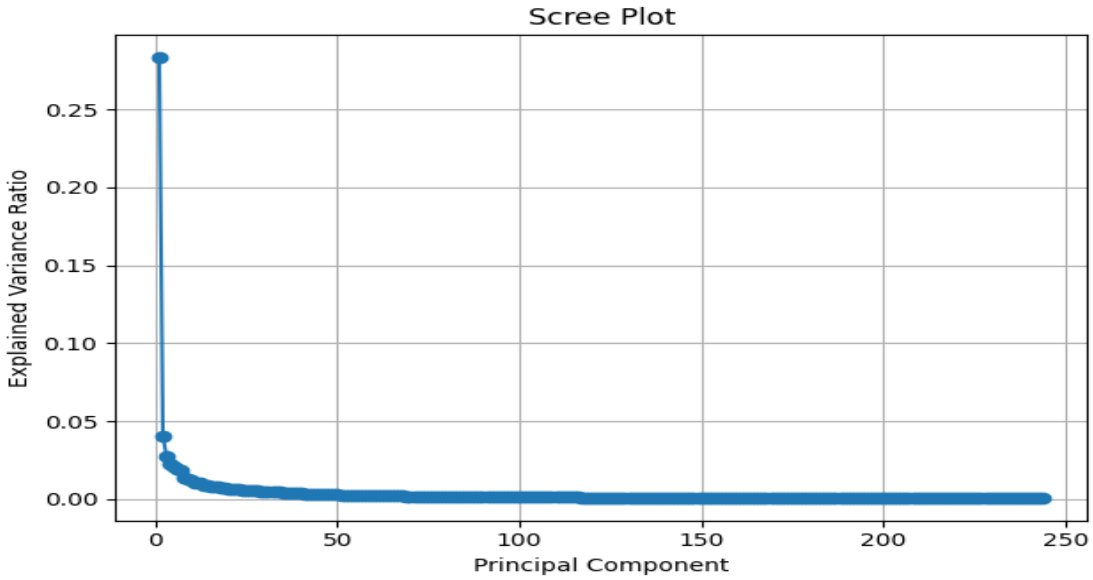

Рисунок В.1 — Объясненная дисперсия

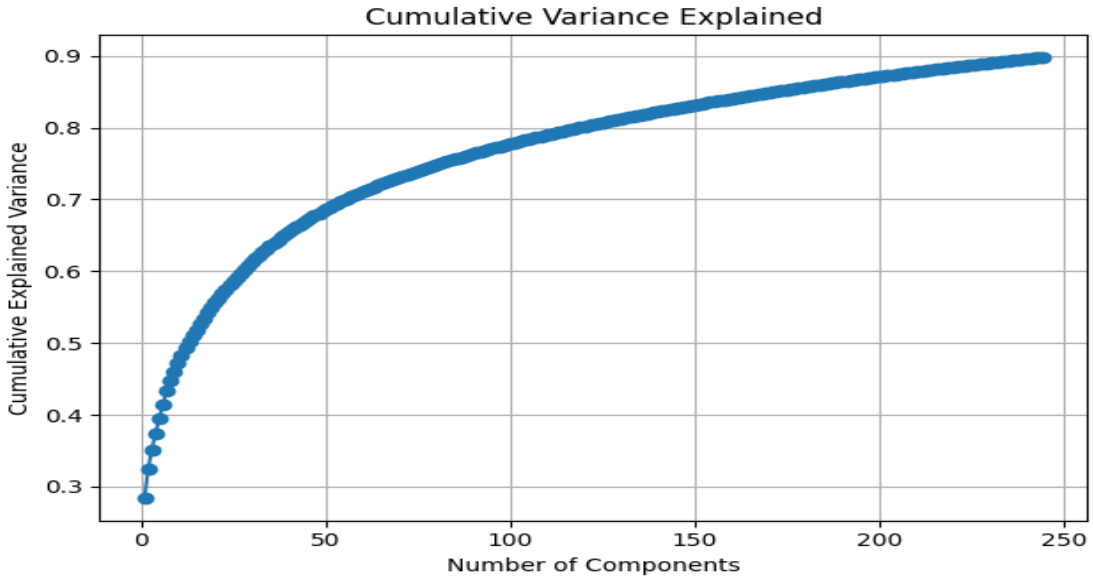

Рисунок В.2 — Кумулятивная дисперсия

Приложение Г Обучение моделей

Листинг Г.1 — Разбиение на тренировочную и тестовую выборки

```
train_idx, test_idx = (train_test_split(pca_df.index,
    test_size=0.20, random_state=0))
X_train, X_test = (pca_df.loc[train_idx],
    pca_df.loc[test_idx])
```

Листинг Г.2 — Функции оценки и обучения

```
def eval_and_store(name, k, labels):
    sil = silhouette_score(X_train, labels)
    results[(name,k)] = (labels, sil)

def fit_predict(method, k, data):
    if method=="kmeans":
        return KMeans(n_clusters=k, n_init='auto',
            random_state=0).fit_predict(data)
    if method=="ward":
        return AgglomerativeClustering(n_clusters=k,
            linkage="ward").fit_predict(data)
    if method=="gmm":
        return GaussianMixture(n_components=k,
            covariance_type="full",
            random_state=0).fit_predict(data)
```

Листинг Г.3 — Грид-поиск и выбор модели

```
results = {}

for method in ["kmeans","ward","gmm"]:
    for k in range(2, 10):
        lbl = fit_predict(method,k,X_train)
        eval_and_store(method,k,lbl)

best_key,(train_labels_arr,
    best_sil)=max(results.items(),
    key=lambda x:x[1][1])
method_best, k_best = best_key
print(f"Best (train) = {best_key}, silhouette={best_sil:.3f}")
```

Листинг Г.4 — Прогнозирование и сохранение

```
if method_best=="kmeans":
    best_model = KMeans(n_clusters=k_best, n_init='auto',
        random_state=0).fit(X_train)
    test_labels = best_model.predict(X_test)
elif method_best=="gmm":
    best_model = GaussianMixture(n_components=k_best,
            covariance_type='full',
                random_state=0).fit(X_train)
    test_labels = best_model.predict(X_test)
elif method_best=="ward":
    best_model = AgglomerativeClustering(n_clusters=k_best,
        linkage='ward')
    test_labels = best_model.fit_predict(X_test)

labels_full = pd.Series(index=pca_df.index,
    dtype=int, name="cluster")
labels_full.loc[train_idx] = train_labels.values
labels_full.loc[test_idx] = test_labels
labels_full.to_csv(OUT_DIR/"luad_clusters.csv", header=True)
```

Приложение Д График кластеризации

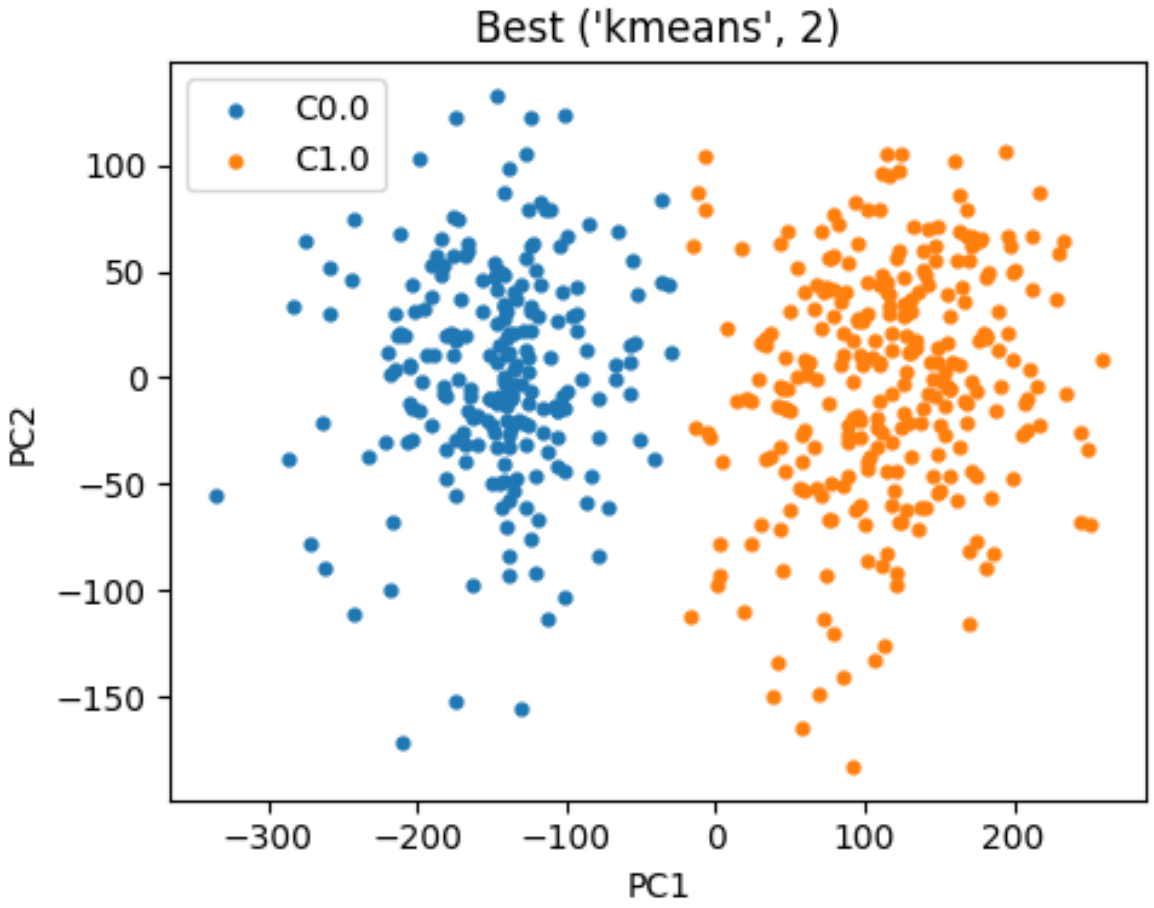

Рисунок Д.1 — Кластеры

Приложение Е Генерация результатов

Листинг Е.1 — Генерация графиков кластеров

```
    plt.figure(figsize=(5,4))
    for cl in sorted(set(labels_full)):
        mask = labels_full == cl
        plt.scatter(pca_df.loc[mask,"PC1"],
            pca_df.loc[mask,"PC2"], s=12, label=f"C{cl}")
    plt.xlabel("PC1")
    plt.ylabel("PC2")
    plt.legend()
    plt.title(f"Best {best_key}")
    plt.savefig(OUT_DIR/"pca_best_clusters.png")
```

Листинг Е.2 — Формирование TOP-10 частот мутаций

```

for c in sorted(labels_full.unique()):
    mask = labels_full == c
    freq = mut_bin.loc[:, mask].mean(axis=1)
    df = (pd.DataFrame({
            "Entrez_Gene_Id": freq.index,
            "HUGO_Symbol": freq.index.map(
                                lambda x: gene_map.get(x, "?")),
            "Frequency": freq.values})
          .sort_values("Frequency", ascending=False))

    df.to_csv(OUT_DIR / f"cluster{c}_gene_frequencies.csv",
              index=False)
    df.head(10).to_csv(OUT_DIR /
              f"cluster{c}_top10.tsv", sep="\t", index=False)
```

Листинг Е.3 — Генерация графика методов кластеризации

```

    plt.figure(figsize=(6,4))
    plt.bar([f"{k[0]}-{k[1]}" for k in results],
        [v[1] for v in results.values()])
    plt.xticks(rotation=90, fontsize=7)
    plt.ylabel("Silhouette (train)"))
    plt.savefig(OUT_DIR/"silhouette_methods.png", dpi=300)
```

```
8       plt.savefig(OUT_DIR/"pca_best_clusters.png")
```

Приложение Ж Код валидаций

Листинг Ж.1 — Bootstrap-устойчивость

```
BOOT_ROUNDS = 100
ari_vals = []

for b in range(BOOT_ROUNDS):
    boot_idx = resample(train_idx, replace=True,
                        random_state=b)
    if method_best == "kmeans":
        lb = KMeans(k_best, n_init='auto',
                    random_state=0).fit_predict(X_train.loc([
                        boot_idx]))
    elif method_best == "gmm":
        lb = GaussianMixture(k_best,
                    covariance_type='full',
                    random_state=0).fit_predict(X_train.loc(
                        [boot_idx]))
    else
        lb = AgglomerativeClustering(
                k_best,
                    linkage='ward').fit_predict(X_train.loc(
                    [boot_idx]))

    ari_vals.append(adjusted_rand_score(
        train_labels.loc[boot_idx], lb))

pd.Series(ari_vals, name="ARI").to_csv(
        OUT_DIR/"cluster_stability_bootstrap.csv",
            index=False)

plt.figure(figsize=(4,3))
plt.boxplot(ari_vals, vert=False, labels=["ARI"])
plt.title(f"Bootstrap ARI (mean {np.mean(ari_vals):.2f})")
plt.tight_layout()
plt.savefig(OUT_DIR/"bootstrap_ari.png", dpi=300)
```

Листинг Ж.2 — Расчёт KM и лог-ранка

```
time  = pat["OS_MONTHS"]
event = pat["OS_STATUS"].str.startswith("1").astype(int)
```

```
3
4       plt.figure(figsize=(6,4)); ax = plt.gca()
5       km = KaplanMeierFitter()
6
7       for c in sorted(pat["cluster"].unique()):
8           mask = pat["cluster"] == c
9           km.fit(time[mask], event_observed=event[mask],
10                  label=f"Cluster {c}")
11          km.plot(ax=ax, ci_show=False, linewidth=2)
12
13      ax.set_xlabel("Overall Survival (months)")
14      ax.set_ylabel("Survival probability")
15      ax.set_title("KaplanБ Meier by cluster")
16      plt.tight_layout()
17      plt.savefig(OUT_DIR/"km_clusters.png", dpi=300)
18
19      c0, c1 = 0, 1
20      res = statistics.logrank_test(
21              time[pat.cluster==c0], time[pat.cluster==c1],
22              event[pat.cluster==c0], event[pat.cluster==c1])
23      p_val = res.p_value
24      print(f"Log-rank p = {p_val:.4f}")
```

Приложение З Картинки валидаций

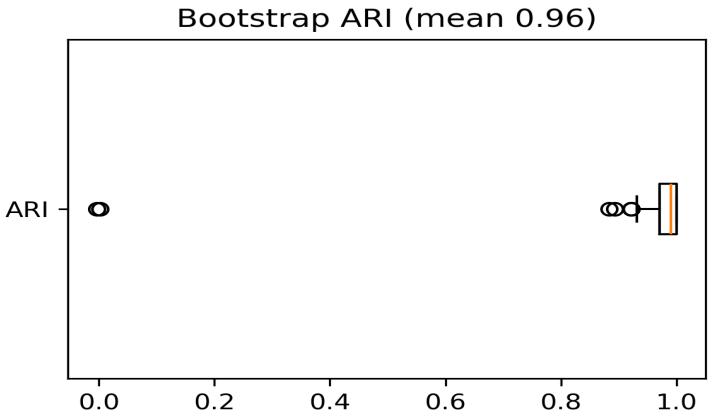

Рисунок З.1 — Устойчивость кластеров по 100 бутстрэп-репликам (ARI)

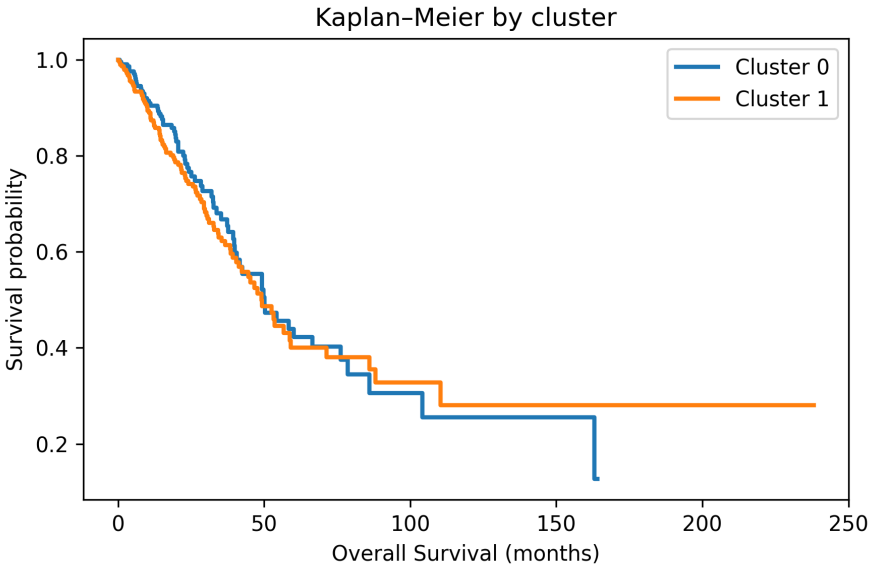

Рисунок З.2 — Кривые Kaplan–Meier для двух кластеров LUAD. Различия статистически незначимы (лог-ранк p = 0.61)

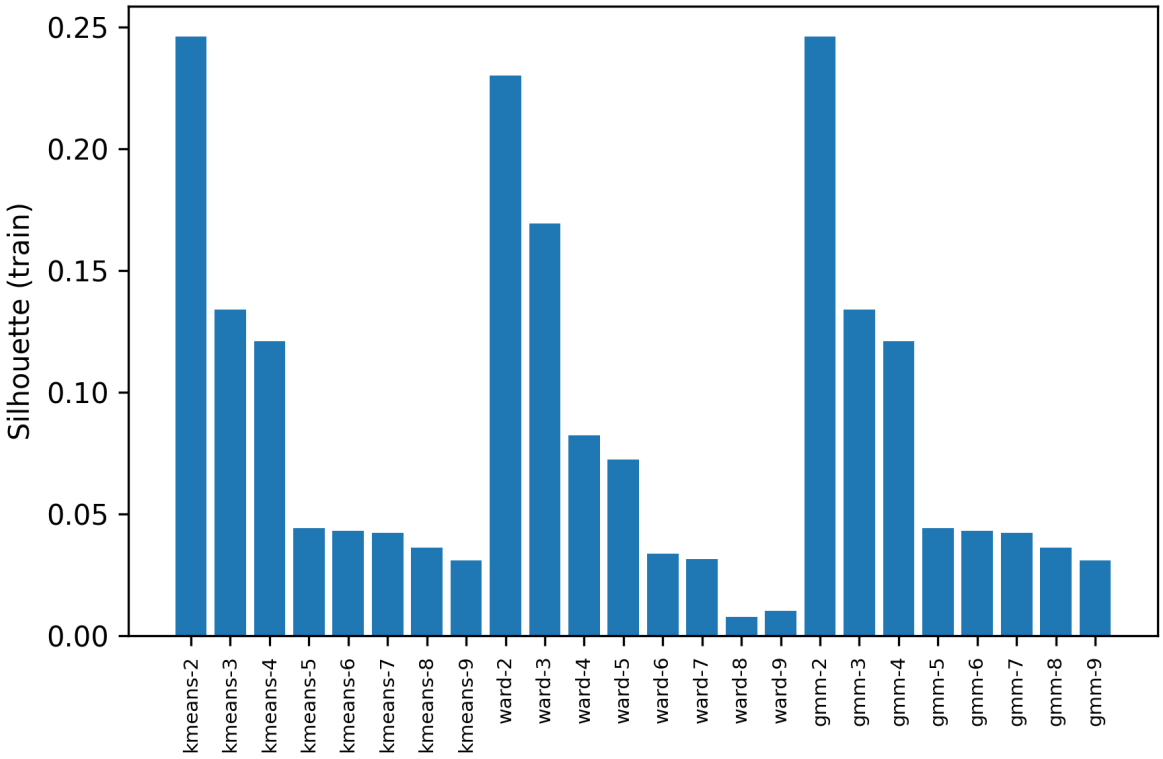

Рисунок 3.3 — Визуализация алгоритмов кластеризации при разных количествах кластеров с помощью метода силуэтов