МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Факультет компьютерных технологий и прикладной математики
Кафедра прикладной математики

Допустить к защите
Заведующий кафедрой
д-р физ.-мат. наук, профессор
_____ М.Х. Уртенов
_____ 2022 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(БАКАЛАВРСКАЯ РАБОТА)

ПРОГРАММНАЯ СИСТЕМА ОЦЕНКИ ПЕРСОНАЛА

Работу выполнил _____ И.А. Сюкол

Направление подготовки 01.03.02 Прикладная математика и информатика

Направленность Математическое и информационное обеспечение экономической деятельности

Научный руководитель
канд. физ.-мат. наук, доц. _____ К.В. Малыхин

Нормоконтролер
преподаватель _____ Е.С. Троценко

Краснодар
2022

РЕФЕРАТ

Выпускная квалификационная работа 57 с., 24 рис., 10 источ.

ОЦЕНКА СОТРУДНИКОВ, ЛИНЕЙНОЕ УПОРЯДОЧЕНИЕ, КЛАСТЕРНЫЙ АНАЛИЗ, МЕТОД К-СРЕДНИХ, ИЕРАРХИЧЕСКОЕ РАЗБИЕНИЕ

В первой главе работы будет кратко рассмотрена гуманитарная часть задачи оценки персонала, а именно, составление списка критериев и способы получения оценок сотрудников по ним. Во второй главе будут освещены простейшие модели ранжирования, применимые к данной задаче. В большинстве своем они основаны на методе Кондорсе. В третьей главе будут рассмотрены математические модели, позволяющие произвести линейное ранжирование сотрудников от лучших к худшим и проведено их сравнение. Четвертая глава будет включать в себя обзор некоторых методов кластерного анализа и поиску наилучшего метода для оценки сотрудников. Пятая глава посвящена программной реализации методов кластерного анализа и сравнению результатов из работы.

Целью данной работы является рассмотрение математических моделей, позволяющих осуществлять оценку персонала и методов ее реализации, начиная от простейших методик, и заканчивая более углубленными, анализ их применимости к поставленной. Признанные наилучшими модели будут реализованы программно, а результаты их работы сравнены.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современной экономике одним из важнейших ресурсов для любого предприятия считается его персонал. Поскольку конечной целью любого предприятия является получение прибыли, становится очевидна необходимость отслеживания характеристик сотрудников, их качеств и способностей с целью определения наиболее и наименее полезных в этом деле людей, а также оптимального распределения должностей и задач между ними. Впрочем, подобная проблема не ограничена одной экономикой, к примеру, набор абитуриентов в ВУЗ тоже сводится к этой задаче – задаче оценивания людей с точки зрения некоторых, заранее определенных вводных.

Для решения задачи оценки чего-либо по определенным критериям существует довольно обширная теоретическая база, включающая в себя методики разной направленности. Задача оценки персонала, допускает применение широкого спектра моделей.

Будучи задачей принятия решений, поставленная задача сводится к 5 шагам, а именно:

1) предварительный анализ и выявление проблем,
2) постановка задачи,
3) получение исходных данных,
4) решение задачи исходя из исходных данных,
5) анализ решения.

Данная работа будет сосредоточена преимущественно на шагах 4 и 5, поскольку ее основным фокусом выступает математическая и алгоритмическая сторона задачи. Шаги 1-3 вкратце рассмотрены в первой главе.

Объектом исследования данной работы будут выступать методы оценки, ранжирования и классификации.

Предметом исследования будет эффективность их применения к задаче оценки персонала.

Целью работы будет нахождение наиболее подходящих моделей, их программная реализация и сравнение результатов. Будут рассмотрены несколько классов моделей, строящихся на различных подходах к задаче.

Актуальность работы обусловлена рациональным стремлением любого предприятия к оптимизации производства. Правильно выполненная оценка

сотрудников позволит установить, какие из сотрудников этому мешают, а какие способствуют, и в какой степени.

1 Основы оценки персонала

1.1 Теоретический базис

Целями процесса оценки будет выступать определение качества выполнения сотрудниками возложенных на них обязательств и установления их сильных и слабых сторон. Такая информация может быть использована начальством для оптимизации штата предприятия (например, повышения наилучших сотрудников и сокращения наихудших, или прием на работу наиболее подходящих для нее кандидатов), а также более оптимального распределения задача между сотрудниками.

Предметом оценки будет являться качество работы сотрудников (скорость выполнения поставленных задач, корректность их выполнения и т.п.), их профессиональные и личностные качества.

Под персоналом в рамках данной работы будем понимать множество сотрудников или соискателей на должность сотрудника некоего предприятия, которые будут подвергаться процессу оценивания. Персонал будет выступать объектом оценки [1].

Субъектом оценки будет выступать начальство предприятия, отдел кадров, или иное подразделение, проводящее сбор данных и их обработку, и применяющее полученные результаты в целях оптимизации работы предприятия. Зачастую проведение оценки и использование ее результатов осуществляется разными лицами или группами лиц, но, поскольку данная работа фокусируется на математическом аспекте задачи, а не на управленческом, такое различие можно не принимать во внимание.

Понятие критериев оценивания подразумевает некоторые характеристики, признанные субъектом оценивания важными для оптимальной работы предприятия. Вследствие этого, набор критериев сильно зависит от задач, выполнение которых ожидается от персонала. Будучи

единожды составлен, набор критериев не предполагает частых и значительных изменений.

Пригодность методики оценивания принято определять по четырем аспектам [2]:

1) теоретический – определяет соответствие полученных результатов результатам, получаемым применением других методик;

2) эмпирический – соответствие полученных показателей реальному поведению сотрудника;

3) внутренний – соответствие элементов методики ее общей цели;

4) внешний – значимость показателей, рассмотренных в методике с точки зрения конкретного предприятия.

Следует подчеркнуть, что это гуманитарные термины, применимые только к методикам составления списка критериев и сбора информации о сотрудниках. Математические же модели будут оценены и сравнены сначала с точки зрения их возможностей и концептуальных недостатков, а затем по результатам их работы на специально составленном наборе данных, достаточно ограниченном, чтобы проверить качество оценивания и характерные особенности результатов эмпирически.

1.2 Сбор входных данных

Для начала необходимо обозначить критерии, по которым будет происходить оценка персонала. Конкретный набор будет зависеть от специфики предприятия, но, тем не менее, можно выделить общие направления, которые желательно взять в рассмотрение:

1) критерии, связанные с выполнением сотрудником должностных обязанностей: профессиональные знания, скорость работы и качество ее выполнения;

2) критерии, определяющие отношение сотрудника к работе: ответственность, дисциплина, лояльность компании, желание развиваться;

3) критерии, отображающие коммуникативные навыки: умение работать в команде, отношения с коллективом, способность понять и принять предоставленные инструкции;

4) критерии, характеризующие личностные качества сотрудника: стрессоустойчивость, честность, умение принимать критику, обучаемость;

5) специальные критерии: могут включать в себя полезные навыки, не имеющие прямого отношения к должности, состояние здоровья, положение в коллективе. Такие критерии сложно нормализовать и свести, например, к десятибалльной шкале, поэтому их можно опустить.

Далее необходимо собрать информацию о персонале, то есть оценить каждого сотрудника по каждому критерию. Логично будет использовать одну систему оценки, к примеру, от 1 до 10, для всех критериев. Есть множество методов получения подобной информации. Рассмотрим наиболее широко используемые на практике:

1) анкетирование – сотрудник заполняет специально составленную анкету, в которую включены вопросы, которые могут указать на его качества, умения и отношение к работе и коллективу. Очевидным минусом является тот факт, что сотрудник будет описывать себя сам и заинтересован скрыть свои недостатки, что может повлечь завышенные результаты;

2) психологическое тестирование – вариант анкетирования с применением психологии. Сосредоточен больше на личностных качествах сотрудника;

3) квалификационное тестирование (аттестация) – сотруднику предлагается решить ряд задач, касающихся его профессиональной деятельности. Анкета должна быть составлена людьми, в эту деятельность вовлеченными, иначе ценность полученных результатов будет невелика. Позволяет объективно оценить профессиональные навыки, но мало говорит об остальных критериях;

4) проверка отзывов – информация о сотруднике получается от его коллег, что исключает искажение информации сотрудником, но взамен допускает такое искажение коллегами;

5) личное собеседование – проводится начальником сотрудника или представителем отдела кадров. Поскольку собеседующий заинтересован в получении правдивых результатов, этот метод наиболее точен, но также наиболее трудозатратен.

Очевидно, что помимо своих преимуществ и недостатков, каждый метод позволяет оценить только определенный набор критериев. В таблице 1 представлена применимость и эффективность данных методик к различным критериям [3].

Таблица 1 – Эффективность методик

Направление критериев	Анкетирование	Псих. тестирование	Аттестация	Проверка отзывов	Собеседование
Профессиональные	+		++		+
Отношение к работе	+	+			++
Коммуникативные		+		++	+
Личностные		++		+	++

В таблице «+» означает, что методика эффективна и применяется на практике, «++» – что методика максимально эффективна. Исходя из этой таблицы, можно видеть, что для получения максимально приближенных к действительности результатов необходимо комбинировать методики.

В практической части работы будем полагать, что данные уже получены.

2 Простейшие методы оценки персонала

2.1 Принцип Кондорсе

Оценивая персонал можно руководствоваться принципом Кондорсе. Он основывается на известном парадоксе Кондорсе, согласно которому, финальная ранжировка кандидатов в демократических выборах может быть нетранзитивна даже если отдельные голоса таковыми являлись, то есть, может привести к ситуации вида «A лучше B, B лучше C, а C лучше A». Кондорсе предложил альтернативную модель выборов, которая применима и к поставленной задаче оценки персонала.

Суть ее такова: для каждого сотрудника заранее известны результаты по ряду критериев (будем обозначать результат по i критерию для k сотрудника как a_i^k). Проводя попарное сравнение сотрудников выясняем однозначного победителя в каждой паре. Им будет тот, у кого число критериев, в которых он превосходит соперника выше числа критериев, где он его не превосходит. Из полученных в итоге соотношений составляется итоговая ранжировка.

Рассмотрим простой пример. Пусть для 3 сотрудников по 3 критериям известны результаты, показанные в таблице 1.

Таблица 2 – Пример 1

	Сотрудник A	Сотрудник B	Сотрудник C
Критерий 1	2	1	25
Критерий 2	4	6	3
Критерий 3	6	5	2

A превосходит B по 2 критериям и проигрывает по 1, отсюда $A \succ B$. Аналогично, $B \succ C$ и $A \succ C$. Следовательно, $A \succ B \succ C$.

Впрочем, хоть эта система и дает более справедливый результат, нежели простой подсчет суммы по всем критериям, она не разрешает парадокс

Кондорсе и также может привести к цикличным результатам. Например показано в таблице 3.

Таблица 3 – Пример 2 (парадокс Кондорсе)

	Сотрудник A	Сотрудник B	Сотрудник C
Критерий 1	10	6	3
Критерий 2	9	8	11
Критерий 3	4	7	5

Тогда получается, что $B \succ C$, $C \succ A$, $A \succ B$, что создает цикл.

Модификацией данного метода является метод Симпсона. В нем так же при попарном сравнении i и j сотрудников вычисляется число u_{ij} – количество критериев, по которым сотрудник i лучше j. Затем для каждого сотрудника вычисляется оценка Симпсона – $\min\limits_{k=1,n} u_{ik}$, где n – количество сотрудников. Финальное ранжирование проводится по убыванию этой оценки.

2.2 Метод Борда

Этот весьма популярный в наши дни метод ранжирования заключается в следующем: n сотрудников упорядочиваются по каждому из m критериев по убыванию оценки. Для каждого критерия сотрудник получает n баллов за первое место, $n-1$ за второе и так далее. Затем его баллы по каждому критерию суммируются. Финальная ранжировка осуществляется по убыванию этих сумм.

Пример показан в таблице 4.

Таблица 4 – Пример 4

	Сотрудник A	Сотрудник B	Сотрудник C
Критерий 1	2	1	25
Критерий 2	4	6	3
Критерий 3	6	5	2

Критерий 1: $C \succ A \succ B$

Критерий 2: $B \succ A \succ \mathrm{C}$

Критерий 3: $A \succ B \succ \mathrm{C}$

Следовательно, A получает 7 баллов, B – 6 баллов, C – 5 баллов. Итого $A \succ B \succ \mathrm{C}$.

Впрочем, этот метод также не исключает проявления парадокса Кондорсе.

2.3 Метод Копленда

Этот метод идейно представляет собой нечто среднее между двумя предыдущими, то есть он сочетает в себе попарное сравнение сотрудников и ранжирование по баллам. Баллы распределяются следующим образом: 1 балл дается сотруднику, если он превосходит оппонента в соответствии с правилами стандартного метода Кондорсе. Проигравший сотрудник получает -1 балл. Если количество доминирующих и доминируемых критериев у них совпадает, то оба получают 0 баллов. По этой разбаловке и проводится результирующее ранжирование.

Приведенные в этой главе методы имеют два ключевых недостатка: парадокс Кондорсе и одномерность результата, то есть они позволяют выстроить сотрудников от лучших к худшим, что, хотя и, безусловно, является полезной информацией, недостаточно полезно само по себе.

3 Методы линейного упорядочения

3.1 Случай без матрицы попарных сравнений

Задача линейного упорядочения довольно проста, если исходные данные представлены в виде матрицы $m \times n$, где n – число сотрудников, а m – число критериев с элементами, представляющими собой оценки по критериям, или в ином виде, подразумевающем оценку всех сотрудников по всем критериям.

Пусть v_i – вес критерия i (если веса не введены, положим их все равными 1), a_{ij} – оценка по этому критерию сотрудника j. Тогда оценку сотрудника k, обозначенную $F(k)$, будем искать по формуле (1):

$$F(k) = \sum_{i=1}^{m} f(a_{ik}, v_i), \quad (1)$$

В формуле (1) функцию $f(a_{ik}, v_i)$ можно выбрать из классов возрастающих функций, представленных формулами (2-4).

– линейная:

$$F(k) = \sum_{i=1}^{m} Cv_i a_{ik},\ C=const \quad (2)$$

– логарифмическая:

$$F(k) = \sum_{i=1}^{m} v_i \log a_{ik} \quad (3)$$

– экспоненциальная:

$$F(k) = \sum_{i=1}^{m} signv_i * e^{v_i a_{ik}} \quad (3)$$

– степенная:

$$F(k) = signq \sum_{i=1}^{m} {a_{ik}}^{q}, q = const \quad (4)$$

Осуществлять выбор рекомендуется отталкиваясь от исходных данных и личного желания оценивающего, поскольку разные классы функций дают разный результат. Так, например, логарифмическая функция позволяет «сглаживать» оценки, то есть, оценки, сильно превышающие остальные, а поэтому вперед выйдут те сотрудники, у кого оценки по критериям более равномерны, вместо тех, кто, к примеру, крайне силен по одному критерию и слаб во всех остальных. Степенная функция, при положительном показателе, может дать противоположный результат в такой ситуации.

Вычислив такую оценку для всех сотрудников, можно упорядочить сотрудников по ней и получить ответ.

Гораздо сложнее обстоит дело, когда исходные данные представляют собой матрицу попарных сравнений самих сотрудников, а критерии в явном виде не присутствуют. Ее также можно задать разными способами (их называют калибровками) [4], представленными формулами (5-9)

– матрица простой структуры:

$$a_{ij} = \begin{cases} 1, F(i) > F(j) \\ \frac{1}{2}, F(i) = F(j) \\ 0, F(i) < F(j) \end{cases} \quad (5)$$

– турнирная калибровка:

$$a_{ij} \geq 0 \ \forall i, j = \overline{1, n} \quad (6)$$
$$a_{ij} + a_{ji} = C, C = const$$

– кососимметрическая калибровка:

$$a_{ij} + a_{ji} = 0 \tag{7}$$

Такая калибровка подразумевает количественное значение превосходства, т.е. сотрудник i превосходит j на a_{ij}.

– степенная калибровка:

$$\begin{gathered} a_{ij} \geq 0 \; \forall i, j = \overline{1, n} \\ a_{ij} a_{ji} = 0 \end{gathered} \tag{8}$$

Здесь подразумевается, что сотрудник i превосходит j в a_{ij} раз.

– вероятностная калибровка:

$$\begin{gathered} 0 \leq a_{ij} \leq 1 \; \forall i, j = \overline{1, n} \\ a_{ij} + a_{ji} = 1 \end{gathered} \tag{9}$$

Здесь элемент понимается, как вероятность превосходства сотрудника i над j.

Далее рассмотрим некоторые методы решения такой задачи в такой постановке.

3.2 Некоторые основные понятия для случая с матрицей попарных сравнений

Модели линейного упорядочивания ставят своей целью получение последовательности сотрудников, упорядоченной по ценности, схоже с тем, что было показано в главе 1. Впрочем, они уходят корнями куда глубже в математику. Такие модели традиционно делятся на 2 группы: в первой группе упорядочивание происходит с помощью парных сравнений и статистическим методам их обработки, тогда как методы второй группы прибегают к

теоретико-графовым и комбинаторным методам для оптимизации всего упорядочивания в целом, дабы максимизировать некоторый введенный показатель качества ранжирования. Впрочем, это разделение довольно нестрогое, поскольку эти классы перекликаются между собой.

Произвольное упорядочивание сотрудников будем обозначать как $I(i_1..i_n)$, где под i_k понимается номер сотрудника. $N_m(I)$ будет означать позицию сотрудника m в упорядочении I. Множество перестановок всех сотрудников обозначим I^* ($|I^*| = n!$). Матрица парных сравнений, представляющая собой входные данные, обозначается как A. Множество упорядочений, оптимальных для матрицы A по выбранной модели, обозначим как $I^*{}_{opt}(A)$. Это множество должно быть непустым.

Введем ряд свойств, описанных формулами (10-15).

1) инвариантность к растяжению:

$$\forall k > 0: I^*{}_{opt}(A) = I^*{}_{opt}(kA) \quad ((10)$$

2) инвариантность к сдвигу:

$$\forall b > 0: I^*{}_{opt}(A) = I^*{}_{opt}(A + B), \quad ((11)$$

где

$B = (b)_{n\times n}$.

3) транспонируемость:

$$\forall I \in I^*: I \in I^*{}_{opt}(A) \Leftrightarrow I^T \in I^*{}_{opt}(A^T), \quad (12)$$

где

I^T есть зеркальное отражение I.

4) положительная реакция:

Пусть есть матрица B, совпадающая с A во всех элементах, кроме b_{ij} и b_{ji}, причем $b_{ij} \geq a_{ij}, b_{ji} \leq a_{ji}$, что равносильно тому, что мы изменили оценку, полученную при попарном сравнении i и j сотрудников в пользу i. Тогда

$$\forall I \in I^*_{opt}(A)\ \exists I` \in I^*_{opt}(B)\colon N_i(I`) \le N_i(I), \qquad (13)$$

то есть улучшение оценки сотрудника i по какому-либо критерию не может привести к ухудшению его позиции в итоговой ранжировке;

5) сохранение доминирования:

Будем говорить, что сотрудник i строго доминирует сотрудника j (обозначим как $x_i \succ\succ x_j$), если во всех их попарных сравнениях он ему не уступает, и хотя бы в одном сравнении превосходит.

Доминирование будем называть полностью сохраняющимся, если

$$x_i \succ\succ x_j \Rightarrow \forall I \in I^*_{opt}(A)\colon N_i(I) < N_j(I) \qquad (14)$$

И сохраняющимся частично, если это соблюдается хотя бы для некоторых $I \in I^*_{opt}(A)$.

Это свойство является довольно важным с точки зрения здравого смысла, поэтому его соблюдение крайне желательно;

6) кусочная оптимальность:

Упорядочение $I = (i_1..i_j..i_m..i_n)$ будем называть кусочно-оптимальным, если любой его фрагмент $I_{jm} = (i_j..i_m)$ также оптимален. Свойство заключается в том, что для любой матрицы попарных сравнений A любое упорядочение из $I^*_{opt}(A)$ кусочно-оптимально. Это свойство довольно строгое, и поэтому, хотя и бывают модели, где оно соблюдается полностью, на практике нередко говорят о его частичном соблюдении;

7) локальная сбалансированность:

Введем на множестве сотрудников X отношение группового доминирования:

$$\forall Y \subset X\ \forall x_i \in X \backslash Y\colon x_i \Rrightarrow Y \Leftrightarrow \sum_{x_j \in Y} (a_{ij} - a_{ji}) \ge 0 \qquad (15)$$

Упорядочение (не обязательно из $I^*_{opt}(A)$) будем называть локально-сбалансированным, если любой его элемент доминирует в смысле (6) любую группу из непосредственно следующих за ним элементов. Свойство заключается в том, что любое оптимальное упорядочение должно быть локально-сбалансировано;

8) возможность получения количественной оценки важности сотрудников. Линейное упорядочение есть суть ряд глобальных оценок важности отдельных элементов, не относительно друг друга, но в некоей общей системе. Удобно и временами весьма полезно иметь возможность получить количественное представление важности, помимо позиции в ранжировке;

9) возможность получения количественной оценки качества аппроксимации данных. На практике очень полезна возможность получить количественно характеристику качества того или иного упорядочения для сравнения.

Модель должна обладать свойствами 1-9 для того, чтобы быть приближенной к реальности и считаться «разумной»[4]. Впрочем, они носят скорее рекомендательный характер и не являются строго обязательными к выполнению. Более полный список можно найти в [5].

3.3 Модели спортивного типа

Название этого класса моделей происходит из-за традиции их применения при подсчете результатов различных спортивных мероприятий. В качестве ранжирующего фактора выступает сумма очков, в простейшем виде это выглядит как показано в формуле (16).

$$\forall i = \overline{1,n}\; s_i = \sum_{i \neq j} a_{ij}, \tag{16}$$

где

s_i – результат i сотрудника.

Финальное упорядочение в простейшей (турнирной) модели осуществляется по убыванию s_i.

Довольно очевидно, что простейший вид такой модели едва ли способен выдавать справедливые результаты в общем случае. Более того, в таком виде никак не предусматривается одинаковое количество очков у двух и более людей. В таком случае их нужно либо внести в результирующее упорядочение в произвольном порядке, либо использовать какие-то дополнительные критерии определения победителя. То есть мы видим, что популярность этой модели обусловлена лишь простотой ее применения.

Нетрудно убедиться, что турнирная модель сохраняет доминирование и обладает свойством положительной реакции. Соблюдение свойств 6, 7, 8 и 9 не происходит. Транспонируемость и инвариантность к сдвигу и растяжению можно гарантировать лишь тогда, когда оптимальное упорядочение не выделено, то есть объекты (сотрудники) распределены без участия вторичных показателей.

Турнирная модель осуществляет ранжирование в один проход. Есть множество ступенчатых альтернатив этой модели. Рассмотрим систему последовательного вычленения лидера[5]. Методы такого типа сводятся к выделению лучших членов исходной выборки, затем лучших из них и т.д.

На первое место результирующего упорядочения ставится сотрудник (сотрудники) с наибольшей суммой баллов. Соответствующая строка и столбец матрицы A вычеркиваются, на второе место ставится сотрудник с максимальной суммой по новой матрице и т.д.

Такой подход несколько отличается от турнирной модели по свойствам, а именно, частично соблюдается кусочная оптимальность и линейная сбалансированность, но при этом сохранение доминирования гарантируется только для невыделенных упорядочений.

3.4 Стохастическая модель Ушакова

Такая модель строится на преобразовании матрицы попарных сравнений A в вероятностную матрицу P, где p_{ij} означает вероятность превосходства x_j над x_i[6]. Если A задана в вероятностной калибровке, то $P = A^T$, если в степенной, то элементы матрицы P вычисляются по формуле (17).

$$p_{ij} = \frac{a_{ji}}{1 - a_{ji}} = 1 - p_{ji} \tag{17}$$

Затем матрица P преобразуется к стохастической матрице $\bar{P}$, элементы которой вычисляются по формулам (18), (19)

$$\overline{p_{ij}} = \frac{p_{ij}}{n-1} = 1 - p_{ji} \text{ при } i \neq j \tag{18}$$

$$\overline{p_{ii}} = 1 - \sum_{i \neq j} p_{ij}\,, i,j = \overline{1,n} \tag{19}$$

Матрица $\bar{P}$ понимается как матрица переходных вероятностей некоторой марковской цепи. Показатели, по которым строится результирующее упорядочение, вычисляются по формуле (20).

$$p_i = \frac{\Delta_{ii}}{\sum_{j=1}^{n} \Delta_{jj}},\ i = \overline{1,n,} \tag{20}$$

где

$\Delta_{ij} -$ минор матрицы $(E - \bar{P})$.

Эта схема частично соблюдает кусочную оптимальность. Свойства инвариантности по сдвигу и растяжению не соблюдаются, поскольку модель подразумевает изменение матрицы попарных сравнений. Свойство транспонируемости не гарантируется, хотя строгое доказательство его

несоблюдения получено не было. Свойство линейной сбалансированности также не соблюдается.

3.5 Сравнение моделей

Сопоставим рассмотренные модели. Данные по моделям представлены в таблицах 5 и 6.

Таблица 5 – Свойства моделей

	ИР	ИС	Т	ПР	СД	КО	ЛС	ОВ	ОК
Турнирная	+	+	+	+	+	-	-	-	-
ПВЛ	+	+	+	+	+	±	±	-	-
СМУ	-	-	-	+	+	±	-	+	-

Таблица 6 – Допустимые калибровки входной матрицы

	Калибровки
Турнирная	Простая структура, турнирная, кососимметрическая
ПВЛ	
СМУ	Степенная, вероятностная

Преобразуем полученные данные в матрицу простой структуры. Стохастическую модель Ушакова придется исключить из рассмотрения, поскольку она предназначена для других калибровок. Оценки моделей будем осуществлять с помощью линейной функции с коэффициентом 1, все веса также возьмем за 1. Полученная матрица представлена в таблице 7.

Таблица 7 – Матрица попарных сравнений моделей

	Т	ПВЛ
Т	0.5	0
ПВЛ	1	0.5

Довольно очевидно, что модель ПВЛ лидирует в данной постановке задачи, но, в целях наглядности, решим эту задачу обеими участвующими в ней моделями с помощью программных средств.

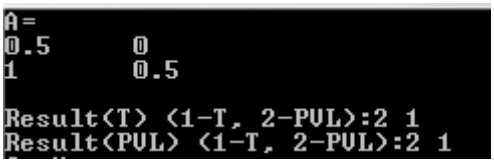

Рисунок 1 – Результат работы программы

Как мы видим на рисунке 1, программа работает корректно. Рассмотренный пример довольно тривиален, но программа, с минимальными изменениями, пригодная и для матриц большей размерности.

Модели представленные в этой главе также не дают никакой дополнительной информации помимо ранга сотрудника. Поэтому, они не будут использованы в дальнейшем. Желательно сгруппировать сотрудников по тем или иным критериям, что позволит начальству, проводящему оценку, оптимизировать их работу не только через поощрение наилучших, но и, например, через улучшенное распределение задач между сотрудниками опираясь на их сильные стороны.

4 Кластер-методики

4.1 Основы

Имея данные в виде оценок всех сотрудников по критериям, можно осуществить их оценку посредством кластер-методик. Суть их заключается в том, чтобы разбить сотрудников на некоторое количество кластеров, внутри каждого из которых будут схожие в некотором смысле сотрудники (каждый сотрудник входит ровно в один кластер) [7]. Каждого сотрудника будем представлять в виде вектора, составленного из его оценок: $A_i = (a_{i1}..a_{im})$.

Понятие кластера, то есть, скопления точек в пространстве, хоть и кажется эмпирически очевидным, на самом деле невозможно однозначно формализовать.

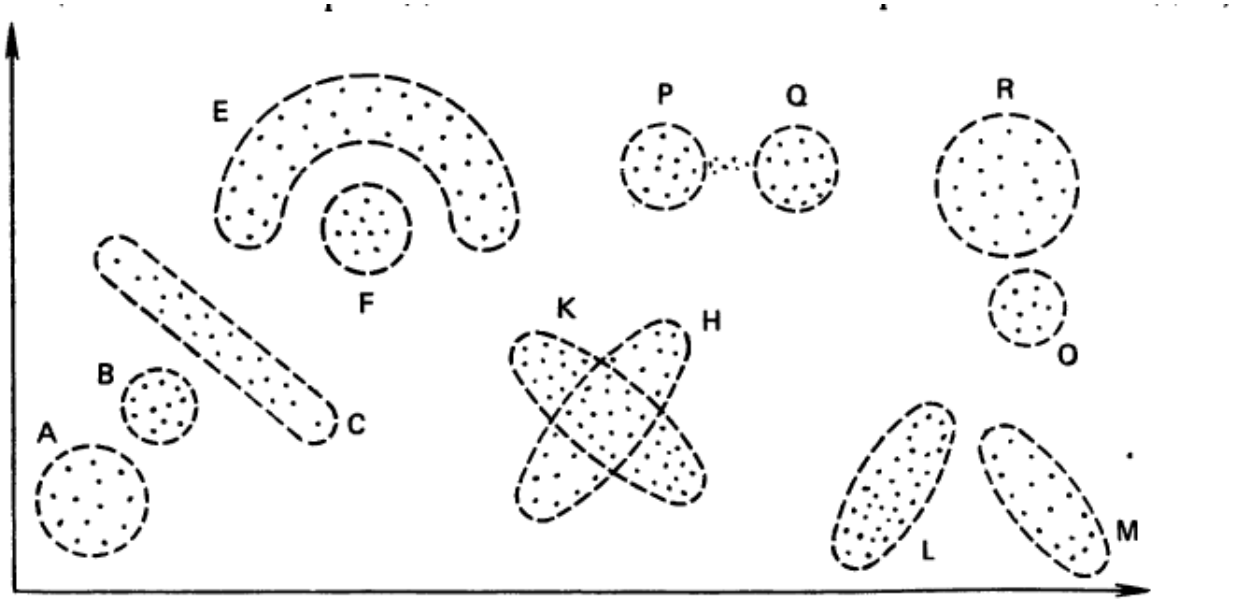

Рисунок 2 – Различные кластеры

На рисунке 2 представлены различные кластеры в двухмерном пространстве в том виде, в каком их легко распознать «на глаз». Однако, например, расстояния между некоторыми точками в классе C меньше, чем расстояния между точками этого кластера и кластера B, кластеры P и Q соединены перемычкой из точек, а средние координаты в кластерах F, E, K и

Н одинаковы. Отсюда становится очевидным, что крайне сложно построить математическую модель, которая сможет учесть все эти детали и выдать разбиение, показанное на рисунке. После десятилетий попыток некоторые ученые начали понимать задачу такого разбиения как задачу распознавания образов, то есть машинного обучения.

В данной работе будут рассмотрены методы разбиения, подразумевающие задание одного четкого определения кластера. Такие алгоритмы называются алгоритмами прямой классификации [8].

4.2 Способы определения кластера

Для методов прямой классификации существует множество способов определения кластера [7]. Рассмотрим некоторые из них:

1) все элементы кластера находятся ближе к другим элементам этого кластера, чем к любым элементам вне него. Такой подход называют кластером сгущения или кластером типа ядра. На рисунке 2 к такому типу относятся кластеры A и B;

2) внутри кластера задается некоторая точка, называемая центром кластера (например, центр тяжести). Элементы кластера должны находиться в пределах заданного расстояния от этой точки. Пример – кластеры B и P на рисунке;

3) каждый элемент кластера должен находиться в пределах заданного расстояния от хотя бы одного другого его элемента. Все кластеры типа 1 также подходят под это определение. Такой кластер называется кластером слабого сгущения или кластером типа ленты. На рисунке этот класс (за вычетом класса 1) представлен кластерами C, E, K, H, L, M;

4) так называемый кластер сгущения в среднем. Схож с типом 1, но подразумевает, что среднее расстояние между элементами кластера меньше, чем среднее расстояния с остальными;

5) комбинация типов 4 и 2, то есть, среднее расстояние элементов кластера до его центра меньше, чем среднее их расстояние до центров других кластеров;

6) модификация типа 4. Среднее расстояние между элементами кластера должно быть в $b > 1$ раз меньше, чем среднее расстояние между любым объектом вне кластера и всеми его элементами. Такой кластер принято называть сильным;

7) кластер типа изолированного облака. Не выдвигает требований к плотности самого кластера и расстояниям между ними. Задается по формуле (22).

$$\exists \tau > 0 : \forall \, \mathrm{c}_i \in C_k \, \forall c_j \, \overline{\in} \, C_k \; d_{ij} > \tau \tag{22}$$

где

c_i, c_j – элементы исходного множества;

C_k – кластер;

d_{ij} – расстояние между c_i и c_j.

Все кластеры на рисунке 2 удовлетворяют данному определению.

Очевидно, что выбор определения может сильно повлиять на полученное разбиение.

Далее будут рассмотрены две популярные методики кластерного анализа: иерархический алгоритм и метод К-средних.

4.3 Иерархический алгоритм

Общая последовательность действий простейшего иерархического алгоритма выглядит следующим образом:

1) каждый объект полагаем отдельным кластером. Вычисляется матрица расстояний между кластерами;

2) находятся два ближайших кластера;

3) два ближайших кластера объединяются в один новый. Матрица расстояний пересчитывается, ее размерность уменьшается на 1.

4) шаг 3 повторяется, пока все объекты не будут включены в один общий кластер. Такое исполнение называется методом одиночной связи или методом ближайших соседей;

Существует модификация данного алгоритма, называемая методом Уорда. Ключевое отличие кроется в третьем шаге алгоритма. Согласно методу Уорда, после начального объединения двух ближайших кластеров, для них вычисляется сумма квадратов отклонений по формуле (23), представленной ниже:

$$V_I = \sum_i \sum_j \left(x_{ij} - \overline{x_{JI}}\right)^2 \tag{23}$$

где

I – номер кластера,

i – номер объекта,

j – номер критерия оценки.

Кластеры объединяются таким образом, чтобы прирост V_I был минимален, что достигается вычислением расстояний по формуле 24:

$$d(u,v) = \sqrt{\frac{|v|+|s|}{T} d(v,s)^2 + \frac{|v|+|t|}{T} d(v,t)^2 - \frac{|v|}{T} d(s,t)^2} \tag{24}$$

где

u,v – рассматриваемые кластеры,

s,t – кластеры, составляющие u,

T – мощность кластера u.

На каждой итерации V_I пересчитывается для каждого кластера.

Результатом работы данного метода является дендрограмма, пример которой представлен ниже на рисунке 3.

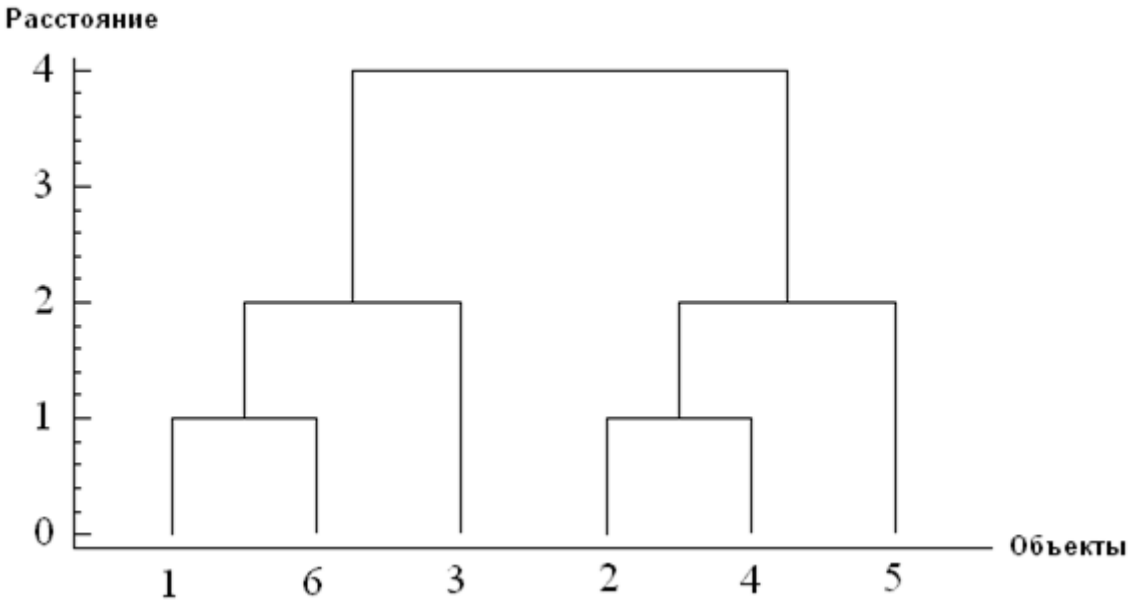

Рисунок 3 – Пример дендрограммы

На рисунке 3 видно, что по оси X расположены объекты, а по оси Y – расстояния между ними. Традиционно расстояния вычисляются в евклидовой метрике. Тогда чтобы выделить из этой структуры кластеры, достаточно найти максимально длинные вертикальные линии, не пересекающиеся горизонтальными. Такие линии будут соединять максимально удаленные друг от друга группы объектов, которые вполне можно счесть итоговыми кластерами. На данном примере отчетливо видно, что оптимальное разбиение содержит 2 кластера из элементов 1, 6, 3 и элементов 2, 4, 5 соответственно.

Данные алгоритмы позволяют не только установить полезность отдельно взятых сотрудников, но и сгруппировать их по схожести оценок. Такая информация окажется намного полезнее для работодателя, чем простое упорядочение сотрудников от лучших к худшим. Поэтому, данный класс методов будет представлен в программной реализации.

4.4 Метод К-средних

Этот метод отличается от иерархического алгоритма в первую очередь тем, что он не позволяет построить полноценную иерархическую структуру, но разбивает имеющееся множество объектов на k кластеров $\{S_1 \dots S_k\}$ так,

чтобы суммарное квадратичное отклонение точек кластеров от центров этих кластеров было минимальным, опираясь в вычислениях на заранее известное число кластеров – k.

Алгоритм имеет следующий вид:

1) после получения k, вычислить начальные центры кластеров $(\mu_1 \dots \mu_k)$. Для первого шага их можно взять случайными;

2) каждый объект определяем в тот кластер, центр которого ближе всех к нему. В качестве метрики принято использовать Евклидово расстояние;

3) центры кластеров пересчитываются по формуле (25):

$$\mu_i = \frac{1}{|S_i|} \sum_{x_j \in S_i}^{j} x_j \tag{25}$$

4) шаги 2-3 повторяются, пока на очередной итерации после пересчета центры не останутся неизменными.

Следует отметить, что если брать случайные центроиды при инициализации, то можно попасть в так называемую ловушку инициализации, и метод сойдется к локальному оптимуму вместо глобально оптимального решения.

Оптимальное количество кластеров можно и нужно аппроксимировать перед получением финального результата. Зачастую это делается по так называемому правилу локтя, где алгоритм запускается последовательно, каждый раз увеличивая число кластеров на 1, пока уменьшение суммарного квадратичного отклонения точек кластеров от их центров не станет достаточно малым. График зависимости отклонения от числа кластеров напоминает согнутую в локте руку, что и дало методу название, и оптимальное количество кластеров находится в районе сгиба.

Впрочем, этот метод является самым простым, но не самым точным [9], поскольку сгиб «локтя», как правило, довольно плавный и может вмещать в

себя несколько потенциально оптимальных вариантов количества кластеров. Более точным будет коэффициент силуэта.

Метод силуэта схож с методом локтя в том, что производится последовательное разбиение с увеличением числа кластеров, но в качестве метрики используется не суммарное квадратичное отклонение, а коэффициент силуэта, вычисляемое по формуле (26):

$$s_k = \frac{b_k - a_k}{max(a_k, b_k)} \tag{26}$$

где

a_k – среднее расстояние k-того элемента до других элементов его кластера,

b_k – среднее расстояние до элементов ближайшего кластера, в который он не входит.

График зависимости будет иметь гораздо более выраженный пик, как видно на рисунке 4.

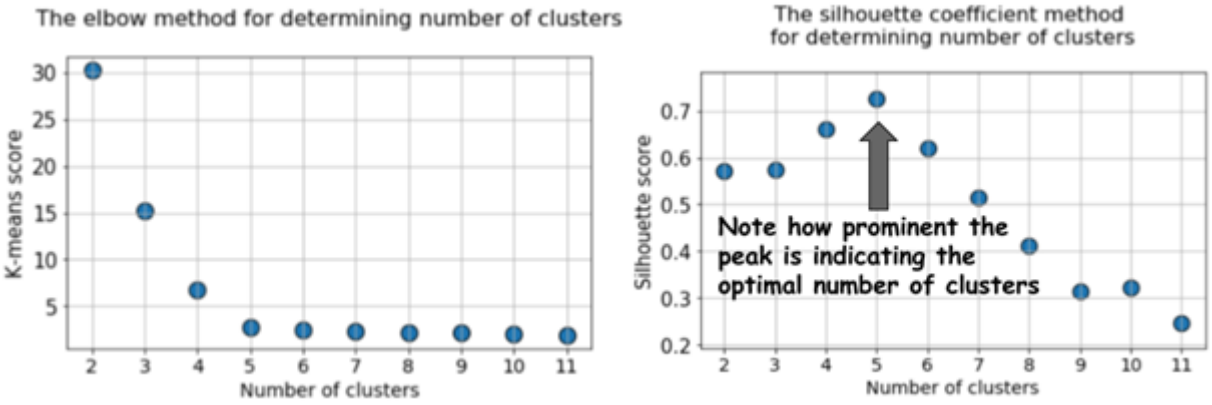

Рисунок 4 – Сравнение метода локтя и метода силуэта

Следует отметить, что приведенный выше алгоритм подразумевает рассчет расстояний от каждого центроида до каждой точки на каждой итерации, что не является оптимальным. Для увеличения быстродействия можно использовать алгоритм Элкана [10], использующий неравенство треугольника. Его суть заключается в том, что если известно расстояние от точки x_i до центроида μ_k и расстояние между центроидами μ_k и μ_l, и

$2d(x, \mu_k) \leq d(\mu_k, \mu_l)$, то можно заключить, что $d(x, \mu_k) \leq d(x, \mu_l)$ без необходимости вычислять $d(x, \mu_l)$.

Данный метод, как и иерархические алгоритмы, позволяет получить групповое упорядочение сотрудников, а потому также будет включен в программную реализацию и сравнен с предыдущим подходом.

5 Программная реализация

5.1 Постановка задачи. Выбор исходных данных

Конечной задачей данной выпускной квалификационной работы является разработка программы, реализующей методы оценки сотрудников, признанные наиболее подходящими для данной цели. Такими методами были сочтены иерархическая классификация и метод К-средних. Программа должна хранить данные о сотрудниках, а именно, ФИО, должность и оценки по обозначенным критериям, а также выполнять классификацию сотрудников по вышеупомянутым алгоритмам, предоставляя визуализацию результата.

Для простоты визуализации, было решено ограничиться тремя критериями. Они включают в себя самые важные направления из рассмотренных в главе 1, а именно, качество выполнения сотрудником его обязанностей, отношение к работе и личностные качества. На практике, первичный этап сбора информации может содержать и большее количество критериев, но каждый из них может быть отнесен к одной из этих групп. В таком случае значения трех итоговых критериев можно вычислить. Ограничим выборку 20 сотрудниками. Фрагмент выборки в том виде, в каком она представляется в программе, показан на рисунке 5.

	ФИО	Должность	Продуктивность	Отношение	Лич. качетсва
1	Иванов И.И.	Программист	9	9	8
2	Петров П.П.	Программист	7	9	10
3	Сидоров С.С.	Мл. менеджер	7	7	8
4	Козлов К.К	Ст. программист	9	9	7
5	Еремин Е.Е.	junior-разработчик	5	6	7

Рисунок 5 – Фрагмент исходных данных

5.2 Выбор средств разработки

Для реализации поставленной задачи был выбран высокоуровневый язык программирования Python. Главным его достоинством как в контексте данной работы, так и вне него, является широкий выбор библиотек, позволяющих значительно сократить количество усилий, требуемых для создания приложения.

Для создания графического интерфейса программы, а именно, окон, кнопок, таблиц и подписей, использована библиотека PyQT5. Она несложна в освоении и широко использует объектно-ориентированный подход для создания и гибкой настройки пользовательского интерфейса. Также она включает в себя инструменты для работы с базами данных.

Для визуализации результатов кластеризации использован модуль pyplot из пакета matplotlib. Он схож с системой построения графиков в MATLAB, и так же позволяет создавать интерактивные графики. Кроме того, модуль включает инструменты для обработки событий, таких, как обновление данных или нажатие кнопки мыши.

Для упрощения реализации, собственно, методов кластерного анализа был применен модуль cluster из пакета scipy. Этот модуль содержит классы, значительно упрощающие выполнение заявленных методов и визуализацию результатов.

5.3 Работа программы

Главное окно программы содержит список сотрудников и кнопки, запускающие кластерный анализ методами иерархической классификации и К-средних. Кроме того, для метода К-средних предусмотрен ввод пользователем числа кластеров и два метода его аппроксимации, описанные в главе 4.

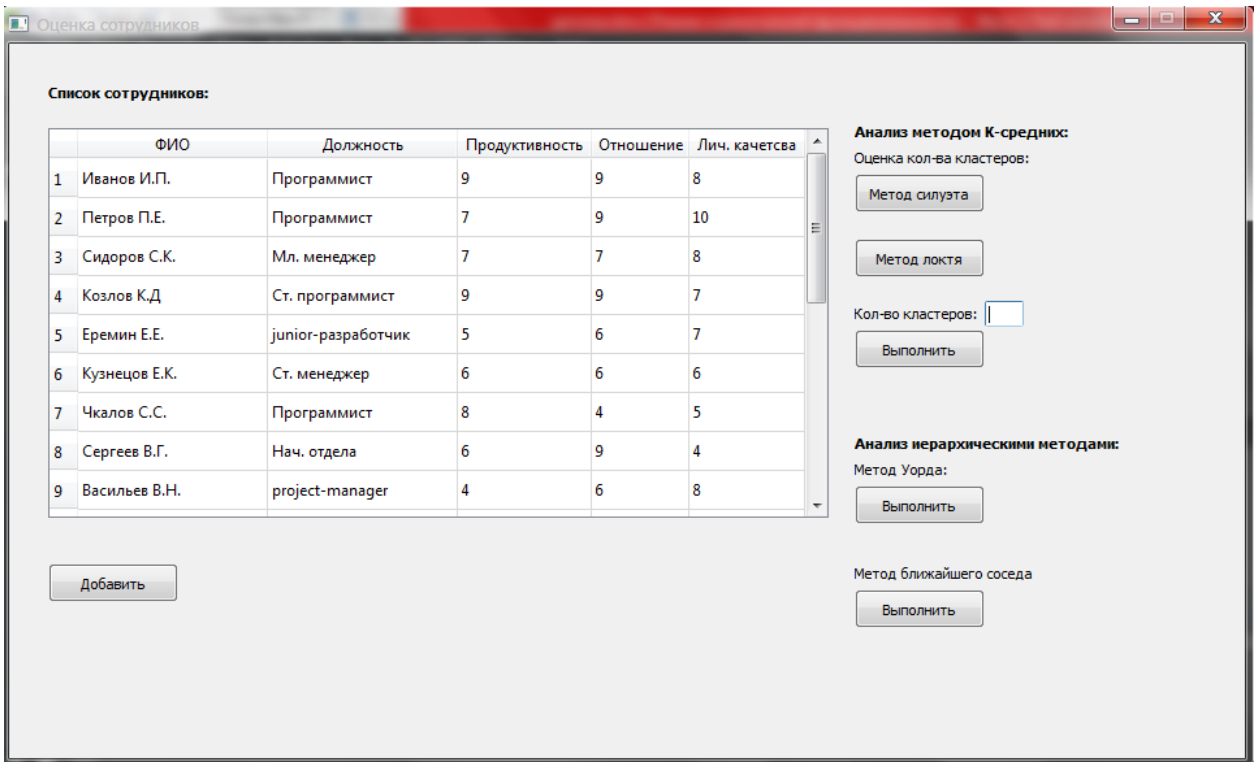

Рисунок 6 – Главное окно

На рисунке 6 видна таблица сотрудников. При нажатии на одну из ее строк откроется окно, позволяющее изменить данные о соответствующем сотруднике или удалить запись о нем из базы данных. Это окно показано на рисунке 7.

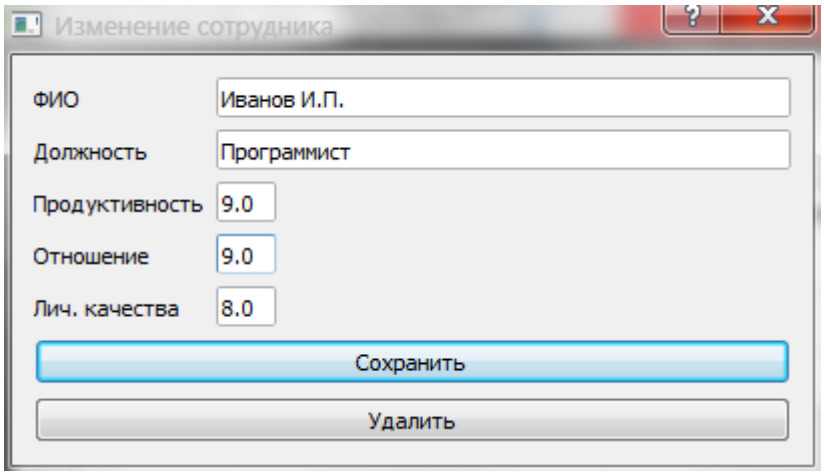

Рисунок 7 – Окно редактирования сотрудника

Кнопка «добавить» вызовет схожее окно, не вместо редактирования и удаления, позволяет внести в базу нового сотрудника с его оценками. Окно показано на рисунке 8.

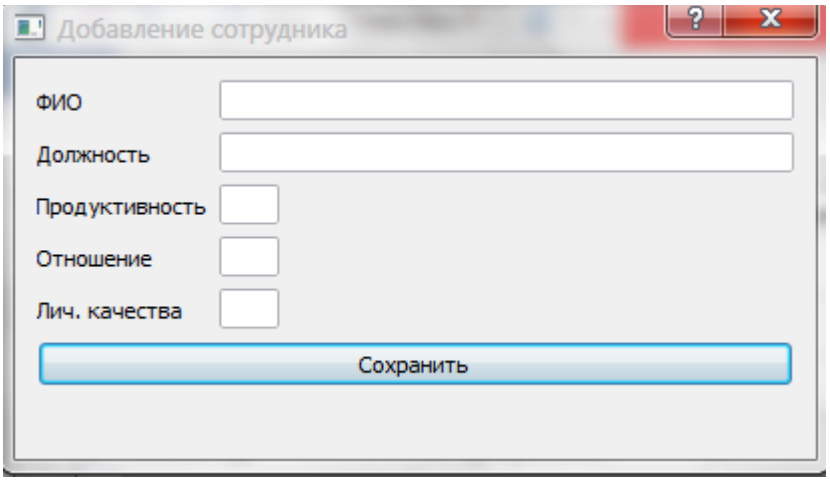

Рисунок 8 – Окно добавления сотрудника

Продемонстрируем, непосредственно, работу программы на введенной выборке. Начнем с метода К-средних. Для начала, оценим количество кластеров с помощью обоих доступных в программе методов.

При нажатии кнопки «Метод силуэта» вызывается функция, последовательно проводящая анализ методом К-средних с увеличением числа кластеров. Метод силуэта требует как минимум двух кластеров, поэтому число кластеров ограничено интервалом от 2 до 10, и на каждой итерации вычисляется коэффициент силуэта по формуле (25). По завершении итераций строится график зависимости коэффициента от числа кластеров, показанный на рисунке 9.

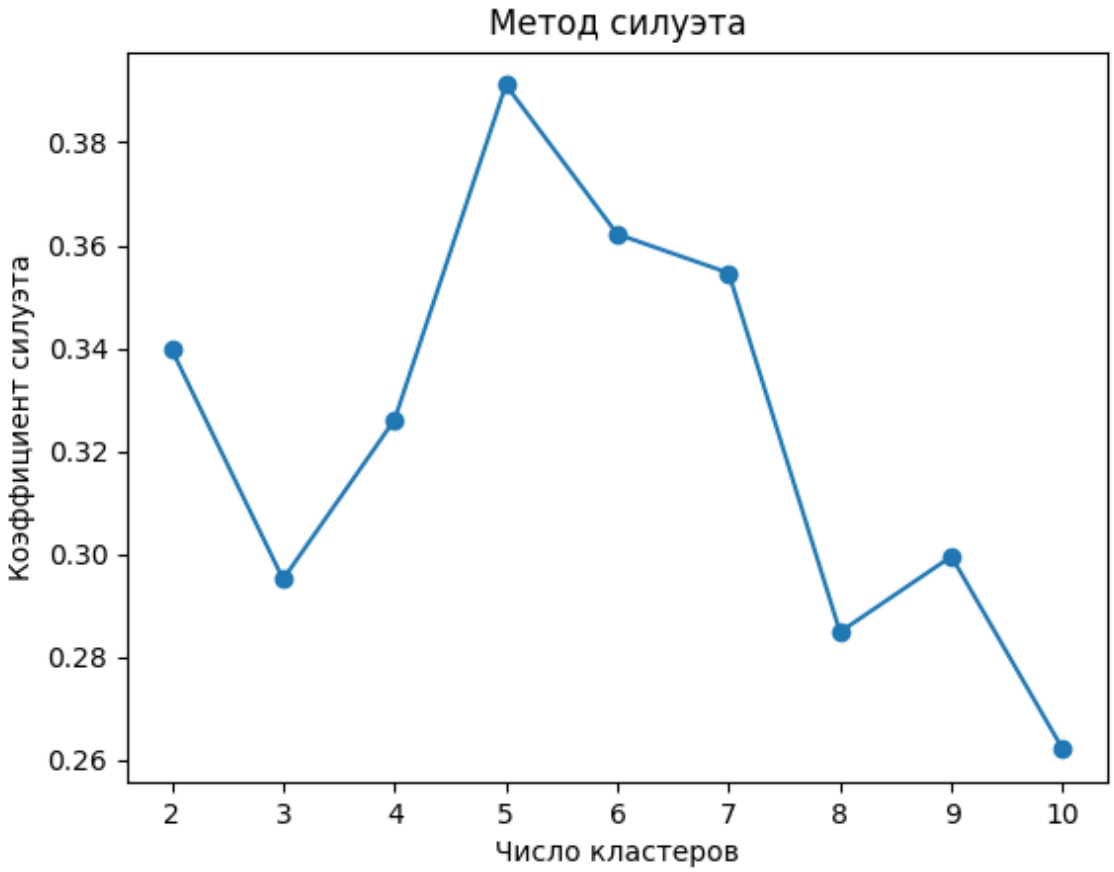

Рисунок 9 – Результат оценки числа кластеров

Видно, что данный метод однозначно выделил 5 как оптимальное число разбиений. Чего нельзя сказать о методе локтя, который, следуя схожему алгоритму, но с другой метрикой, показывает, что оптимальное число находится где-то в диапазоне от 5 до 10.

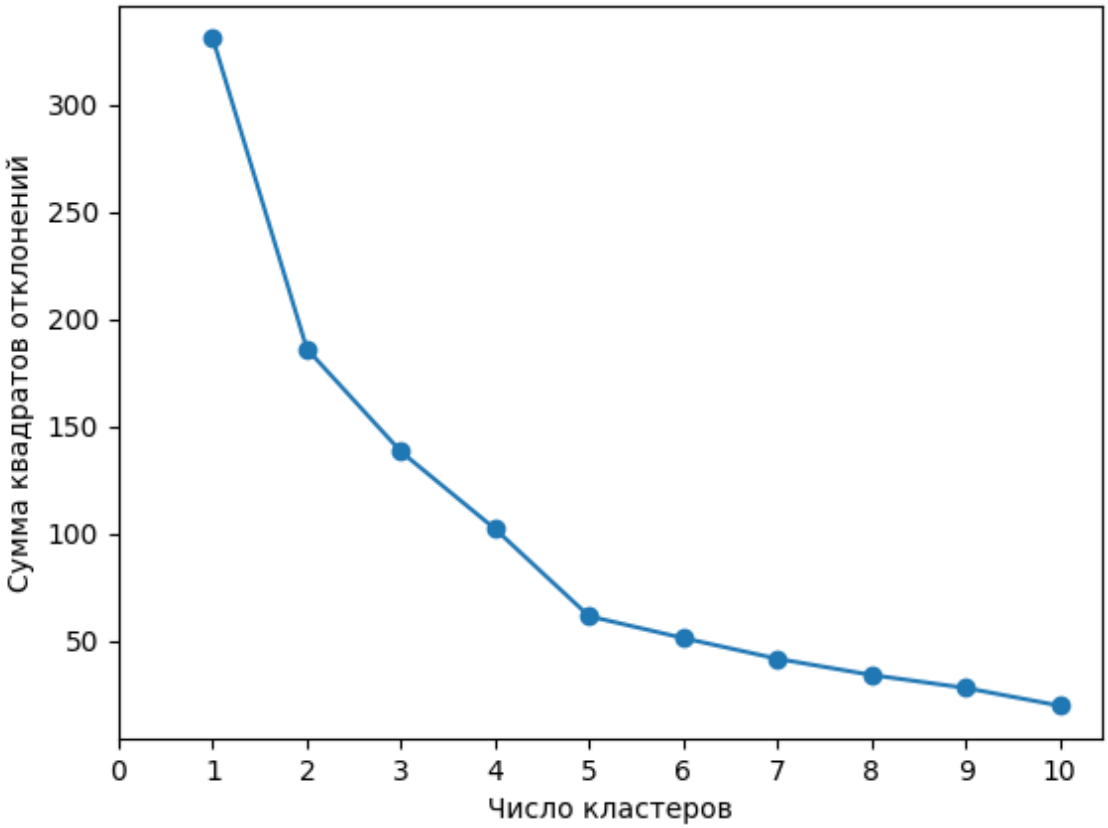

Рисунок 10 – Результат оценки методом локтя

Сравнивая рисунки 9 и 10 можно еще раз убедиться в превосходстве метода силуэта. Однако, это не всегда так. На менее «удачных» наборах данных метод силуэта также может не дать однозначного результата, и тогда оптимальное количество кластеров можно установить только сопоставляя результаты работы обоих методов. Пример представлен ниже на рисунках 11 и 12.

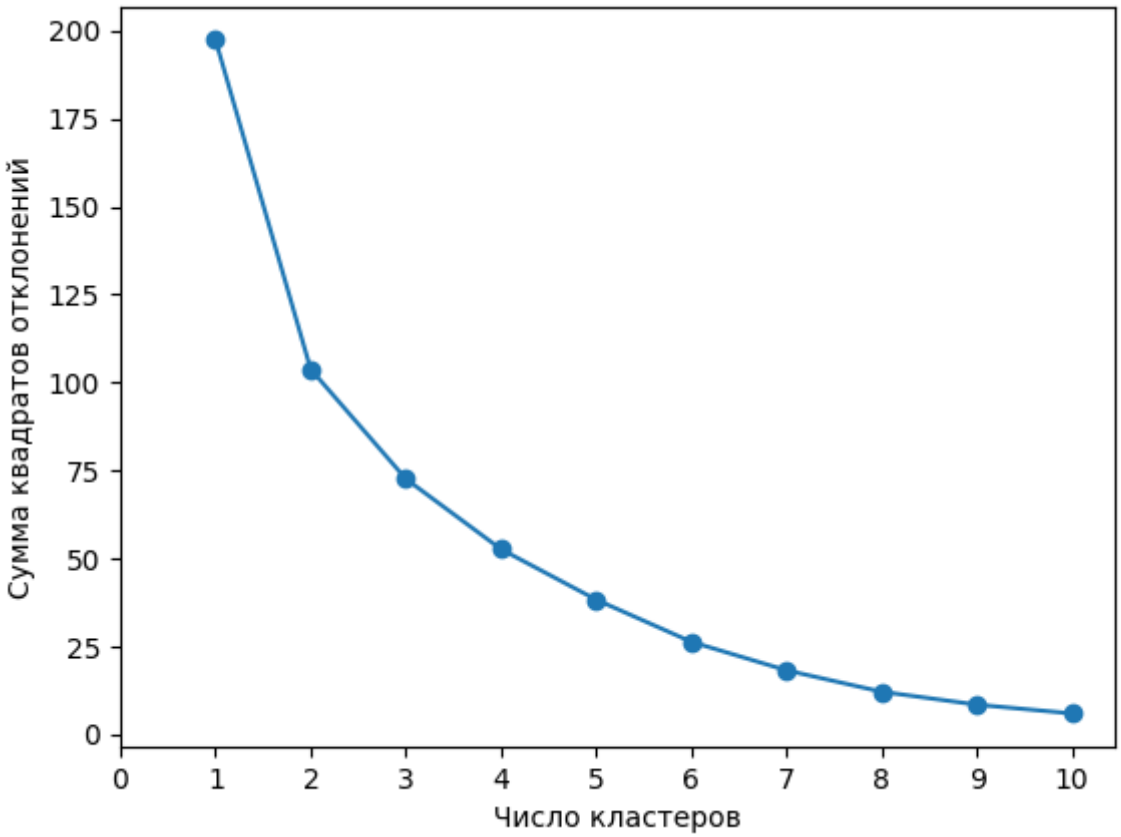

Рисунок 11 – Метод локтя на неудачном наборе данных

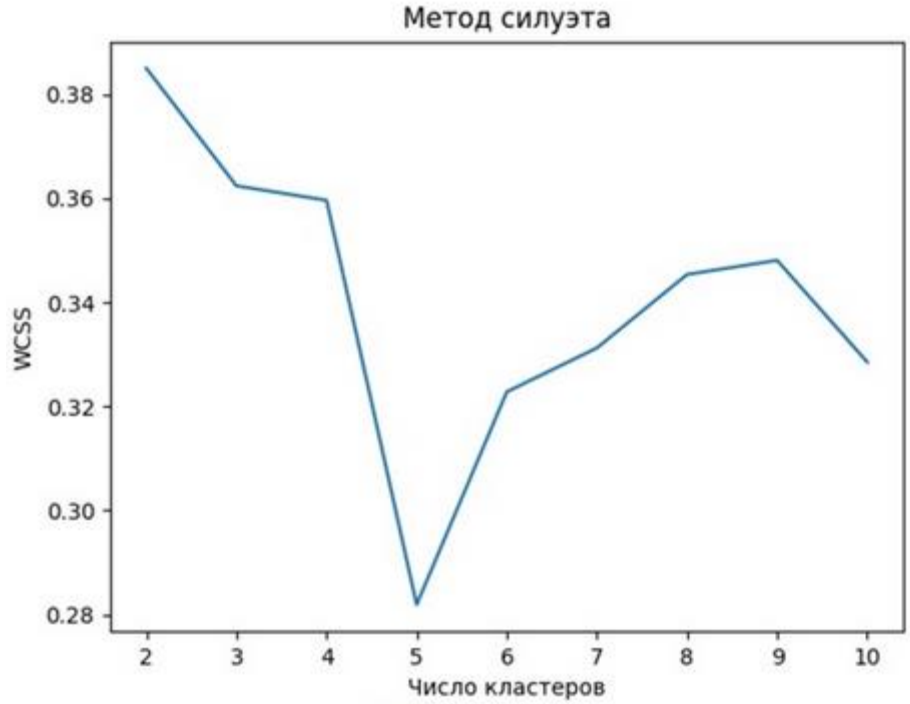

Рисунок 12 – Метод силуэта на неудачном наборе данных

На рисунках 11 и 12 видно, что коэффициент силуэта достигает максимальных значений при количестве кластеров от 2 до 4, тогда как метод

локтя указывает на интервал от 4 до 8. Логично предположить, что оптимальное количество равняется 4.

Итак, установив оптимальное количество кластеров, можно приступить в выполнению метода К-средних. После ввода числа кластеров в соответствующее поле и запуска метода, производится выбор начальных центроидов. Выбор не полностью случаен, центроиды выбираются максимально далекими друг от друга [10], чтобы избежать так называемой ловушки инициализации. После этого вычисления происходят по алгоритму Элкана. Визуальное разбиение и итоговое содержимое кластеров выводятся в отдельных окнах, показанных на рисунках 13 и 14.

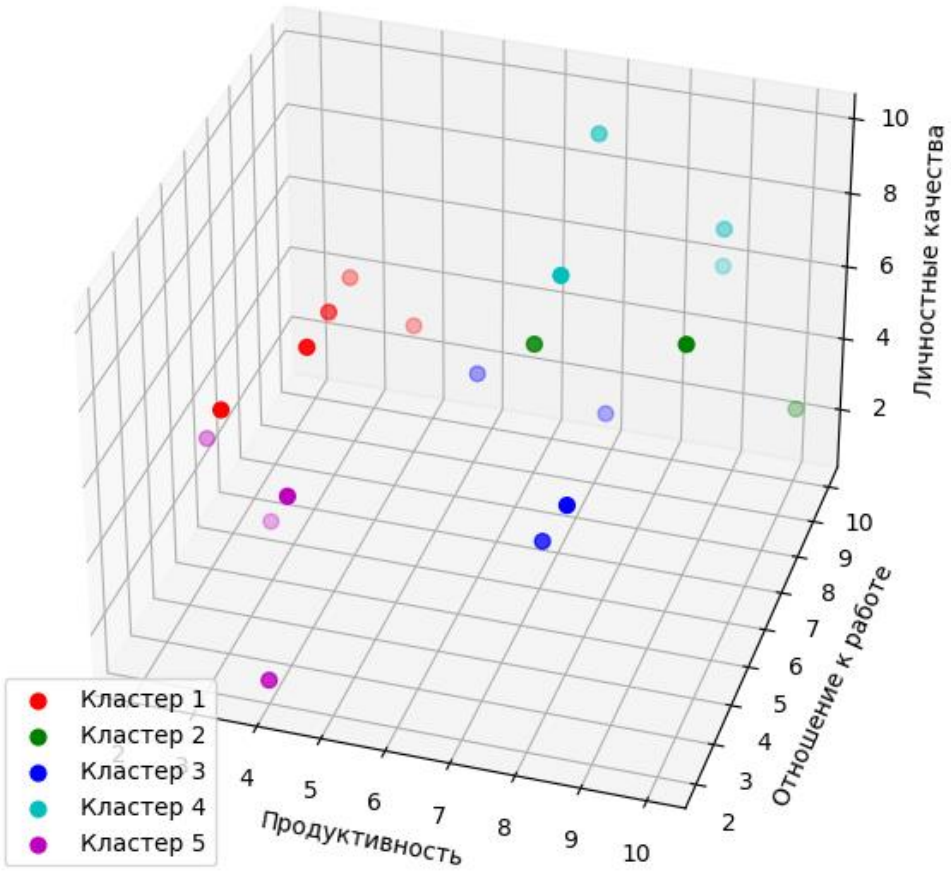

Рисунок 13 – Интерактивный график разбиения

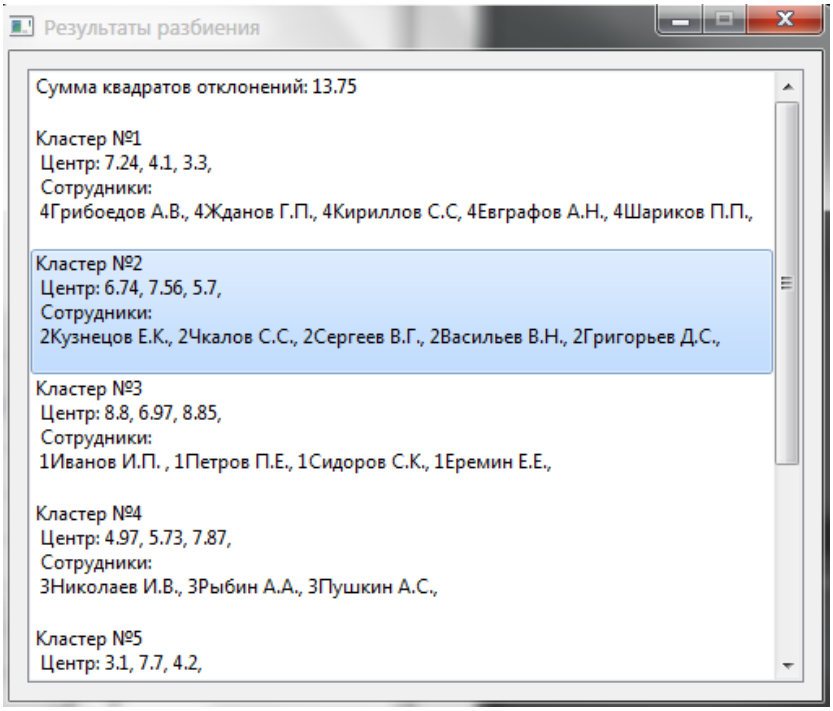

Рисунок 14 – Результат в виде текста

Теперь рассмотрим методы иерархического анализа. Они более прямолинейны и не требуют предварительной оценки числа кластеров, а потому могут выполняться без дополнительных действий.

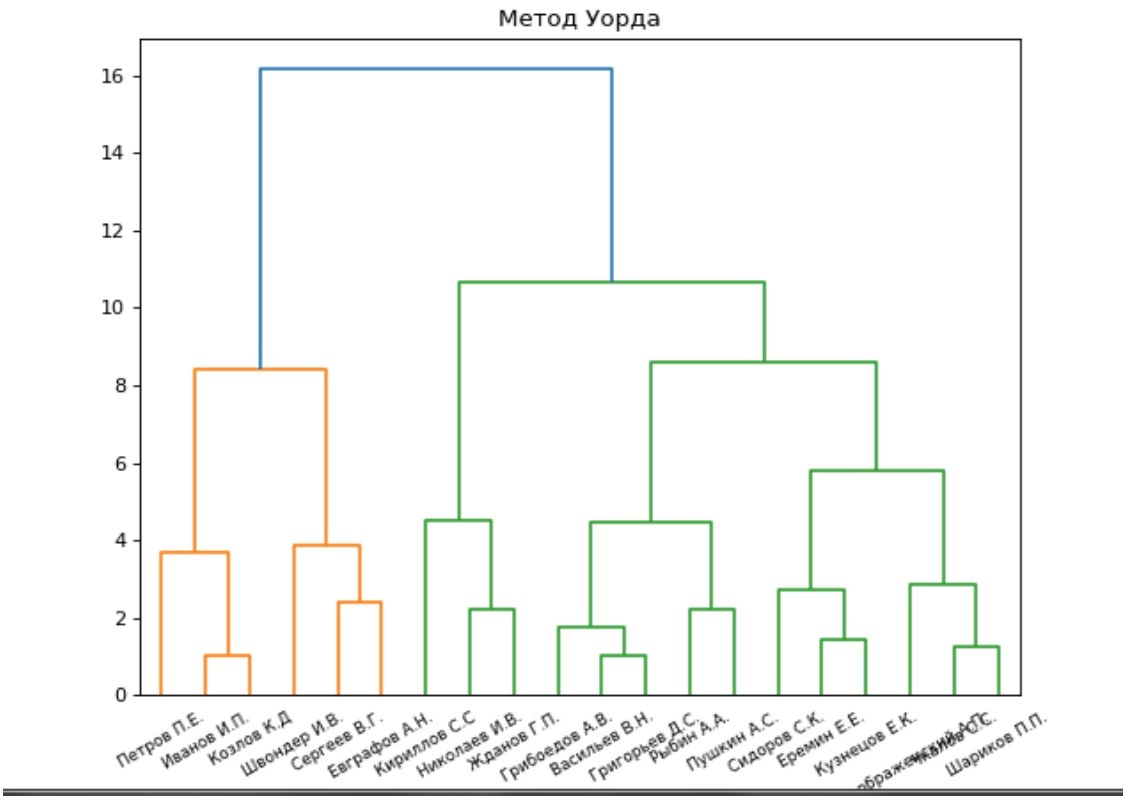

Рисунок 15 – Результат работы метода Уорда

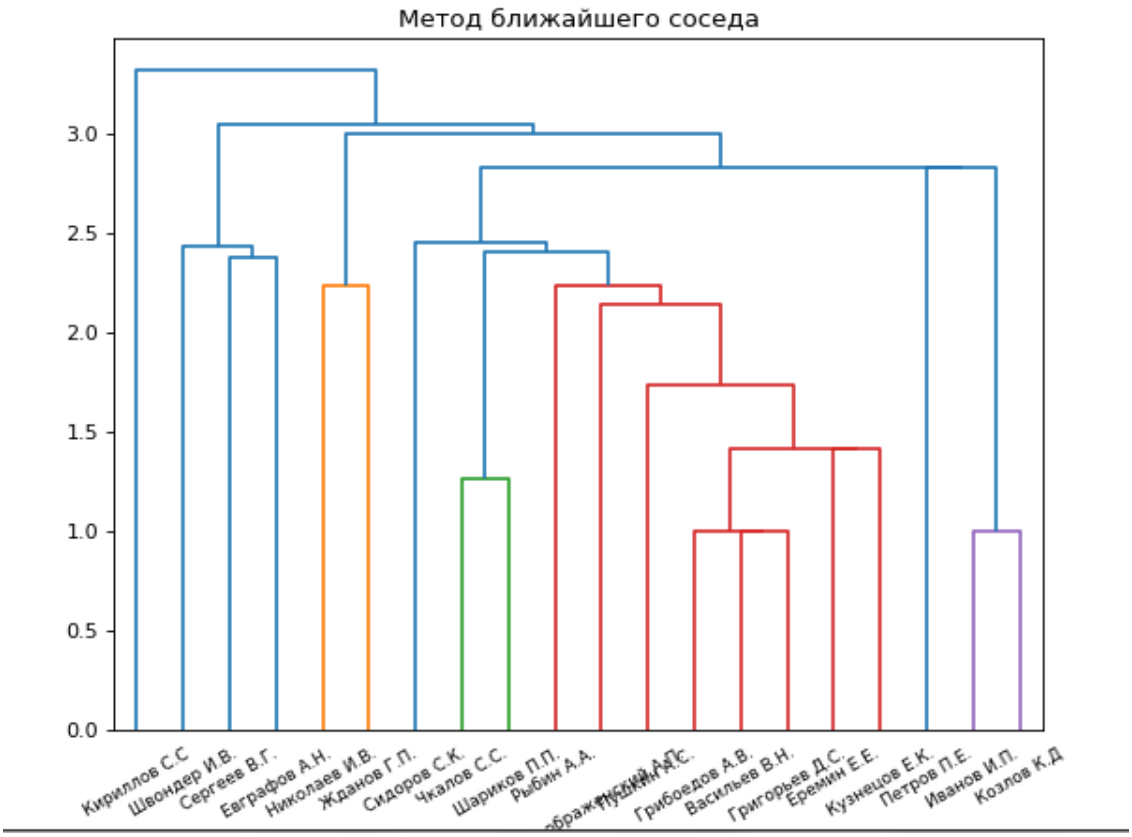

Рисунок 16 – Результат работы метода ближайшего соседа

Сразу бросается в глаза различие результатов, даваемых этими методами, видимое на рисунках 15 и 16. Это можно объяснить различиями в подходе: метод Уорда осуществляет подбор комбинаций кластеров с целью минимизации суммы квадратов расстояний, тогда как метод ближайшего соседа соединяет ближайшие элементы. Следует отметить, что разбиение на рисунке 5.12 схоже, хоть и не совпадает с разбиением, полученным методом К-средних.

5.4 Сопоставление результатов

Выбранная для иерархического анализа структура данных делает крайне затруднительным вычисление каких-либо метрик качества кластеризации, например, суммы квадратов отклонений элементов кластеров от их центров. Поэтому для сравнения эффективности методов придется прибегнуть к эмпирическому анализу результатов их работы.

Повторим анализ тех же 20 сотрудников, но изменим данные так, чтобы сотрудники составляли 5 четко выраженных групп. Для наглядности включим номер группы в поле «должность» и в имя сотрудника. Таблица 8 отображает распределение групп.

Таблица 8 – Распределение оценок по группам

	Группа 1	Группа 2	Группа 3	Группа 4	Группа 5
Продуктивность сотрудника	8-10	6-8	4-6	6-8	2-4
Отношение к работе	6-8	7-9	5-7	3-5	7-9
Личностные качества	8-10	5-7	7-9	2-4	3-5

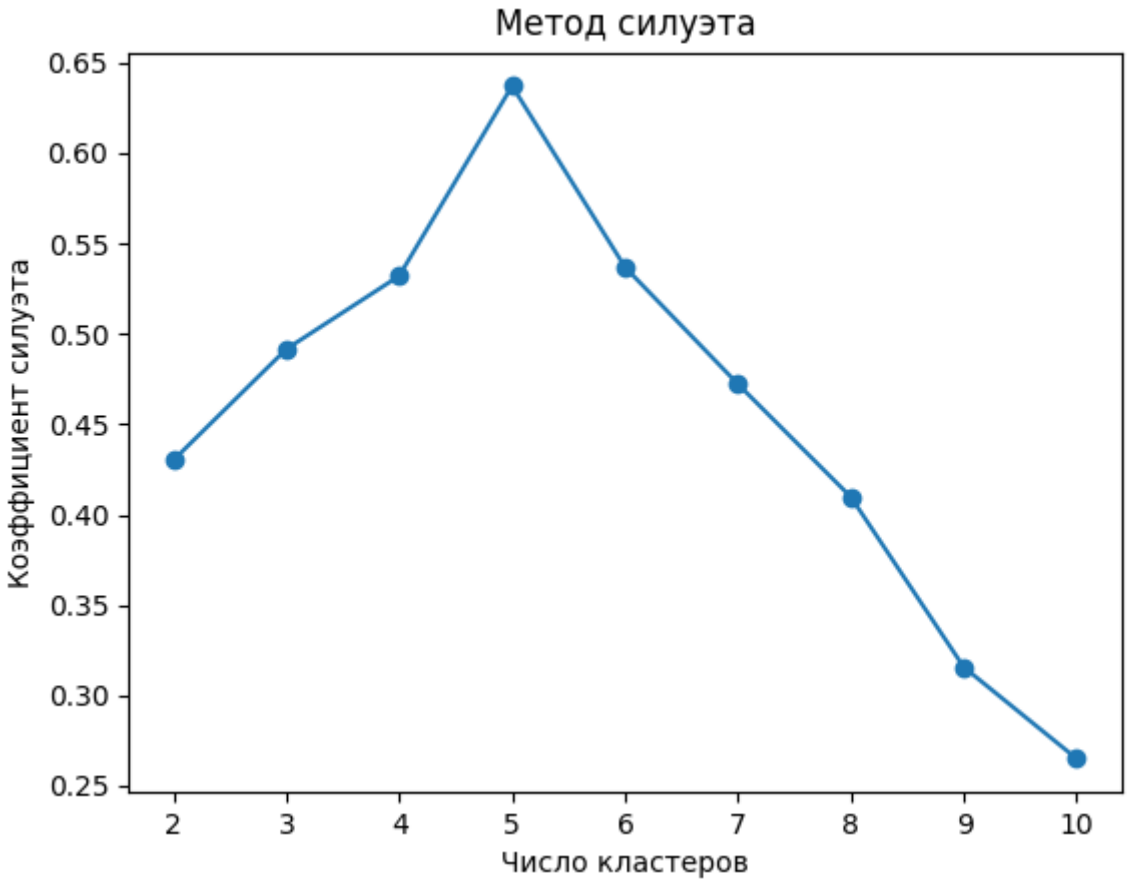

Рисунок 17 – Ретод силуэта

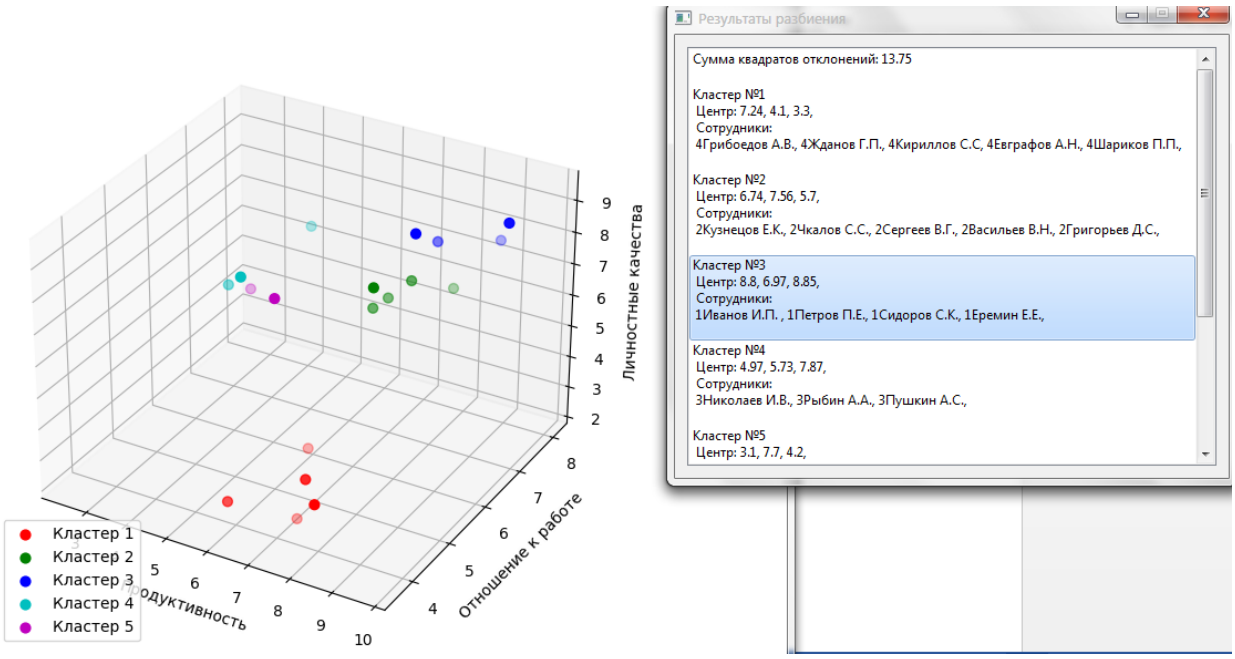

Рисунок 18 – Результат методом К-средних

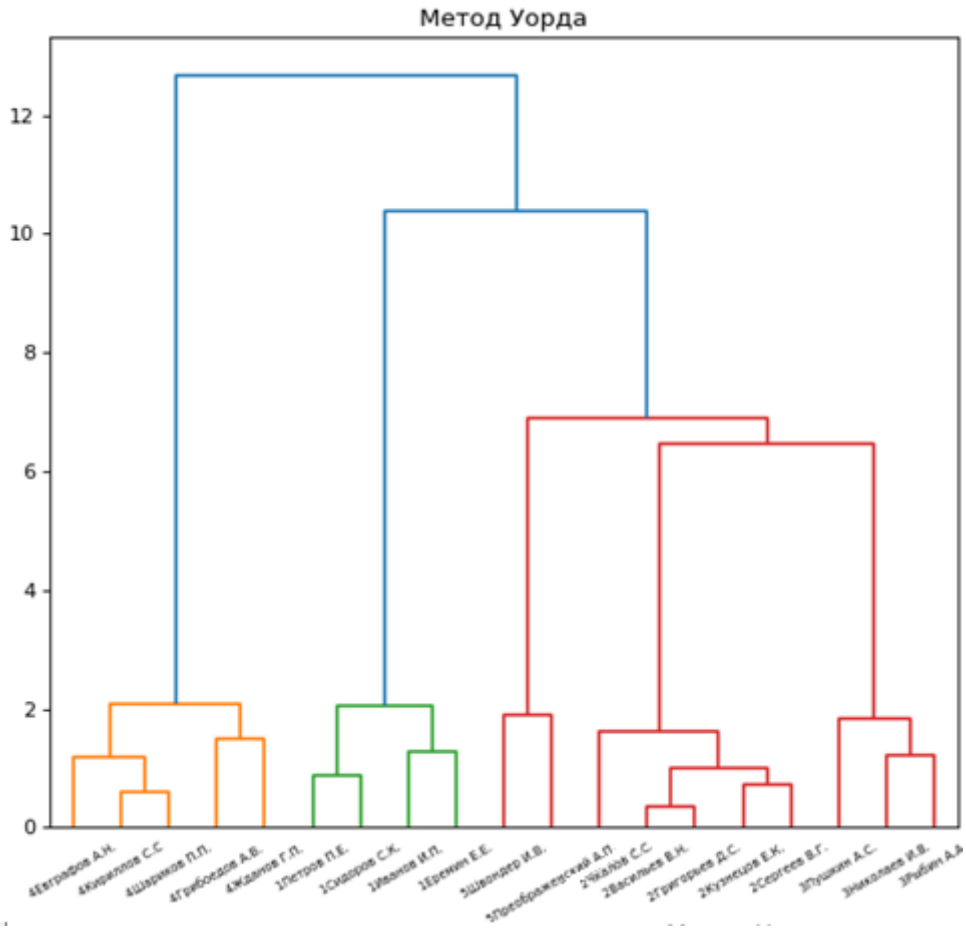

Рисунок 19 – Результат методом Уорда

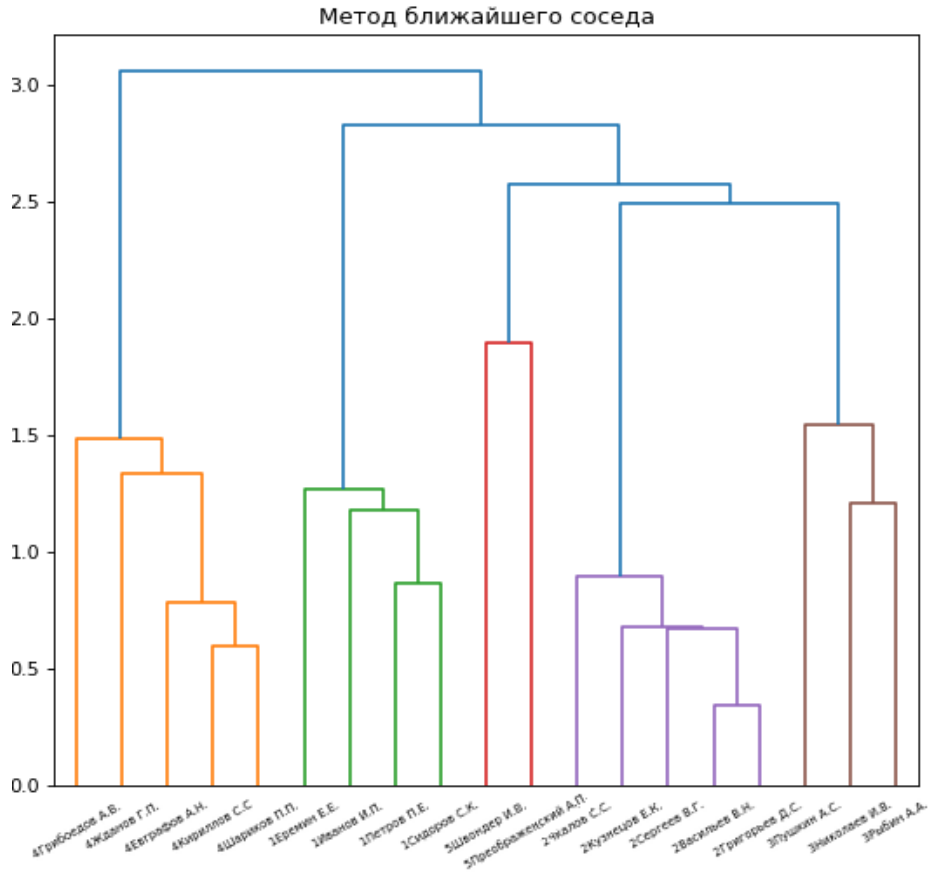

Рисунок 20 – Результат методом ближайшего соседа

На рисунках 17-20 видно, что методы К-средних и ближайшего соседа дали абсолютно корректное разбиение. При применении метода Уорда группы 2, 3 и 5 выделяются, что видно на дендрограмме, но расстояние между ними недостаточно для того, чтобы программа отметила их разными цветами.

Сгустим группы, сократив разброс в оценках между ними, приближая ситуацию к реальной. Новое распределение показано в таблице 9.

Таблица 9 – Распределение оценок по группам

	Группа 1	Группа 2	Группа 3	Группа 4	Группа 5
Продуктивность сотрудника	8-10	6-8	4-6	6-8	2-4
Отношение к работе	6-8	7-9	5-7	6-8	7-9
Личностные качества	4-6	5-7	5-7	2-4	3-5

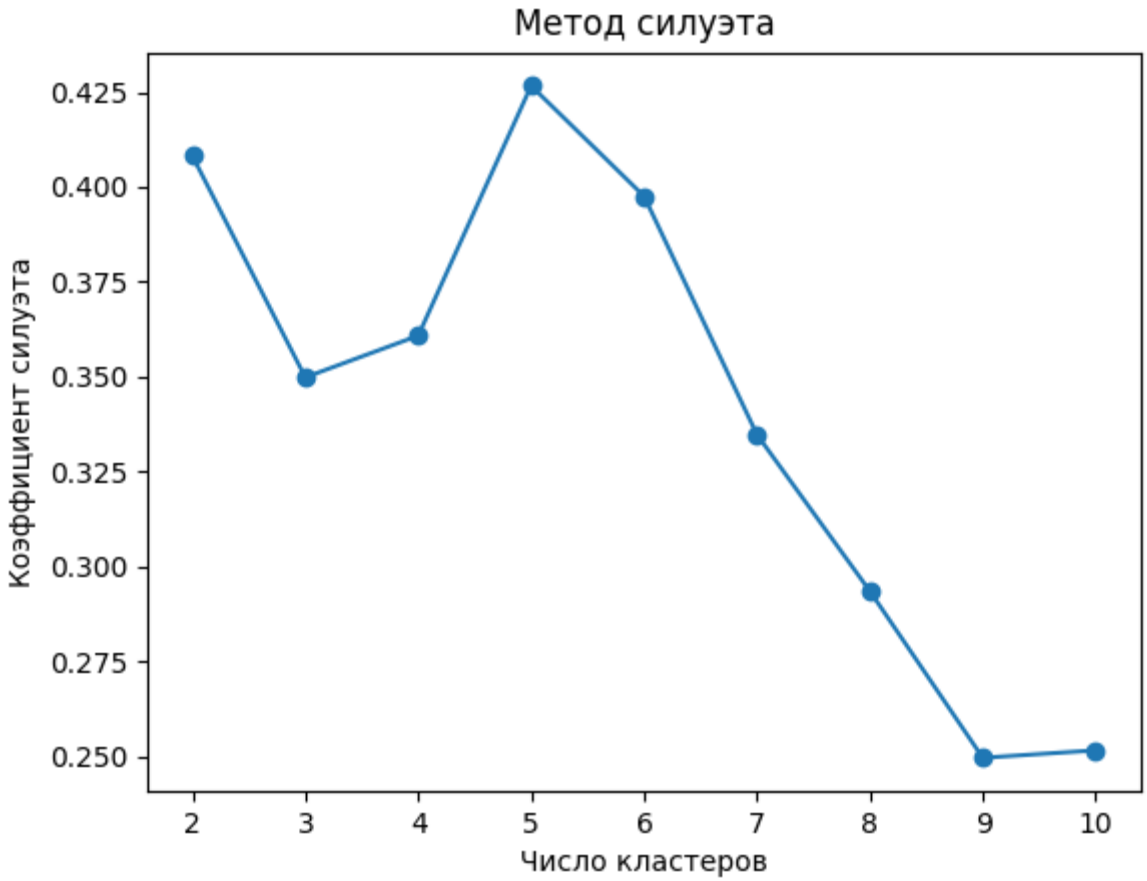

Рисунок 21 – Метод силуэта

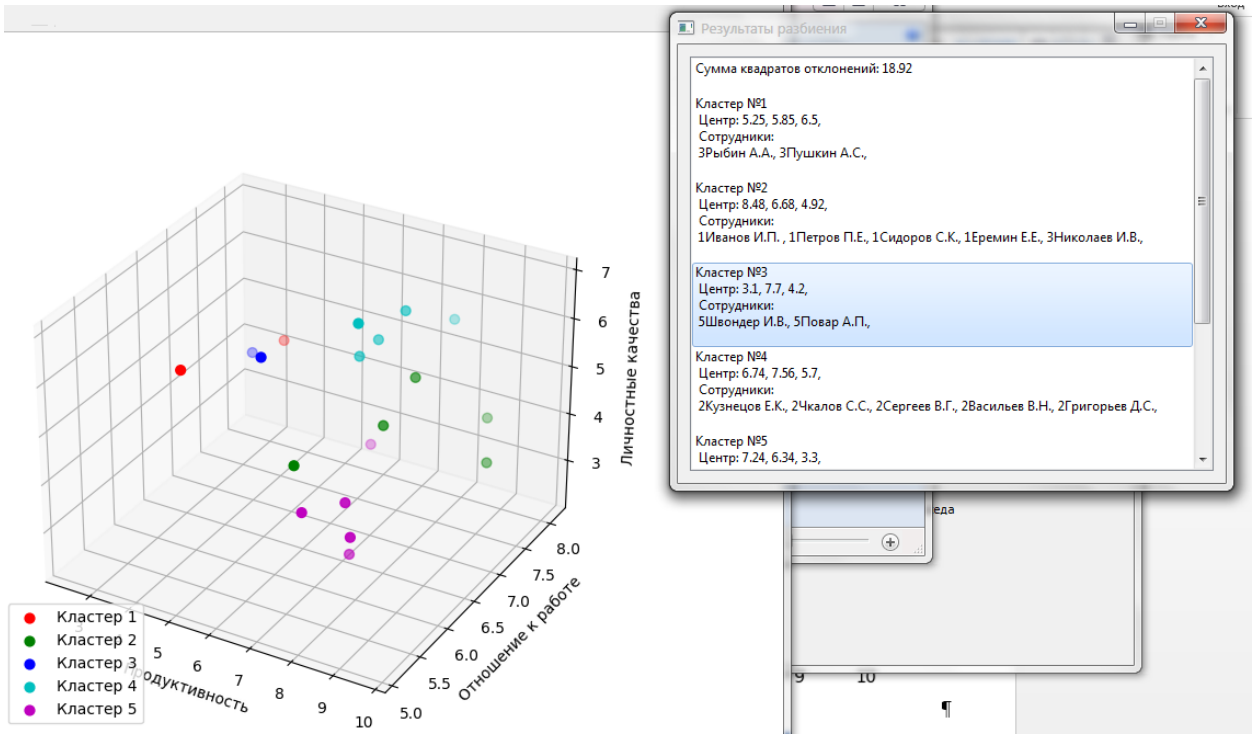

Рисунок 22 – Результат методом К-средних

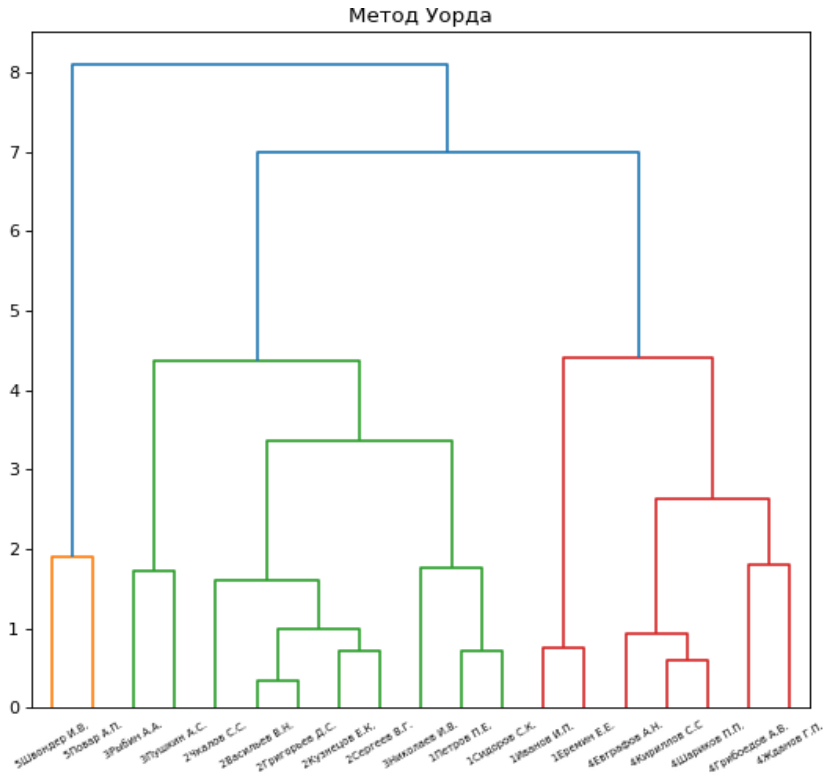

Рисунок 23 – Результат методом Уорда

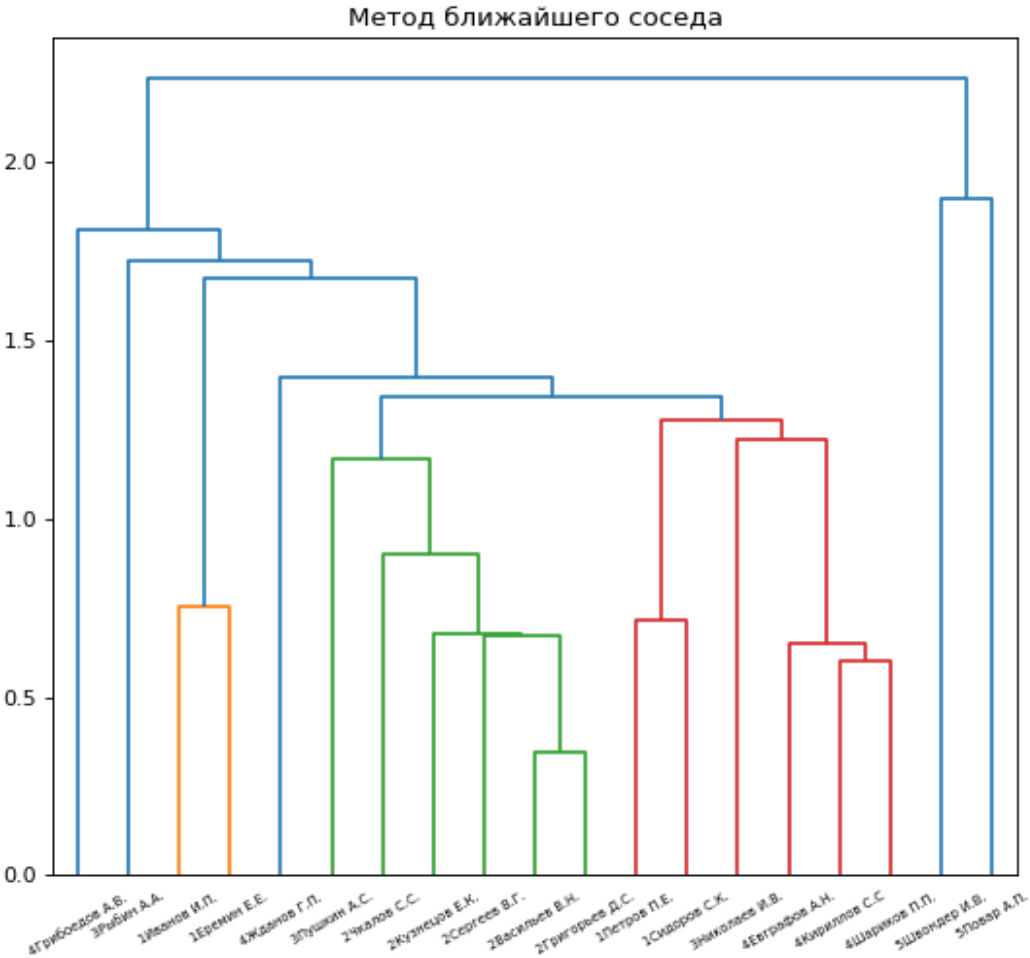

Рисунок 24 – Результат методом ближайшего соседа

На рисунках 21-24 можно видеть, что на данном наборе сотрудников метод К-средних все еще выдал почти безупречный результат с точностью до нумерации кластеров. Группы оказались слишком близки, чтобы алгоритм построения дендрограммы выделил кластеры цветом, но по самим графикам видно, что в иерархической структуре метода Уорда выделяются ожидаемые 5 кластеров, за исключением кластеров 1 и 3, которые разбиты не так, как ожидалось. Впрочем, сумма квадратов отклонений составила около 22, что говорит о том, что такое разбиение также уместно. Алгоритм ближайшего соседа выдал результат, совершенно не похожий на ожидаемый, однако, сумма квадратов отклонений составила всего около 24, так что, такое разбиение также нельзя назвать неприемлемым.

Таким образом можно отметить, что метод ближайшего соседа дает более крупные кластеры, когда расстояние между предполагаемыми кластерами невелико, в силу определения расстояния между кластерами как расстояния между ближайшими их элементами. При малом расстоянии между группами ближайшим соседом для элемента одной группы или кластера, из них составленного, может оказаться элемент другой группы, что дает смешение кластеров или даже объединение нескольких групп, как показано на рисунке 24.

Алгоритм метода Уорда требует доработки, которая позволит ему распознавать группы сотрудников подобные показанным на рисунке 23, как отдельные кластеры, с большей точностью. В остальном его эффективность удовлетворительна.

Метод К-средних дает стабильные результаты и, при условии применения техники избегания ловушки инициализации и ускорения вычислений, является наилучшим.

ЗАКЛЮЧЕНИЕ

Итого, в данной работе был рассмотрен ряд методик оценки и кластеризации. Наилучшими признаны методики кластерного анализа, поскольку они позволяют не только ранжировать сотрудников от лучших к худшим, но и сгруппировать их по схожим качествам.

Конечным результатом работы стало разработанное десктоп-приложение для оценки сотрудников. Оно позволяет хранить информацию о сотрудниках и выполнять их оценку методами К-средних, Уорда и ближайшего соседа с визуализацией результатов.

Полученные результаты были сравнены в той мере, в какой это допускала структура данных. Было установлено, что метод ближайшего соседа плохо справляется с классификацией сотрудников, когда предполагаемые кластеры расположены близко. Метод К-средних показал стабильные результаты, но установление оптимального числа кластеров для него может быть затруднительно на некоторых наборах данных.

Дальнейшая работа должна быть направлена на усовершенствование программы. Желателен экспорт и сохранение результатов, а также мелкие улучшения интерфейса. Также нужно добавить возможность вводить дополнительные критерии и импортировать информацию о сотрудниках, а также доработать визуализацию результатов, сделав ее интерактивной.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Мизинцева, М.Ф. Оценка персонала. Учебник и практикум/ М.Ф. Мизинцева ,А.Р. Сардарян. – М.: Юрайт, 2017. – 378 с.
2. Лухманова, А.С. Диагностика соискателя / А.С. Лухманова, Н.А. Сидорова, О.М. Багомедова и др. М.: Научная книга, 2005. – 250 с.
3. Кибанов, А.Я. Управление персоналом организации / А.Я.Кибанов, И.А. Баткаева, Л.В. Ивановская. НИЦ ИНФРА-М, 2021. – 695 с.
4. Белкин, А. Р. Принятие решений: комбинаторные модели аппроксимации информации / А.Р. Белкин, М.Ш. Левин. – Москва «Наука», 1990. –151с.
5. Белкин, А.Р. Желательные свойства оптимальных линейных упорядочений / Техн. Кибернетика. –1987. –№2. –с.3-21.
6. Ушаков, И.А. Задача о выборе предпочтительного объекта / И.А. Ушаков Техн. – Кибернетика. –1971. –№4. –С.3-8.
7. Odell, P.L. Cluster analisys: a survey/ P.L. Odell, B.S. Duran.Springer-Verlag, 1974
8. Мандель, И.Д. Кластерный анализ/ И.Д.Мандель. М.: Финансы и статистика, 1988. – 176 с.
9. Ajitesh, Kumar. Elbow Method vs Silhouette Score – Which is Better? November 28, 2021 // (Engl). – URL: https://vitalflux.com/elbow-method-silhouette-score-which-better/[14.06.2022]
10. Nipun, Gupta. Using Triangle Inequality to accelerate K-means. May 15, 2015 // (Engl). – URL: https://www.cse.iitd.ac.in/~rjaiswal/2015/col870/Project/Nipun.pdf [15.06.2022]

ПРИЛОЖЕНИЕ А

Код программы

```
import sys
from PyQt5 import QtWidgets, QtGui
from PyQt5.QtGui import QStandardItemModel
from PyQt5.QtSql import QSqlQuery, QSqlDatabase, QSqlQueryModel
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
from PyQt5.uic.properties import QtCore

from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as FigureCanvas
from PyQt5.QtWidgets import QApplication, QWidget
from scipy.cluster.hierarchy import linkage, fcluster, fclusterdata
from scipy.cluster.hierarchy import dendrogram

from sklearn.metrics import silhouette_score

# import the libraries
import numpy as np

from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D, proj3d



class EmployeeData:
    def __init__(self):
        self.names = []
        self.values = [[]]

class MainWindow(QMainWindow): # главное окно
    def __init__(self, parent=None):
        super().__init__(parent)
        self.setupDB()
        self.setupUi()

    def setupDB(self):
        # Create the connection

        con = QSqlDatabase.addDatabase("QSQLITE")

        con.setDatabaseName("employees.sqlite")

        # Open the connection

        if not con.open():
            print("Database Error: %s" % con.lastError().databaseText())

            sys.exit(1)

        # Create a query and execute it right away using .exec()

        createTableQuery = QSqlQuery()

        createTableQuery.exec(

            """
```

```
        CREATE TABLE IF NOT EXISTS employees (

            id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE NOT NULL,

            name VARCHAR(70) NOT NULL,

            job VARCHAR(50) NOT NULL,

            crit1 FLOAT NOT NULL,

            crit2 FLOAT NOT NULL,

            crit3 FLOAT NOT NULL
        )
        """
    )

def setupUi(self):
    boldFont = QtGui.QFont()
    boldFont.setBold(True)

    self.setWindowTitle("Оценка сотрудников") # заголовок окна
    self.move(0, 0) # положение окна
    self.setFixedSize(950, 550) # размер окна

    self.employeeslbl = QLabel('Список сотрудников:', self)
    self.employeeslbl.move(30, 30)
    self.employeeslbl.resize(500, 15)
    self.employeeslbl.setFont(boldFont)

    tabmodel = QSqlQueryModel()
    tabmodel.setQuery("""SELECT name as 'ФИО',
                             job as 'Должность',
                             crit1 as 'Продуктивность',
                             crit2 as 'Отношение',
                             crit3 as 'Лич. качетсва'
                             FROM employees""")

    self.list = QTableView(self)
    self.list.setModel(tabmodel)
    self.list.clicked.connect(self.onTableClick)
    self.list.move(30, 65)
    self.list.resize(600,300)

    header = self.list.horizontalHeader()
    header.setSectionResizeMode(0, QtWidgets.QHeaderView.Stretch)
    header.setSectionResizeMode(1, QtWidgets.QHeaderView.Stretch)
    header.setSectionResizeMode(2, QtWidgets.QHeaderView.ResizeToContents)
    header.setSectionResizeMode(3, QtWidgets.QHeaderView.ResizeToContents)
    header.setSectionResizeMode(4, QtWidgets.QHeaderView.ResizeToContents)

    self.addbtn = QPushButton('Добавить', self)
    self.addbtn.setObjectName("addbtn")
    self.addbtn.move(30, 400)
    self.addbtn.clicked.connect(self.onBtnClick)

    self.Klbl = QLabel('Анализ методом К-средних:', self)
    self.Klbl.move(650, 60)
    self.Klbl.resize(500, 15)
    self.Klbl.setFont(boldFont)

    self.estlbl = QLabel('Оценка кол-ва кластеров:', self)
    self.estlbl.move(650, 80)
    self.estlbl.resize(500, 15)
```

```
        self.Kbtn = QPushButton('Метод силуэта', self)
        self.Kbtn.setObjectName("silbtn")
        self.Kbtn.move(650, 100)
        self.Kbtn.clicked.connect(self.onBtnClick)

        self.Kbtn = QPushButton('Метод локтя', self)
        self.Kbtn.setObjectName("elbbtn")
        self.Kbtn.move(650, 150)
        self.Kbtn.clicked.connect(self.onBtnClick)

        self.employeeslbl = QLabel('Кол-во кластеров:', self)
        self.employeeslbl.move(650, 200)
        self.employeeslbl.resize(500, 15)

        self.nedit = QLineEdit(self)
        self.nedit.move(750, 198)
        self.nedit.setFixedWidth(30)
        self.nedit.setFixedHeight(20)

        self.Kbtn = QPushButton('Выполнить', self)
        self.Kbtn.setObjectName("kbtn")
        self.Kbtn.move(650, 220)
        self.Kbtn.clicked.connect(self.onBtnClick)

        self.Hlbl = QLabel('Анализ иерархическими методами:', self)
        self.Hlbl.move(650, 300)
        self.Hlbl.resize(500, 15)
        self.Hlbl.setFont(boldFont)

        self.wlbl = QLabel('Метод Уорда:', self)
        self.wlbl.move(650, 320)
        self.wlbl.resize(500, 15)

        self.wardbtn = QPushButton('Выполнить', self)
        self.wardbtn.setObjectName("wardbtn")
        self.wardbtn.move(650, 340)
        self.wardbtn.clicked.connect(self.onBtnClick)

        self.slbl = QLabel('Метод ближайшего соседа', self)
        self.slbl.move(650, 400)
        self.slbl.resize(500, 15)

        self.singlebtn = QPushButton('Выполнить', self)
        self.singlebtn.setObjectName("singlebtn")
        self.singlebtn.move(650, 420)
        self.singlebtn.clicked.connect(self.onBtnClick)

    @pyqtSlot()
    def onBtnClick(self):
        sender_name=str(self.sender().objectName())
        if (sender_name == "silbtn"):
            edata=self.getData()
            X = edata.values
            silhouette_coefficients = []

            for k in range(2, 11):
                kmeans = KMeans(n_clusters=k, init='k-means++', max_iter=300, n_init=10,
random_state=0)
                kmeans.fit(X)
                score = silhouette_score(X, kmeans.labels_)
                silhouette_coefficients.append(score)
            plt.plot(range(2, 11), silhouette_coefficients, marker='o')
            plt.title('Метод силуэта')
            plt.xlabel('Число кластеров')
            plt.ylabel('Коэффициент силуэта')
```

```
            plt.show()
        if (sender_name == "elbbtn"):
            edata=self.getData()
            X = edata.values
            WCSS = []
            for i in range(1, 11):
                model = KMeans(n_clusters=i, init='k-means++')
                model.fit(X)
                WCSS.append(model.inertia_)
            plt.plot(range(1, 11), WCSS, marker='o')
            plt.xticks(np.arange(11))
            plt.xlabel("Число кластеров")
            plt.ylabel("Сумма квадратов отклонений")
            plt.show()
        if (sender_name == "kbtn"):
            if self.nedit.text() != '':

                edata=self.getData()
                num=int(self.nedit.text())

                x = edata.values
                names=edata.names
                model = KMeans(n_clusters=num, init="k-means++", max_iter=300, n_init=10,
random_state=0)


                y_clusters = model.fit_predict(x)
                inert = model.inertia_
                clusters=[[]]
                namesInClusters=[[]]
                for k in range(0, num):
                    clusters.append([])
                    namesInClusters.append([])
                c=0
                for i in x:
                    for k in range(0, num):
                        if y_clusters[c]==k:
                            clusters[k].append(i)
                            namesInClusters[k].append(names[x.index(i)])
                    c=c+1

                for k in range(0, num):
                    clusters[k]=[x for xs in clusters[k] for x in xs]

                #namesInClusters = [x for xs in namesInClusters for x in xs]

              #  flatclusters=[x for xs in clusters for x in xs]
              #  xs = flatclusters[0::3]
              #  ys = flatclusters[1::3]
                #zs = flatclusters[2::3]

                centers=[[]]
                for k in range(0, num):
                    centers.append([])
                    centers[k].append(sum(clusters[k][0::3])/len(clusters[k][0::3]))
                    centers[k].append(sum(clusters[k][1::3])/len(clusters[k][0::3]))
                    centers[k].append(sum(clusters[k][2::3])/len(clusters[k][0::3]))

                #sns.countplot(y_clusters)
                fig = plt.figure(figsize=(9, 9))
                ax = fig.add_subplot(111, projection='3d')
                colors=['r','g','b','c','m','y','orange', 'indigo','ivory', 'yellow']

                for k in range(0, num):
```

```
                    ax.scatter(clusters[k][0::3], clusters[k][1::3], clusters[k][2::3],
s=40, color=colors[k],
                               label=f"Кластер {k+1}", picker=True)
                    #for i,name in namesInClusters:

                #def annotate_onclick(event):
                #    print(event.ind)
                 #   point_index = int(event.ind[0])
                 #   print(point_index)
                  #  proj = ax.get_proj()
                   # x_p, y_p, _ = proj3d.proj_transform(xs[point_index],
ys[point_index], zs[point_index], proj)

                  #  annotation=plt.annotate(str(namesInClusters[point_index]), xy=(x_p,
y_p))

                   # plt.pause(3)
                   # annotation.remove()
                   # fig.canvas.draw_idle()

               # fig.canvas.mpl_connect('pick_event', annotate_onclick)

                ax.set_xlabel('Продуктивность')
                ax.set_ylabel('Отношение к работе')
                ax.set_zlabel('Личностные качества')
                ax.legend(loc='lower left')
                print(namesInClusters)
                plt.show()
                self.w=KWindow(namesInClusters,centers,num,inert)
                print(namesInClusters)
                self.w.show()
        if (sender_name == "wardbtn"):
            edata = self.getData()
            X = edata.values
            names = edata.names
            Z = linkage(X, 'ward')
            flat=fclusterdata(Z,t=1.154)
            print(flat)
            plt.figure(figsize=(8, 6), dpi=80)
            plt.title("Метод Уорда")
            dendrogram(Z, leaf_rotation=30, leaf_font_size=8, labels=names)
            plt.show()
        if (sender_name == "singlebtn"):
            edata = self.getData()
            X = edata.values
            names=edata.names
            Z = linkage(X, 'single')
            print(Z)
            plt.figure(figsize=(8, 6), dpi=80)
            plt.title("Метод ближайшего соседа")
            dendrogram(Z, leaf_rotation=30, leaf_font_size=8, labels=names)
            plt.show()
        if (sender_name == "addbtn"):
            self.w=addWindow()
            self.w.exec_()
            self.refresh()

    def getData (self):
        result=EmployeeData()
        tabmodel = QSqlQueryModel()
        tabmodel.setQuery("""SELECT * FROM employees""")
        i=0
        while True:
            if (tabmodel.data(tabmodel.index(i, 0))==None):
                break
```

```
            if i>0:
                result.values.append([])
            result.names.append(tabmodel.data(tabmodel.index(i, 1)))
            result.values[i].append(tabmodel.data(tabmodel.index(i, 3)))
            result.values[i].append(tabmodel.data(tabmodel.index(i, 4)))
            result.values[i].append(tabmodel.data(tabmodel.index(i, 5)))
            i=i+1
        return result

    def onTableClick (self, item):
        clickedid=item.row()
        tabmodel = QSqlQueryModel()
        tabmodel.setQuery("""SELECT * FROM employees""")

        self.data={}
        self.data["id"]=tabmodel.data(tabmodel.index(clickedid, 0))
        self.data["name"]=tabmodel.data(tabmodel.index(clickedid, 1))
        self.data["job"]=tabmodel.data(tabmodel.index(clickedid, 2))
        self.data["crit1"]=tabmodel.data(tabmodel.index(clickedid, 3))
        self.data["crit2"]=tabmodel.data(tabmodel.index(clickedid, 4))
        self.data["crit3"]=tabmodel.data(tabmodel.index(clickedid, 5))
        #print(self.data)

        self.w=editWindow(self.data)
        self.w.exec_()
        self.refresh()
    def refresh(self):
        self.tabmodel1 = QSqlQueryModel()
        self.tabmodel1.setQuery("""SELECT name as 'ФИО',
                                   job as 'Должность',
                                   crit1 as 'Продуктивность',
                                   crit2 as 'Отношение',
                                   crit3 as 'Лич. качетсва'
                                   FROM employees""")
        self.list.setModel(self.tabmodel1)



class KWindow(QWidget):
    def __init__(self, namesInClusters,centers,num,inert):
        super().__init__()
        print(1)
        self.setupUI(namesInClusters,centers,num,inert)

    def setupUI(self, namesInClusters, centers, num,inert, listWidget=None):
        self.move(600,0)
        self.setFixedSize(500,400)
        layout = QVBoxLayout(self)
        print(1)
        self.lview=QListWidget()
        self.lview.addItem(f"Сумма квадратов отклонений: {round(inert,2)} \n")
        print(2)
        for i in range(0,num):
            cent =''
            for c in centers[i]:
                cent+=str(round(c,2))+', '
            print(cent)
            emp=''
            print(3)
            for e in namesInClusters[i]:
                emp+=e+', '
            string=QListWidgetItem(f"Кластер №{i+1} \n Центр: {cent} \n Сотрудники: \n
{emp} \n", listWidget)
            self.lview.addItem(string)
        print(4)
```

```
        layout.addWidget(self.lview)

class HWindow(QWidget):
    def __init__(self):
        super().__init__()


class Canvas(FigureCanvas):
    def __init__(self, parent):
        fig, self.ax = plt.subplots(figsize=(5, 4), dpi=50)
        super().__init__(fig)
        self.setParent(parent)

        """
        Matplotlib Script
        """
        t = np.arange(0.0, 2.0, 0.01)
        s = 1 + np.sin(2 * np.pi * t)

        self.ax.plot(t, s)

        self.ax.set(xlabel='time (s)', ylabel='voltage (mV)',
                    title='About as simple as it gets, folks')
        self.ax.grid()

class addWindow(QDialog):
    def __init__(self):
        super().__init__()
        self.setupUi()

    def setupUi(self):
        self.setWindowTitle("Добавление сотрудника") # заголовок окна
        self.move(0, 0) # положение окна
        self.resize(400, 200) # размер окна

        self.fioedit = QLineEdit()
        self.jobedit = QLineEdit()
        self.crit1edit = QLineEdit()
        self.crit1edit = QLineEdit()
        self.crit1edit.setFixedWidth(30)
        self.crit2edit = QLineEdit()
        self.crit2edit.setFixedWidth(30)
        self.crit3edit = QLineEdit()
        self.crit3edit.setFixedWidth(30)

        self.savebtn = QPushButton('Сохранить', self)
        self.savebtn.setObjectName("savebtn")
        self.savebtn.clicked.connect(self.onClick)

        flo = QFormLayout()
        flo.addRow("ФИО", self.fioedit)
        flo.addRow("Должность", self.jobedit)
        flo.addRow("Продуктивность", self.crit1edit)
        flo.addRow("Отношение", self.crit2edit)
        flo.addRow("Лич. качества",  self.crit3edit)
        flo.addRow(self.savebtn)

        self.setLayout(flo)

    @pyqtSlot()
    def onClick(self):

        fio = self.fioedit.text()
        job = self.jobedit.text()
        crit1 = self.crit1edit.text()
```

```
        crit2 = self.crit2edit.text()
        crit3 = self.crit3edit.text()

        createTableQuery = QSqlQuery()
        createTableQuery.exec(
        f"""INSERT INTO employees (name, job, crit1,crit2,crit3)
                VALUES ('{fio}', '{job}', '{crit1}', '{crit2}', '{crit3}')"""
        )
        self.close()

class editWindow(QDialog):
    def __init__(self, data):
        super().__init__()
        self.secretcrutch = data["id"]
        self.setupUi(data)

    def setupUi(self, data):
        self.setWindowTitle("Изменение сотрудника") # заголовок окна
        self.move(0, 0) # положение окна
        self.resize(400, 200) # размер окна

        self.fioedit = QLineEdit(str(data["name"]))
        self.jobedit = QLineEdit(str(data["job"]))
        self.crit1edit = QLineEdit(str(data["crit1"]))
        self.crit1edit.setFixedWidth(30)
        self.crit2edit = QLineEdit(str(data["crit2"]))
        self.crit2edit.setFixedWidth(30)
        self.crit3edit = QLineEdit(str(data["crit3"]))
        self.crit3edit.setFixedWidth(30)

        self.savebtn = QPushButton('Сохранить', self)
        self.savebtn.setObjectName("savebtn")
        self.savebtn.clicked.connect(self.onClick)

        self.delbtn = QPushButton('Удалить', self)
        self.delbtn.setObjectName("delbtn")
        self.delbtn.clicked.connect(self.onClick)

        flo = QFormLayout()
        flo.addRow("ФИО", self.fioedit)
        flo.addRow("Должность", self.jobedit)
        flo.addRow("Продуктивность", self.crit1edit)
        flo.addRow("Отношение", self.crit2edit)
        flo.addRow("Лич. качества",  self.crit3edit)
        flo.addRow(self.savebtn)
        flo.addRow(self.delbtn)

        self.setLayout(flo)

    @pyqtSlot()
    def onClick(self):
        sender_name = str(self.sender().objectName())
        employeeid = str(self.secretcrutch)
        if sender_name == "savebtn":
            fio = self.fioedit.text()
            job = self.jobedit.text()
            crit1 = self.crit1edit.text()
            crit2 = self.crit2edit.text()
            crit3 = self.crit3edit.text()

            createTableQuery = QSqlQuery()
            createTableQuery.exec(
            f"""UPDATE employees
                    SET name='{fio}',
                     job='{job}',
```

```
                    crit1='{crit1}',
                    crit2='{crit2}',
                    crit3='{crit3}'
                   WHERE id={employeeid}"""
            )
        if sender_name == "delbtn":
            createTableQuery = QSqlQuery()
            createTableQuery.exec(
            f"""DELETE FROM employees
                    WHERE id={employeeid}"""
            )
        self.close()


if __name__ == "__main__":
    app = QApplication(sys.argv)
    win = MainWindow()
    win.show()
    sys.exit(app.exec_())
```