Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления» (ИУ)

КАФЕДРА «Системы обработки информации и управления» (ИУ5)

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

«Анализ данных вакансий с сайтов работодателей»

Студент группы ИУ5-23М _____ **Д. А. Тислюк**
(Подпись, дата)

Руководитель _____ **Ю. Е. Гапанюк**
(Подпись, дата)

2021 г.

Оглавление

Введение

В настоящее время, когда активно идет процесс информатизации и компьютеризации всех сфер деятельности человека, с каждым годом увеличивается ценность данных. Если раньше данные использовались лишь как некоторая хранимая идентификационная и/или полезная информация, например, карточка пациента поликлиники в электронном виде, то сейчас все большие обороты набирает не просто хранение, а анализ имеющейся информации с целью получения новых знаний, выявления закономерностей, прогнозирования.

В 1989 году американским ученым в области информатики Григорием Пятецким-Шапиро было введено понятие интеллектуальный анализ данных (DM). Именно он описал какими должны быть новые полученные знания [1]:

1. новыми – целью любого анализа данных является получение ранее неизвестных знаний;
2. нетривиальными – это неожиданные, неочевидные, т.е. скрытые знания, которые не могут быть получены более простыми способами, например, путем использования средств визуализации;
3. практически полезными – приносить определенную выгоду при их получении или применении;
4. логически объяснимыми – полученные результаты должны быть понятны человеку, в противном случае, может иметь место фактор случайности.

Для получения нового знания в зависимости от целей анализа оператору данных необходимо решить одну или несколько задач DM:

- классификация – определение класса объекта по его характеристикам;
- регрессия – определение дискретных значений некоторых параметров объекта по его известным характеристикам;
- ассоциация – нахождение частных зависимостей между объектами;
- кластеризация – поиск характеристик, присущих независимым группам (кластерам), на всем множестве анализируемых данных;
- визуализация.

Постановка задач

1. Провести кластерный анализ данных с предварительной обработкой данных.

2. Изучить и применить методы по обработки выбросов в данных.

3. Изучить и применить методы по кодированию категориальных признаков.

4. Изучить и применить методы масштабирования данных для анализа.

1. Кластерный анализ данных

В статье [2] был рассмотрен основополагающий метод любого анализа данных – кластеризация, а также алгоритмы его реализации. Кластерный анализ, кластеризация, или естественная классификация, — это процедура или комплекс процедур, которые осуществляют разбиение исходного множества данных на подмножества, обладающие максимально схожими характеристиками и в то же время максимально отличными от других подмножеств. Задача кластеризации решается на начальных этапах исследования, ее решение помогает лучше понять данные и их природу[3,4]. Формальная постановка задачи кластеризации имеет следующий вид:

$X\{x_1, \dots, x_i, \dots x_j, \dots x_n\}$ – исходное множество объектов, характеризующихся некоторым набором атрибутов $X\{a_1, \dots, a_i, \dots a_j, \dots a_n\}$;

$\rho(x_i, x_j)$ – функция расстояния, характеризующая меру близости между объектами исходного множества;

$C\{c_1, \dots, c_i, \dots c_j, \dots c_n\}$ – искомое множество, являющееся совокупностью непересекающихся подмножеств (кластеров), состоящих из объектов множества X и являющихся близкими в соответствии с метрикой $\rho: \{x_i, x_j | x_i, x_j \in X \text{ и } \rho(x_i, x_j) < \sigma\}$, где σ – величина, определяющая меру близости.

Таким образом, выбор меры расстояния является важным этапом при кластерном анализе. Выделяют меры следующих типов:

- евклидовы основаны на местоположении точек в евклидовом пространстве;
- неевклидовы основаны на свойствах точек, но не на их положении в пространстве.

Рассмотрим поподробнее первый тип [4]. К нему относят такие меры как (формулы представлены в общем виде)

- Евклидово расстояние

$$\rho(x_i, x_j) = \sqrt{\sum_{m=1}^{n} (x_{im} - x_{jm})^2},$$

где n – размерность пространства;

• Квадрат евклидова расстояния. Позволяет задать большее расстояние отдаления между исследуемыми объектами:

$$\rho(x_i, x_j) = \sum_{m=1}^{n} (x_{im} - x_{jm})^2,$$

• Расстояние городских кварталов (Manhattan distances). В отличие от евклидова расстояния, для этой меры влияние отдельных больших разностей (выбросов) уменьшается:

$$\rho(x_i, x_j) = \sum_{m=1}^{n} |x_{im} - x_{jm}|.$$

• Расстояние Чебышёва. Позволяет задать такое расстояние, в соответствии с которым из исходной выборки будут определены максимально отличные друг от друга объекты:

$$\rho(x_i, x_j) = \max(|x_{im} - x_{jm}|).$$

Следующим этапом является определение мощности множества кластеров, то есть числа кластеров. В зависимости от целей и методов, применяемых при решении поставленной задачи, количество кластеров может быть либо не определено заранее, либо определено. Принято выделять три метода разбиения исходного множества на кластеры [5].

1. Иерархический метод: число кластеров заранее неизвестно. Предположения об их количестве делаются субъективно, на основе дендрограмм и/или динамики порога расщепления/слияния кластеров. Дендрограмма — это граф, отображающий связи между объектами исходного множества. В этой группе методов выделяют подвиды [6]:

– агломеративные методы: элементы исходного множества, которые сначала представляют собой отдельные кластеры, в дальнейшем объединяются в группы, тем самым уменьшая число кластеров до тех пор, пока не будет получен один единственный кластер;

– дивизимые методы: изначально все элементы принадлежат одному единственному кластеру, а с увеличением числа шагов разбиения количество кластеров увеличивается, то есть такие методы противоположны агломеративным.

2. Неиерархический: для достижения цели необходимо заранее определиться с результирующим количеством кластеров, а также с методами кластеризации.

3. Нечеткий метод: объекту исходной выборки не ставят в соответствие какой-либо конкретный кластер, а определяют степень принадлежности объекта к тому или иному кластеру.

Рассмотрим два алгоритма класетризации: Joining/tree clustering и K-mean clustering.

Joining/tree clustering (объединение/древовидная кластеризация) — иерархический агломеративный метод, результатом применения которого является дендрограмма, определяющая возможное количество кластеров. Помимо выбора типа меры расстояния между кластерами, метод предполагает и более тонкую настройку путем выбора правил объединения данных в кластеры. Ниже представлены некоторые из них.

1. Правило одиночной связи: расстояние между двумя кластерами определяется как расстояние между двумя наиболее близкими объектами в различных кластерах. Результирующие кластеры имеют тенденцию объединяться в цепочки.

2. Правило полных связей: расстояния между кластерами определяются наибольшим расстоянием между любыми двумя объектами.

3. Правило невзвешенного попарного среднего: расстояние между двумя кластерами определяется как среднее расстояние между всеми парами объектов в них.

4. Правило взвешенного попарного среднего. Метод идентичен предыдущему, за исключением того, что при вычислении размер соответствующих кластеров используется в качестве весового коэффициента.

Данный метод рекомендуется применять при первоначальной оценке данных, когда нет четкого представления о классах исходного множества. Однако важно понимать, что результат зависит от выбранной меры и правила объединения данных. При большом объеме данных, когда определение числа кластеров по дендрограмме может быть затруднено, следует воспользоваться описательными статистиками или графиком зависимости расстояния связи от числа шагов объединения.

K-mean clustering (алгоритм k-средних) — это неиерархический метод, предполагающий знание конечного числа групп. В результате исходное множество разбивается на k максимально удаленных друг от друга кластеров [7]. Данный алгоритм основан на минимизации среднеквадратичного отклонения точек кластеров от центров этих кластеров. В качестве настроек здесь выступают число кластеров, итераций, а также способ выбора центра кластеров.

Достоинствами алгоритма k-средних являются простота реализации и интуитивная понятность, недостатками — необходимость знать заранее число кластеров и зависимость результата от инициализации центров кластеров.

2. Данные с сайта HeadHunter

HeadHunter (hh.ru) — крупнейшая российская компания интернет-рекрутмента, развивающая бизнес в России, Белоруссии, Казахстане. Клиентами HeadHunter являются свыше 350 тыс. компаний[8]. Обширная база соискателей на hh.ru содержит более чем 50 млн резюме, а среднее дневное количество вакансий превышает 691 тыс. По данным SimilarWeb, HeadHunter занимает третье место в мире по популярности среди порталов по поиску работы и сотрудников.

У сайта hh.ru есть открытый API, с помощью которого можно очень детально анализировать рынок на больших объемах актуальных данных. Например, внешние разработчики могут получить все актуальные и архивные вакансии с зарплатами и остальными деталями. Через API работает поиск по вакансиям. Доступны различные справочники: регионы, используемые на сайте, специализации работников, отрасли компаний, станции метро и прочее. Для

авторизованных пользователей доступна работа с вашими вакансиями или резюме: в зависимости от того, работодатель вы или соискатель. Для работодателей есть поиск по вакансиям и возможность работы с ними. Всё взаимодействие происходит по протоколу HTTPS. Чтобы получить информацию — GET-запрос, удалить — DELETE, создать — POST, редактировать — PUT. Обмен данными производится в формате JSON [9].

В таблице 1 приведены основные данные, которые можно получить по каждой вакансии.

Таблица 1. Основные данные вакансий

Имя	Тип	Описание
id	string	Идентификатор вакансии
description	string	Описание вакансии, содержит html
key_skills	array	Информация о ключевых навыках, заявленных в вакансии. Список может быть пустым.
schedule	object	График работы. Элемент справочника schedule
experience	object	Требуемый опыт работы. Элемент справочника experience
address	object или null	Адрес вакансии
alternate_url	string	Ссылка на представление вакансии на сайте
employment	object или null	Тип занятости. Элемент справочника employment.
salary	object или null	Оклад
name	string	Название вакансии
area	object	Регион размещения вакансии
specializations[].profarea_name	string	Название профессиональной области, в которую входит специализация

3. Подготовка данных к анализу

Этап подготовки исходных данных к анализу является очень важным, поскольку результаты анализа целиком и полностью зависят от качества данных. На данном этапе работы с данными может возникнуть необходимость очистки данных различными способами, например [10]: путем изменения и/или дополнения записей в таблице БД, или путем удаления записей, и/или атрибутов из-за невозможности трактовки данных, что может привести к ошибочным или искаженным результатам.

Очевидно, что первоочередным из методов оценки данных является визуальный анализ, поскольку он наиболее естественен для оператора, работающего с данными. Но позднее, в зависимости от структуры и общей тематики данных, могут потребоваться более сложные подходы к решению данной проблемы.

4.1 Визуальный анализ данных

Первой, и наиболее заметной проблемой в полученных таблицах, является наличие строк с незаполненными полями в рамках одной записи. Эти записи являются неинформативными - удалим их.

Второй проблемой в данных является то, что минимальные значения зарплат у ряда полей являются не реалистичными. Удалим зарплаты, которые меньше прожиточного минимума утвержденного в России на 2021 год.

Следующей проблемой являются столбцы, дублирующие информацию, как следствие – неинформативные столбцы.

Так же в данных присутствуют два столбца характеризующие зарплаты от и до. Для упрощения анализа добавим столбец со средней зарплатой, в который запишем среднее значение зарплаты от и зарплаты до.

4.2 Обработка выбросов

Существует ГОСТ Р ИСО 16269-4-2017 Статистические методы. Статистическое представление данных. Часть 4. Выявление и обработка выбросов посвященный обработке выбросов. В соответствии с ГОСТ, выброс (outlier) - это

элемент маломощного подмножества выборки, существенно отличающийся от остальных элементов выборки.

Выбросы негативно влияют на результат анализа, поэтому их необходимо устранять. Выбросы появляются по следующим причинам. Ошибки в измерениях, например, часть значений признака "расстояние" была измерена не в километрах, а в метрах. Технические ошибки форматирования данных.

Основные задачи обработки выбросов это обнаружение выбросов и устранение (удаление или замена) их в зависимости от требований задачи.

По определению гистограммы распределения, выбросы - это значения на краях гистограммы (очень большие или очень маленькие по сравнению со всей выборкой). Задача обнаружения выбросов - это задача выделения элементов, находящихся на краях гистограммы.

Существует несколько подходов для определения выбросов. Подход в случае нормального распределения или распределения, похожего на нормальное.

Использование правила трех сигм.

Правило, утверждающее, что для любой случайной величины ξ с конечной дисперсией вероятность того, что случайная величина отклонится от своего математического ожидания вероятность того, что случайная величина отклонится от своего математического ожидания $M[\xi]$ не менее, чем на три среднеквадратических отклонения σ, не более $\frac{1}{9}$:

$$P(|\xi - M[\xi]| \geq 3\sigma) \leq \frac{1}{9}$$

Для большинства случайных величин эта вероятность меньше, например, для нормального распределения:

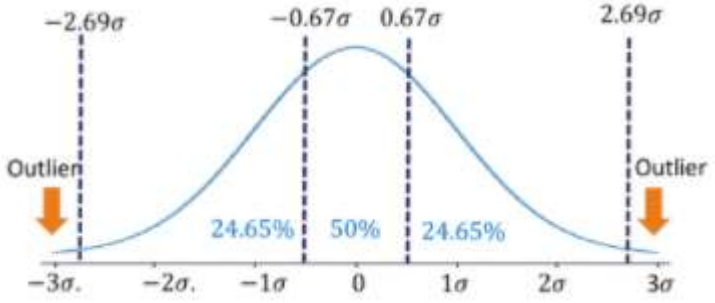


Рисунок 1 – Правило трех сигм

$$outlier < mean(x) - 3 * std(x)$$
$$outlier > mean(x) - 3 * std(x)$$

Использование 5% и 95% квантилей

95% данных располагаются выше 5% квантиля и 95% данных располагаются ниже 95% квантиля. Значения ниже 5% квантиля и выше 95% квантиля можно считать выбросами.

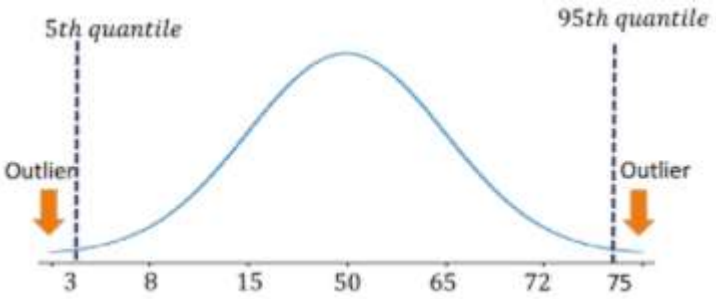


Рисунок 2 - Использование 5% и 95% квантилей

$$outlier < x.quantile(5\%)$$
$$outlier > x.quantile(95\%)$$

В нашем случае распределение данных зарплат асимметричное, в этом случае применяется подход использования межквартального размаха.

Межквартильный размах IQR (interquartile range, IQR) - это разность третьего квартиля и первого квартиля:

$$IQR = Q3(x) - Q1(x)$$

Тогда:

$$outlier < Q1(x) - K * IQR$$
$$outlier > Q3(x) + K * IQR$$

Значение K обычно выбирается равным 1,5.

После обнаружения всех выбросов удалим эти объекты из данных.

4.3 Переход от категориальных значений к количественным

На первый взгляд, решение данной проблемы может показаться достаточно тривиальным – заменить каждую категорию на некоторое числовое значение. Однако, сразу возникают закономерные вопросы: почему нумерация начата с 1 и почему именно эти числа присвоены этим категориальным значениям.

В связи с этим были рассмотрены другие часто используемые методы кодировки категориальных значений [11]:

- Label encoding – кодирование категорий целочисленными значениями. Недостатков является присвоение весов признакам (важность одних признаков по сравнению с другими), которые могут быть несравнимы друг с другом.
- Dummy или one-hot-кодирование – переход от символьного представления категориальных значений к бинарному представлению. Недостатком данного метода является избыточность и затрудненность дальнейшего анализа.
- Замена категориального признака количественным на основе некоторого логически связанного количественного признака. Недостатком данного метода является то, что может отсутствовать подходящий количественный параметр.
- Замена категории на число входящих в нее объектов. Недостатком данного метода является то, что новые значения параметров не стандартизированы.

В нашем случае будем использовать три метода кодирования категориальных признаков. Для кодирования городов будем использовать модификацию последнего метода, которая математически полностью обоснована и обладает полезным свойством нормализации всех значений выборки, что важно при осуществлении корреляционного анализа. Предложенный алгоритм перехода от категориальных параметров к количественным на основе частоты появления их значений приведен ниже.

На первом шаге выбирается параметр-столбец таблицы, для которого будет осуществлен переход. Вторым шагом является подсчет числа повторений в таблице уникальных значений столбца.

На третьем шаге необходимо рассчитать отношение числа повторений значения атрибута к общему числу строк:

$$w_{\text{знач.парам.}} = \frac{v_{\text{появл.знач.парам.}}}{S_{\text{чис.стр.табл.}}},$$

где $w_{\text{знач.парам.}}$ – вес значения параметра,

$v_{\text{появл.знач.парам.}}$ – частота появления значения параметра в таблице,

$S_{\text{чис.стр.табл.}}$ – суммарное число строк в таблице.

Таким образом, это позволяет определить важность (вес) значения параметра в контексте параметра. Заключительным шагом является замена категориального значения полученным количественным.

Важной особенностью данного метода является то, что после выполнения данной процедуры помимо того, что будет произведен непосредственный переход от категориальных параметров к количественным, все столбцы таблицы будут масштабированы по количеству строк в таблице в диапазоне от [0, 1].

Для признака, характеризующего требуемый опыт работы кандидата будем использовать кодирование категорий целочисленными значениями - label encoding, так как мы можем сравнивать значения этого признака.

Для остальных категориальных признаков применим one-hot-кодирование.

4.4 Масштабирование признаков

Масштабирование - это изменение диапазона измерения признака с целью улучшения качества построения модели.

Данные необходимо масштабировать потому**,** что признаки с меньшей амплитудой оказываются "оштрафованы" по сравнению с признаками с большей амплитудой, и оказывают меньшее влияние результаты анализа.

В кластерном анализе данных часто используется подсчет расстояния между кластерами с помощью метода ближайших соседей. Ключевым шагом этого метода является вычисление расстояний между соседями. Чаще всего в методе ближайших соседей используется Евклидово расстояние:

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_1)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}.$$

Если один из признаков имеет амплитуду значительно меньшую по сравнению с другими признаками, то этот признак почти не будет вносить вклад при вычислении расстояния.

В нашем случае значения всех признаков находятся в диапазоне от 0 до 1, а зарплаты указаны в десятках тысяч. Существует несколько подходов к масштабированию признаков. Рассмотрим наиболее часто применяемые.

Масштабирование данных на основе Z-оценки

$$x' = \frac{x - \mu(x)}{\sigma(\mathrm{x})},$$

где x – признак,

$\mu(x) = mean(x)$ – среднее значение,

$\sigma(\mathrm{x}) = std(x) -$ среднеквадратичное отклонение.

Особенности метода:

- Среднее значение приводится к 0.
- Среднеквадратичное отклонение приводится к 1.
- Форма исходного распределения сохраняется.
- Максимальные и минимальные значения могут варьироваться.
- Выбросы сохраняются.

MinMax-масштабирование

$$x' = \frac{x - min(x)}{\max(x) - \min(x)},$$

Особенности метода:

- Среднее значение может варьироваться.
- Среднеквадратичное отклонение может варьироваться.
- Форма исходного распределения может изменяться.
- Максимальные и минимальные значения в диапазоне [0;1].
- Выбросы сохраняются.

Масштабирование по медиане

$$x' = \frac{x - median(x)}{IQR},$$

где

$$IQR = Q3(x) - Q1(x)$$

IQR – разность между 1 и 3 квартилями.

Особенности метода:

- Медиана приводится к 0.
- Среднеквадратичное отклонение может варьироваться.
- Форма исходного распределения может изменяться.
- Максимальные и минимальные значения могут варьироваться.
- Устраняются выбросы.

Для упрощения последующей интерпретации результатов анализа будем использовать MinMax-масштабирование.

5. Эксперименты и интерпретация результатов анализа

В ходе работы была проведена кластеризация двумя методами: Joining/tree clustering и K-mean clusterin. В качестве расстояния между объектами будем использовать евклидово расстояние, а в качестве расстояния между кластерами: среднее невзвешенное расстояние, метод ближайшего соседа и метод Варда. В ходе экспериментов были получены следующие результаты.

Дендрограмма для метода среднее невзвешенное расстояние:

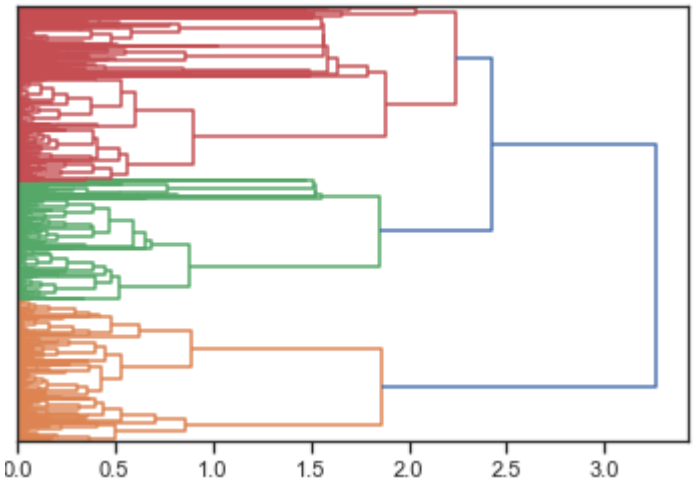


Рисунок 3 – Дендрограмма полученная в результате метода среднего невзвешенного расстояния

Дендрограмма для метода ближайшего соседа:

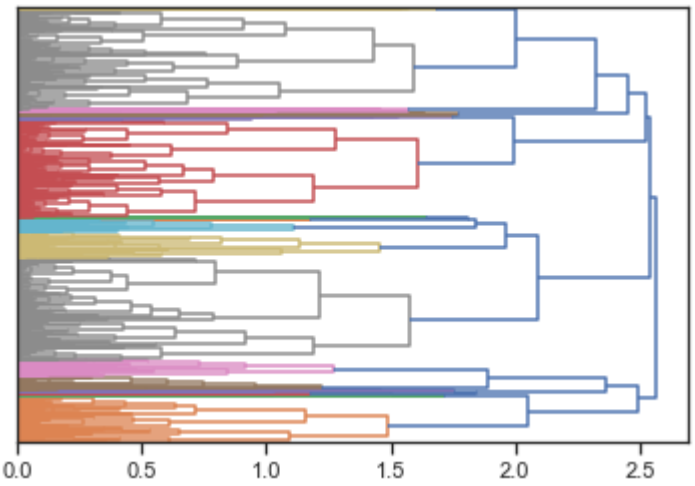


Рисунок 4 - Дендрограмма полученная в результате метода ближайшего соседа

Дендрограмма для метода Варда:

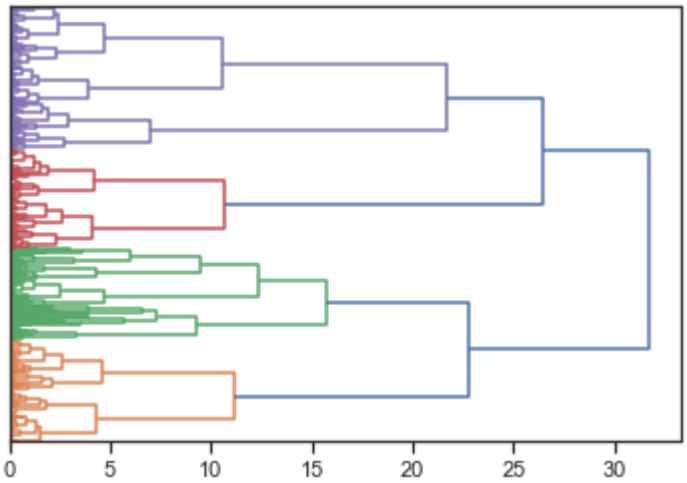


Рисунок 5 - Дендрограмма полученная в результате метода Варда

Видно, что лучше всего отработал метод Варда, продолжим анализ с ним. По дендрограмме определим количество кластеров равное пяти и посмотрим результаты.

cluster	area_name	prof_roles_name_Аналитик	prof_roles_name_Инженер-конструктор, инженер-проектировщик	prof_roles_name_Программист, разработчик	prof_roles_name_Руководитель группы разработки
1	0.592504	0.000000	0.000000	1.0	0.000000
2	0.675705	0.307692	0.053254	0.0	0.050296
3	0.614353	0.000000	0.000000	1.0	0.000000
4	0.545397	0.000000	0.000000	1.0	0.000000
5	0.510325	0.000000	0.000000	1.0	0.000000

prof_roles_name_Системный администратор	prof_roles_name_Системный инженер	prof_roles_name_Специалист по информационной безопасности	prof_roles_name_Специалист технической поддержки	prof_roles_name_Тестировщик
0.000000	0.000000	0.000000	0.000000	0.000000
0.127219	0.023669	0.044379	0.017751	0.363905
0.000000	0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000	0.000000

prof_roles_name_Учитель, преподаватель, педагог	roles	skill_C#	skill_C++	skill_Java	skill_Python	salary_minmax
0.000000	0.459354	0.000000	0.00000	0.000000	1.000000	0.338206
0.011834	0.411243	0.059172	0.06213	0.221893	0.656805	0.300689
0.000000	0.448847	0.000000	0.00000	1.000000	0.000000	0.399165
0.000000	0.410930	0.000000	1.00000	0.000000	0.000000	0.273758
0.000000	0.464192	1.000000	0.00000	0.000000	0.000000	0.302420

Рисунок 6 – Результаты кластерного анализа

Для проверки полученных результатов, проведем анализ методом K-Means и построим график каменистой осыпи.

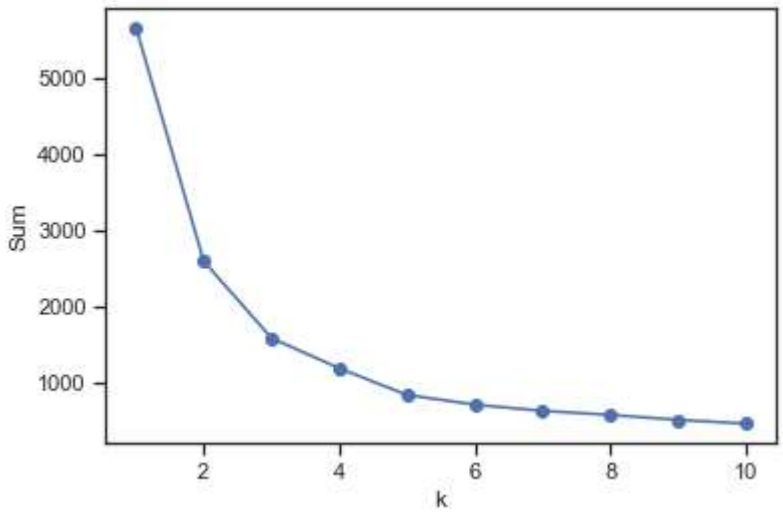


Рисунок 7 – График каменистой осыпи

Видно, что выбирать более 6 кластеров не имеет особого смысла.

Результаты, полученные в ходе данного метода, полностью совпадают с результатами метода иерархического кластерного анализа за исключением порядка кластеров, что говорит о их достоверности.

По полученным результатам можно сделать следующие выводы. Все данные разбились на пять кластеров. В 1, 3, 4 и 5 кластер вошли вакансии разработчиков по конкретным языкам программирования: Java, C#, Python и C++. Во второй кластер вошли все остальные вакансии с другими специализациями, это, например: аналитики, тестировщики, системные администраторы и т.д.

В столбце area_name значение характеризует величину города, чем больше город, тем больше значение. Видно, что все вакансии, связанные с

программированием, но с другой специализацией находятся в более крупных городах, чем вакансии программистов. Так же видно, что для таких вакансий наиболее востребованными являются языки программирования Python и Java, а языки C# и C++ практически ими не используются.

По столбцу salary_minmax можно сделать вывод, что программисты на Java имеют самые высокие зарплаты, далее идут Python и C#, а самые низкие зарплаты у программистов C++.

Столбец roles характеризует необходимый опыт работы, получается, чтобы устроиться работать программистом на C# необходимо больше всего опыта работы, а на Python и C++ меньше.

Вывод

В представленной научно исследовательской работе были рассмотрены вопросы, связанные с исследованием и анализом данных с популярного сайта вакансий с целью выявления полезных знаний. В рамках работы были решены следующие задачи. Проведен кластерный анализ данных двумя методами, исследованы и применены такие методы предварительной обработки данных как: методы по обработки выбросов в данных, методы по кодированию категориальных признаков и методы масштабирования данных для анализа.

Список использованных источников

1. Барсегян А. А. Анализ данных и процессов: учеб. пособие. / А. А. Барсегян. – СПб.: БХВ-Петербург, 2009. – 512 с.
2. Тихонов И. А. Обзор возможностей кластерного анализа данных в программном пакете STATISTICA / И. А. Тихонов // Политехнический молодежный журнал, – 2018. –№1– С. 1 – 10.
3. Дюран Б., Оделл П. Кластерный анализ. / Б. Дюран. – “Статистика”, 1977. – 128 с.
4. Айвазян А. А., Бухштабер В. М., Енюков И. С., Мешалкин Л. Д. Прикладная статистика: Классификации и снижение размерности. / А. А. Айвазян. – М.: Финансы и статистика, 1989. – 607 с.
5. Калинина В.Н., Соловьев В.И. Введение в многомерный статистический анализ. М.: ГУУ, 2003. 66 с.
6. Методы кластерного анализа. Иерархические методы. URL: http://www.intuit.ru/studies/courses/ 6/6/lecture/182?page=2 (дата обращения 19.12.2021).
7. Smola Alex, Vishwanathan S.V.N. Introduction to machine learning. Cambridge University Press, 2008. 234 p.
8. https://ru.wikipedia.org/wiki/HeadHunter (дата обращения 19.12.2021).
9. https://dev.hh.ru/(дата обращения 19.12.2021).
10. Сираева К. С., Катасёва Д. В., Катасёв А. С., Кирпичников А.П. Методика очистки персональных данных в информационных системах организаций. / К. С. Сираева // Вестник Технологического университета, 2018. – Т. 21, вып. 2. – С. 104 – 108.
11. Дьяконов А.Г. Методы решения задач классификации с категориальными признаками. Прикладная математика и информатика. Труды факультета Вычислительной математики и кибернетики МГУ имени М.В. Ломоносова. 2014. № 46. С. 103 – 127.

Приложение А

```
# Получение DataFrame из БД
engine = create_engine('postgresql://postgres:0000@localhost:5432/postgres')
```

```
df = pd.read_sql('select * from vacancies', con=engine)
```

```
df_skill = pd.read_sql('select * from skills', con=engine)
```

```
# Сохранить df в файл
df.to_pickle("./StartDataFrame")
```

```
# Сохранить df в файл
df_skill.to_pickle("./StartDataFrameSkill")
```

```
# Прочитать df из файла
start_df = pd.read_pickle("./StartDataFrame")
df = pd.read_pickle("./StartDataFrame")
```

```
# Прочитать df из файла
df_skill = pd.read_pickle("./StartDataFrameSkill")
```

```
df.head(1)
```

	id	name	address	area_name	salary_from	salary_to	prof_roles_id	prof_roles_name	spec_id	spec_name	spec_profarea_id
0	16142474	Специалист по сопровождению ПО (техническая по...	None	Ижевск	20000.0	70000.0	121	Специалист технической поддержки	1.172	Начальный уровень, Мало опыта	1

spec_profarea_name	experience_id	employment_id	schedule_id	url	salary_currency	description
Информационные технологии, интернет, телеком	between1And3	full	fullDay	https://hh.ru/vacancy/16142474	RUR	<p>Центр Информационных Технологий БАРС — совр...

```
df_skill.head()
```

	id_vacancy	skill
0	23863827	C#
1	23863827	COM
2	23863827	COM+
3	23863827	XML
4	23863827	XSL

```
# Удаление лишних колонок

df.drop(["address"], axis = 'columns', inplace = True)
df.drop(["prof_roles_id"], axis = 'columns', inplace = True)
df.drop(["spec_id"], axis = 'columns', inplace = True)
df.drop(["spec_profarea_id"], axis = 'columns', inplace = True)
df.drop(["url"], axis = 'columns', inplace = True)
df.drop(["description"], axis = 'columns', inplace = True)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7841 entries, 0 to 7840
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   id                  7841 non-null   int64  
 1   name                7841 non-null   object 
 2   area_name           7841 non-null   object 
 3   salary_from         6792 non-null   float64
 4   salary_to           4553 non-null   float64
 5   prof_roles_name     7841 non-null   object 
 6   spec_name           7841 non-null   object 
 7   spec_profarea_name  7841 non-null   object 
 8   experience_id       7841 non-null   object 
 9   employment_id       7841 non-null   object 
 10  schedule_id         7841 non-null   object 
 11  salary_currency     7841 non-null   object 
dtypes: float64(2), int64(1), object(9)
memory usage: 735.2+ KB
```

```
# Удалим строки с null

df.dropna(inplace=True)
```

```
#
df["salary_currency"].value_counts()
```

```
RUR    3263
USD     197
EUR      43
UAH       1
Name: salary_currency, dtype: int64
```

```
# Удаление зарпалат не в рублях
df = df.query("salary_currency not in ['USD', 'EUR', 'UAH']")
df["salary_currency"].value_counts()
```

```
RUR    3263
Name: salary_currency, dtype: int64
```

```
df.drop(["salary_currency"], axis = 'columns', inplace = True)
```

```
df
```

	id	name	area_name	salary_from	salary_to	prof_roles_name	spec_name	spec_profarea_name	experience_id	employment_id
0	16142474	Специалист по сопровождению ПО (техническая по...	Ижевск	20000.0	70000.0	Специалист технической поддержки	Начальный уровень, Мало опыта	Информационные технологии, интернет, телеком	between1And3	full
1	20120535	Ведущий программист Scala / Java (Senior Scal...	Тольятти	130000.0	275000.0	Программист, разработчик	Программирование, Разработка	Информационные технологии, интернет, телеком	moreThan6	full

```
# Добавим столбец со средней зарплатой

df['salary'] = (df['salary_from'] + df['salary_to']) / 2
```

```
# spec_name - описывает область занятости, prof_roles_name - профессию человека, spec_profarea_name
# - не дает конкретной информации, поэтому удалим

df.drop(["spec_profarea_name"], axis = 'columns', inplace = True)
```

```
# Добавим skill для каждого человека

df_skill.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45013 entries, 0 to 45012
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   id_vacancy  45013 non-null  int64 
 1   skill       45013 non-null  object
dtypes: int64(1), object(1)
memory usage: 703.5+ KB
```

```
df_skill = df_skill.rename(columns={'id_vacancy': 'id'})
```

```
df_skill
```

	id	skill
0	23863827	C#
1	23863827	COM

```
# Объеденим таблицы вакансий и навыков

df_new = pd.merge(df, df_skill, on='id', how='left')
```

```
# Удаление всех навыков кроме языков программирования
df_new = df_new.query("skill in ['Java', 'C++', 'Python', 'C#', 'java', 'python', 'c#', 'c++']")
```

```
df_new["skill"].value_counts()
```

```
Python    746
Java      575
C#        528
C++       272
Name: skill, dtype: int64
```

```
# Удаление лишних колонок

df_new.drop(["id"], axis = 'columns', inplace = True)
df_new.drop(["name"], axis = 'columns', inplace = True)
df_new.drop(["salary_from"], axis = 'columns', inplace = True)
df_new.drop(["salary_to"], axis = 'columns', inplace = True)
df_new.drop(["spec_name"], axis = 'columns', inplace = True)
```

```
df_new
```

	area_name	prof_roles_name	experience_id	employment_id	schedule_id	salary	skill
11	Тула	Программист, разработчик	between3And6	full	fullDay	170000.0	Java
20	Санкт-Петербург	Системный администратор	between3And6	full	remote	125000.0	Python
21	Санкт-Петербург	Программист, разработчик	between1And3	full	fullDay	150000.0	C++
24	Ульяновск	Программист, разработчик	between3And6	full	flexible	140000.0	Java
30	Ульяновск	Программист, разработчик	between3And6	full	flexible	140000.0	Python

```
# Кодирование категориальных признаков

min_city = 14
max_city = 793
```

```
df.loc[(df.area_name == 'Москва'), 'area_name'] = 1
```

```
df.loc[(df.area_name == 'Санкт-Петербург'), 'area_name'] = 342/max_city
```

```
df.loc[(df.area_name == 'Новосибирск'), 'area_name'] = 131/max_city
```

```
df.loc[(df.area_name == 'Екатеринбург'), 'area_name'] = 71/max_city
```

```
df.loc[(df.area_name == 'Ростов-на-Дону'), 'area_name'] = 41/max_city
```

```
df.loc[(df.area_name == 'Краснодар'), 'area_name'] = 37/max_city
```

```
df.loc[(df.area_name == 'Нижний Новгород'), 'area_name'] = 37/max_city
```

```
df.loc[(df.area_name == 'Самара'), 'area_name'] = 30/max_city
```

```
df.loc[(df.area_name == 'Уфа'), 'area_name'] = 26/max_city
```

```
df.loc[(df.area_name == 'Красноярск'), 'area_name'] = 25/max_city
```

```
df.loc[(df.area_name == 'Калининград'), 'area_name'] = 23/max_city
```

```
df.loc[(df.area_name == 'Воронеж'), 'area_name'] = 22/max_city
```

```
df.loc[(df.area_name == 'Челябинск'), 'area_name'] = 22/max_city
```

```
df.loc[(df.area_name == 'Казань'), 'area_name'] = 57/max_city
```

```
df['area_name'].count()
```

```
2121
```

```
df["area_name"] = pd.to_numeric(df["area_name"], errors='coerce')
```

```
df = df[df["area_name"].notnull()]
```

```
df
```

	area_name	prof_roles_name	experience_id	employment_id	schedule_id	salary	skill
20	0.431274	Системный администратор	between3And6	full	remote	125000.0	Python
21	0.431274	Программист, разработчик	between1And3	full	fullDay	150000.0	C++
42	1.000000	Системный администратор	between3And6	full	remote	125000.0	Python
63	0.032787	Программист, разработчик	between1And3	full	fullDay	105000.0	Java
71	1.000000	Программист, разработчик	between3And6	full	remote	175000.0	Python
...	...	...	...	...	...	...	...
20593	1.000000	Тестировщик	between1And3	full	remote	200000.0	Java
20602	1.000000	Программист, разработчик	between1And3	full	remote	185000.0	C#
20624	0.089533	Программист, разработчик	moreThan6	full	fullDay	120000.0	C++
20649	1.000000	Программист, разработчик	between3And6	full	flexible	275000.0	Python
20650	1.000000	Тестировщик	noExperience	full	fullDay	275000.0	Python

```
prof_roles = ['Программист, разработчик','Тестировщик',
'Аналитик',
'Системный администратор',
'Инженер-конструктор, инженер-проектировщик',
'Руководитель группы разработки',
'Специалист по информационной безопасности',
'Системный инженер',
'Специалист технической поддержки',
'Учитель, преподаватель, педагог']
```

```
df = df.query("prof_roles_name in {0}".format(prof_roles))
```

```
df_roles = pd.get_dummies(df[['prof_roles_name']])
```

```
df = pd.concat([df, df_roles], axis=1, sort = False)
```

```
df['experience_id'].value_counts()[0:50]
```

```
between1And3    843
between3And6    631
noExperience    179
moreThan6        64
Name: experience_id, dtype: int64
```

```
L = []
for i in df.index:

    #print(df['experience_id'][[i]].values)
    L.append(df['experience_id'][[i]].values)
```

```
roles = []
for i in range(len(L)):
    # print(L[i][0])
    roles.append(L[i][0])
```

```
codding_roles = []
for i in roles:
    if (i == 'between3And6'):
        codding_roles.append(0.66)
    if (i == 'between1And3'):
        codding_roles.append(0.33)
    if (i == 'noExperience'):
        codding_roles.append(0)
    if (i == 'moreThan6'):
        codding_roles.append(1)
```

```
df['employment_id'].value_counts()[0:50]
```

```
full         1649
part           43
probation      17
project         8
Name: employment_id, dtype: int64
```

```
df_empl = pd.get_dummies(df[['employment_id']])
```

```
df = pd.concat([df, df_empl], axis=1, sort = False)
```

```
df.drop(["employment_id"], axis = 'columns', inplace = True)
```

```
df['schedule_id'].value_counts()[0:50]
```

```
fullDay     930
remote      632
flexible    151
shift         4
Name: schedule_id, dtype: int64
```

```
df_sched = pd.get_dummies(df[['schedule_id']])
```

```
df_sched
```

	schedule_id_flexible	schedule_id_fullDay	schedule_id_remote	schedule_id_shift
20	0	0	1	0
21	0	1	0	0
42	0	0	1	0
63	0	1	0	0
71	0	0	1	0

```
df = pd.concat([df, df_sched], axis=1, sort = False)
```

```
df.drop(["schedule_id"], axis = 'columns', inplace = True)
```

```
df['skill'].value_counts()[0:50]
```

```
Python    610
Java      450
C#        386
C++       194
Name: skill, dtype: int64
```

```
df_skill = pd.get_dummies(df[['skill']])
```

```
df_skill
```

	skill_C#	skill_C++	skill_Java	skill_Python
20	0	0	0	1
21	0	1	0	0
42	0	0	0	1
63	0	0	1	0
71	0	0	0	1

```
df = pd.concat([df, df_skill], axis=1, sort = False)
```

```
df.drop(["skill"], axis = 'columns', inplace = True)
```

```
# Сохранить df в файл
df.to_pickle("./CodingDataFramev3")
```

```
df = pd.read_pickle("./CodingDataFramev3")
```

```
# Обработка выбросов
df.describe()
```

	area_name	salary	prof_roles_name_Аналитик	prof_roles_name_Инженер-конструктор, инженер-проектировщик	prof_roles_name_Программист, разработчик	prof_roles_name_Руководитель группы разработки	prof_
count	1640.000000	1.640000e+03	1640.000000	1640.000000	1640.000000	1640.000000	
mean	0.591777	1.711926e+05	0.063415	0.010976	0.793902	0.010366	
std	0.410142	9.902503e+04	0.243782	0.104220	0.404625	0.101315	
min	0.027743	8.000000e+01	0.000000	0.000000	0.000000	0.000000	
25%	0.165195	1.100000e+05	0.000000	0.000000	1.000000	0.000000	
50%	0.431274	1.600000e+05	0.000000	0.000000	1.000000	0.000000	
75%	1.000000	2.250000e+05	0.000000	0.000000	1.000000	0.000000	
max	1.000000	2.350000e+06	1.000000	1.000000	1.000000	1.000000	

Видно, что минимальная зарплата - 80р - некорректные данные. Посмотрим детальнее.

```
df.nsmallest(10,'salary')
```

	area_name	salary	prof_roles_name_Аналитик	prof_roles_name_Инженер-конструктор, инженер-проектировщик	prof_roles_name_Программист, разработчик	prof_roles_name_Руководитель группы разработки	prof_roles_
18885	0.431274	80.0	0	0	1	0	
17943	1.000000	265.0	0	0	1	0	
13554	1.000000	300.0	0	0	1	0	
11535	0.032787	10000.0	0	0	1	0	
13428	1.000000	15000.0	0	0	1	0	
6742	0.089533	20000.0	0	0	1	0	
20561	0.071879	20500.0	0	0	1	0	
10946	0.431274	30000.0	0	0	1	0	
12512	0.071879	30000.0	0	0	1	0	
10738	0.032787	32500.0	0	0	1	0	

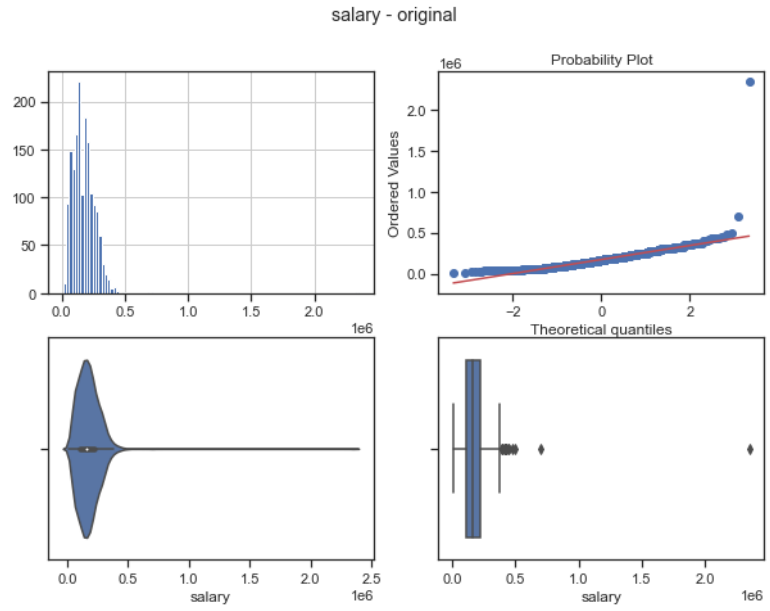
```
df = df.query("salary > 9000")
```

```
df.nsmallest(10,'salary')
```

	area_name	salary	prof_roles_name_Аналитик	prof_roles_name_Инженер-конструктор, инженер-проектировщик	prof_roles_name_Программист, разработчик	prof_roles_name_Руководитель группы разработки	prof_roles_
11535	0.032787	10000.0	0	0	1	0	
13428	1.000000	15000.0	0	0	1	0	
6742	0.089533	20000.0	0	0	1	0	
20561	0.071879	20500.0	0	0	1	0	
10946	0.431274	30000.0	0	0	1	0	
12512	0.071879	30000.0	0	0	1	0	
10738	0.032787	32500.0	0	0	1	0	
15344	0.431274	32500.0	0	0	1	0	
16792	0.165195	32500.0	0	0	1	0	
20352	0.431274	32500.0	0	0	1	0	

```
def diagnostic_plots(df, variable, title):
    fig, ax = plt.subplots(figsize=(10,7))
    # гистограмма
    plt.subplot(2, 2, 1)
    df[variable].hist(bins=100)
    ## Q-Q plot
    plt.subplot(2, 2, 2)
    stats.probplot(df[variable], dist="norm", plot=plt)
    # ящик с усами
    plt.subplot(2, 2, 3)
    sns.violinplot(x=df[variable])
    # ящик с усами
    plt.subplot(2, 2, 4)
    sns.boxplot(x=df[variable])
    fig.suptitle(title)
    plt.show()
```

```
diagnostic_plots(df, 'salary', 'salary - original')
```

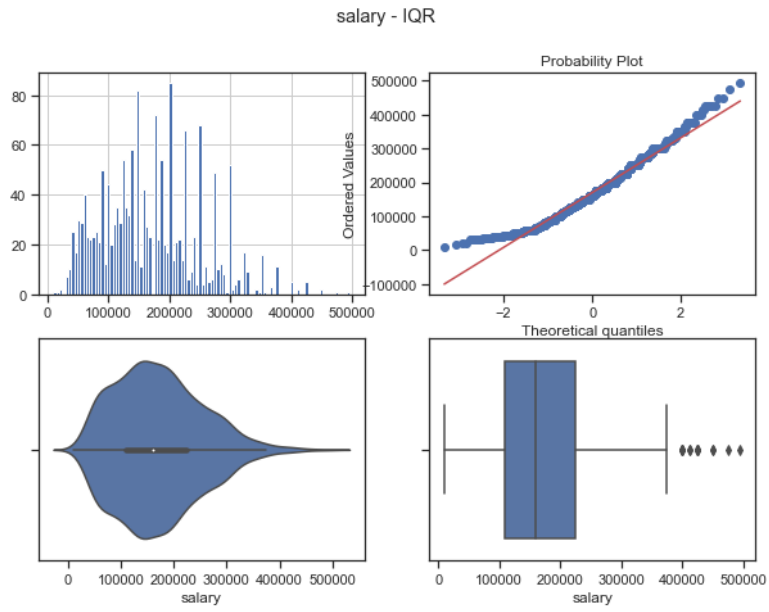



```
# Функция вычисления верхней и нижней границы выбросов
def get_outlier_boundaries_IQR(df, col):
    K2 = 3
    IQR = df[col].quantile(0.75) - df[col].quantile(0.25)
    lower_boundary = df[col].quantile(0.25) - (K2 * IQR)
    upper_boundary = df[col].quantile(0.75) + (K2 * IQR)
    return lower_boundary, upper_boundary
```

```
# удаление выбросов
col = 'salary'

# Вычисление верхней и нижней границы
lower_boundary, upper_boundary = get_outlier_boundaries_IQR(df, col)
# Флаги для удаления выбросов
outliers_temp = np.where(df[col] > upper_boundary, True,
                         np.where(df[col] < lower_boundary, True, False))
# Удаление данных на основе флага
data_trimmed = df.loc[~(outliers_temp), ]
title = 'Поле-{}, метод-IQR, строк-{}'.format(col, data_trimmed.shape[0])
df_salary_IQR = data_trimmed
```

```
diagnostic_plots(df_salary_IQR, 'salary', 'salary - IQR')
```




```
df_salary_IQR.describe()
```

	area_name	salary	prof_roles_name_Аналитик	prof_roles_name_Инженер-конструктор, инженер-проектировщик	prof_roles_name_Программист, разработчик	prof_roles_name_Руководитель группы разработки	pro
count	1635.000000	1635.000000	1635.000000	1635.000000	1635.000000	1635.000000	
mean	0.591387	169850.266972	0.063609	0.011009	0.793272	0.010398	
std	0.410241	81878.482149	0.244129	0.104377	0.405082	0.101468	
min	0.027743	10000.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.165195	110000.000000	0.000000	0.000000	1.000000	0.000000	
50%	0.431274	160000.000000	0.000000	0.000000	1.000000	0.000000	
75%	1.000000	225000.000000	0.000000	0.000000	1.000000	0.000000	
max	1.000000	495000.000000	1.000000	1.000000	1.000000	1.000000	

```
df = df_salary_IQR
```

```
# MinMax масштабирование

df['salary_minmax'] = (df['salary'] - df['salary'].min())/(df['salary'].max()-df['salary'].min())
```

```
df
```

	area_name	salary	prof_roles_name_Аналитик	prof_roles_name_Инженер-конструктор, инженер-проектировщик	prof_roles_name_Программист, разработчик	prof_roles_name_Руководитель группы разработки	prof_roles_
20	0.431274	125000.0	0	0	0	0	
21	0.431274	150000.0	0	0	1	0	
42	1.000000	125000.0	0	0	0	0	
63	0.032787	105000.0	0	0	1	0	
71	1.000000	175000.0	0	0	1	0	

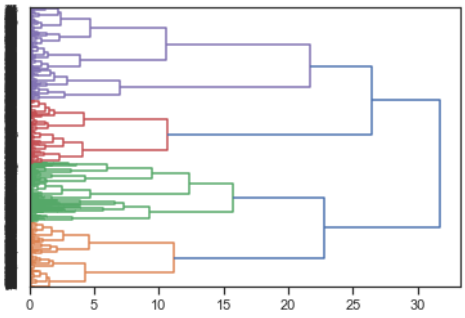
```
df.drop(["salary"], axis = 'columns', inplace = True)
```

```
# Иерархический кластерный анализ

from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
```

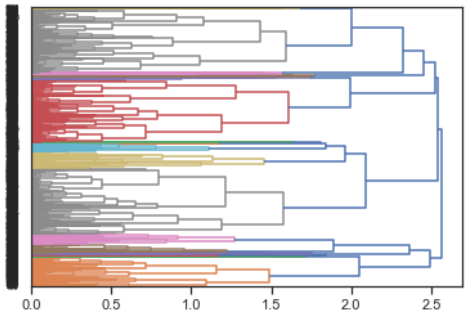
```
# метод варда
link = linkage(df, 'ward', 'euclidean')
```

```
dn = dendrogram(link, orientation='right')
```



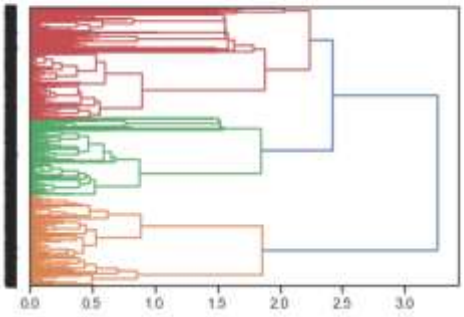

```
# метод ближайшего соседа
link = linkage(df, 'complete', 'euclidean')
```

```
dn = dendrogram(link, orientation='right')
```




```
#метод среднее невзвешенное растояние
link = linkage(df, 'average', 'euclidean')
```

```
dn = dendrogram(link, orientation='right')
```



```
# метод Варда показал лучшую дендрограмму, применим его

df['cluster'] = fcluster(link, 20, criterion='distance')
```

```
df.groupby('cluster').size()
```

```
cluster
1    387
2    338
3    373
4    172
5    365
dtype: int64
```

```
df.groupby('cluster').mean()
```

cluster	area_name	prof_roles_name_Аналитик	prof_roles_name_Инженер-конструктор, инженер-проектировщик	prof_roles_name_Программист, разработчик	prof_roles_name_Руководитель группы разработки
1	0.592504	0.000000	0.000000	1.0	0.000000
2	0.675705	0.307692	0.053254	0.0	0.050296
3	0.614353	0.000000	0.000000	1.0	0.000000
4	0.545397	0.000000	0.000000	1.0	0.000000
5	0.510325	0.000000	0.000000	1.0	0.000000

prof_roles_name_Системный администратор	prof_roles_name_Системный инженер	prof_roles_name_Специалист по информационной безопасности	prof_roles_name_Специалист технической поддержки	prof_roles_name_Тестировщик
0.000000	0.000000	0.000000	0.000000	0.000000
0.127219	0.023669	0.044379	0.017751	0.363905
0.000000	0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000	0.000000

prof_roles_name_Учитель, преподаватель, педагог	roles	skill_C#	skill_C++	skill_Java	skill_Python	salary_minmax
0.000000	0.459354	0.000000	0.00000	0.000000	1.000000	0.338206
0.011834	0.411243	0.059172	0.06213	0.221893	0.656805	0.300689
0.000000	0.448847	0.000000	0.00000	1.000000	0.000000	0.399165
0.000000	0.410930	0.000000	1.00000	0.000000	0.000000	0.273758
0.000000	0.464192	1.000000	0.00000	0.000000	0.000000	0.302420

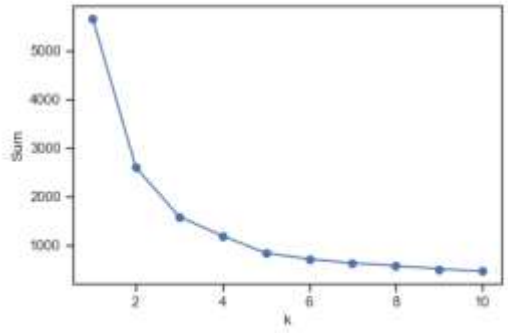
```
# Кластерный анализ методом K-means

from sklearn.cluster import KMeans

model = KMeans(n_clusters=5,random_state=42)
```

```
K = range(1, 11)
models = [KMeans(n_clusters=k, random_state=42).fit(df) for k in K]
dist = [model.inertia_ for model in models]

plt.plot(K, dist, marker='o')
plt.xlabel('k')
plt.ylabel('Sum')
plt.show()
```




```
from sklearn import mixture
model = mixture.GaussianMixture(n_components=50, max_iter=10000)
#Задание параметров
model.fit(df)
#Нахождение кластеров
print(model.means_)
#Распечатка центров полученных кластеров
```

```
model = KMeans(n_clusters=5, random_state=42)
model.fit(df)
df['cluster']=model.labels_
df.groupby('cluster').mean()
```

cluster	area_name	prof_roles_name_Аналитик	prof_roles_name_Инженер-конструктор, инженер-проектировщик	prof_roles_name_Программист, разработчик	prof_roles_name_Руководитель группы разработки
0	0.675705	0.307692	0.053254	0.0	0.050296
1	0.545397	0.000000	0.000000	1.0	0.000000
2	0.592504	0.000000	0.000000	1.0	0.000000
3	0.614353	0.000000	0.000000	1.0	0.000000
4	0.510325	0.000000	0.000000	1.0	0.000000

prof_roles_name_Системный администратор	prof_roles_name_Системный инженер	prof_roles_name_Специалист по информационной безопасности	prof_roles_name_Специалист технической поддержки	prof_roles_name_Тестировщик
0.127219	0.023669	0.044379	0.017751	0.363905
0.000000	0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000	0.000000

prof_roles_name_Учитель, преподаватель, педагог	roles	skill_C#	skill_C++	skill_Java	skill_Python	salary_minmax
0.011834	0.411243	0.059172	0.06213	0.221893	0.656805	0.300689
0.000000	0.410930	0.000000	1.00000	0.000000	0.000000	0.273758
0.000000	0.459354	0.000000	0.00000	0.000000	1.000000	0.338206
0.000000	0.448847	0.000000	0.00000	1.000000	0.000000	0.399165
0.000000	0.464192	1.000000	0.00000	0.000000	0.000000	0.302420