



МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА - Российский технологический университет»
РТУ МИРЭА
Институт искусственного интеллекта
Кафедра высшей математики

ОТЧЁТ ПО НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ
(получение первичных навыков научно-исследовательской работы)

Тема НИР: Анализ набора данных быстрых свиданий «Speed Dating Experiment»
(kaggle.com)
приказ университета о направлении на НИР
от «9» февраля 2023 г. № 735 - С

Отчет представлен к
рассмотрению:
Студент группы КМБО-03-
22

Лыков Д.С.
(расшифровка подписи)
«31» мар 2023г.

Отчет утвержден.
Допущен к защите:

Руководитель НИР от
кафедры

Петруевич Д.А.
(расшифровка подписи)
«8» апр 2023г.



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА - Российский технологический университет»

РТУ МИРЭА

ЗАДАНИЕ

на НАУЧНО-ИССЛЕДОВАТЕЛЬСКУЮ РАБОТУ

(получение первичных навыков научно-исследовательской работы)

Студенту 1 курса учебной группы КМБО-03-22 института искусственного
интеллекта Лыкову Даниилу Сергеевичу

(фамилия, имя и отчество)

Место и время НИР: Институт искусственного интеллекта, кафедра высшей математики

Время НИР: с «09» февраля 2023 по «31» мая 2023

Должность на НИР: практикант

1. ЦЕЛЕВАЯ УСТАНОВКА: изучение основ анализа данных и машинного обучения

2. СОДЕРЖАНИЕ НИР:

2.1 Изучить: литературу и практические примеры по темам: 1) построение линейной регрессии, 2) использование метода главных компонент, 3) поиск и устранение линейной зависимости в данных, 4) основы нормализации данных, 5) методы классификации и кластеризации («решающее дерево», «случайный лес», «к ближайших соседей»).

2.2 Практически выполнить: 1) снижение размерности исходных задач при помощи метода главных компонент при возможности; построение линейной регрессии для некоторого параметра, исключение регрессоров, не коррелирующих с объясняемой переменной; решение задачи классификации или кластеризации на основе открытого набора данных с ресурса kaggle.com

2.3 Ознакомиться: с применением метода главных компонент; методов классификации («решающего дерева», «случайного леса»); методов кластеризации («к ближайших соседей»); построением модели линейной регрессии.

3. ДОПОЛНИТЕЛЬНОЕ ЗАДАНИЕ: анализ набора данных быстрых свиданий «Speed Dating Experiment» (kaggle.com)

4. ОГРАНИЦИОННО-МЕТОДИЧЕСКИЕ УКАЗАНИЯ: определить, каковы наименее желательные качества в мужчине? Отличается ли это для женщин? Являются ли общие интересы более важными, чем общее расовое происхождение? Произвести кластеризацию, выявить аномальные случаи. Что в них особенного?

Заведующий кафедрой
высшей математики

«09» февраля 2023 г.

СОГЛАСОВАНО

Руководитель НИР от кафедры:

«09» февраля 2023 г.

Задание получил:
«09» февраля 2023 г.

(подпись)

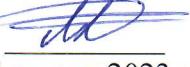
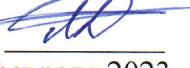
(подпись)

Ю.И.Худак

(Петрусевич Д.А.)
(фамилия и инициалы)

(Лыков Д.С.)
(фамилия и инициалы)

ИНСТРУКТАЖ ПРОВЕДЕН:

Вид мероприятия	ФИО ответственного, подпись, дата	ФИО студента, подпись, дата
Охрана труда	Петрусевич Д.А.  «09» февраля 2023 г.	Лыков Д.С.  «09» февраля 2023 г.
Техника безопасности	Петрусевич Д.А.  «09» февраля 2023 г.	Лыков Д.С.  «09» февраля 2023 г.
Пожарная безопасность	Петрусевич Д.А.  «09» февраля 2023 г.	Лыков Д.С.  «09» февраля 2023 г.
Правила внутреннего распорядка	Петрусевич Д.А.  «09» февраля 2023 г.	Лыков Д.С.  «09» февраля 2023 г.



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА - Российский технологический университет»
РТУ МИРЭА

**РАБОЧИЙ ГРАФИК ПРОВЕДЕНИЯ
НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЫ**
(получение первичных навыков научно-исследовательской работы)

студента Лыкова Д.С. 1 курса группы КМБО-03-22 очной формы обучения, обучающегося по направлению подготовки 01.03.02 «Прикладная математика и информатика», профиль «Математическое моделирование и вычислительная математика»

Неделя	Сроки выполнения	Этап	Отметка о выполнении
1	09.02.2023	Выбор темы НИР. Пройти инструктаж по технике безопасности	✓
1	09.02.2023	Вводная установочная лекция	✓
2	18.02.2023	Построение и оценка парной регрессии с помощью языка R	✓
3	25.02.2023	Построение и оценка множественной регрессии с помощью языка R	✓
4	04.03.2023	Построение доверительных интервалов. Обработка факторных переменных. Мультиколлинеарность	✓
5	11.03.2023	Гетероскедастичность	✓
6	18.03.2023	Классификация	✓
7	25.03.2023	Кластеризация. Предобработка данных	✓
8	01.04.2023	Метод главных компонент	✓
9	08.04.2023	Ансамбли классификаторов.	

		Беггинг. Бустинг	
16	27.05.2023	Представление отчётных материалов по НИР и их защита. Передача обобщённых материалов на кафедру для архивного хранения	✓
		Зачётная аттестация	✓

Согласовано:

Заведующий кафедрой

/ ФИО / Худак Ю.И.

Руководитель НИР от
кафедры

/ ФИО / Петrusевич Д.А.

Обучающийся

/ ФИО / Лыков Д.С.

Оглавление

Задачи.....	3
Задача №1	3
Задача №2	6
Задача №3	15
Задача №4	24
Задача №5	31
Задача №6	42
Заключение.....	56
Список литературы.....	58
Приложения	59
Приложение 1	59
Приложение 2	62
Приложение 3	69
Приложение 4	84
Приложение 5	89
Приложение 6	93

Задачи

Задача №1

Необходимо загрузить данные из указанного набора и произвести следующие действия.

Набор данных: *Swiss*.

Объясняемая переменная: *Education*.

Регрессоры: *Agriculture, Fertility*.

1. Оцените среднее значение, дисперсию и СКО переменных, указанных во втором и третьем столбце.

Чтобы высчитать среднее значение, дисперсию и СКО объясняемой переменной и регрессоров воспользуемся уже готовыми командами в R.

Для того, чтобы высчитать среднее значение воспользуемся командой *mean*.

- Среднее значение столбца *Education* равно 10.97872, значит образованных призывников после начальной школы в то время было мало.
- Среднее значение столбца регрессора *Fertility* равно 70.14255, значит рождаемость в браке тогда была на довольно высоком уровне.
- Среднее значение столбца *Agriculture* равно 50.65957, значит мужчин, занимающихся сельским хозяйством, было тогда почти половина.

Для того, чтобы высчитать дисперсию воспользуемся командой *var*.

Для того, чтобы высчитать СКО воспользуемся определением. СКО — это корень из дисперсии. В языке R существует команда *sqrt*, с помощью которой можно найти СКО. СКО равен *sqrt(var)*.

2. Постройте зависимости вида $y = a + bx$, где y – объясняемая переменная, x – регрессор.

Построим модель вида $y = a + bx$, с помощью команды *lm* пакета *lmtest* в R, где *Education* (объясняемая переменная), *Agriculture* (регрессор). С помощью команды *summary*, встроенной в R, найдем коэффициенты a , b (в таблице 1.1 они находятся в столбце *Estimate*: a – свободный коэффициент (*Intercept*), b – коэффициент *Agriculture*).

Таким образом, в модели, где *Education* выступает в роли объясняемой переменной, а *Agriculture* в роли регрессора значения коэффициентов будут равны: $a = 24.6952$, $b = -0,2707$.

Модель вида $y = a + bx$, где *Education* – объясняемая переменная, а *Agriculture* - регрессор (подробные характеристики данной модели представлены в таблице 1.1):

$$y = 24.6952 + -0,2707 * x$$

Таблица 1.1. Характеристики модели зависимости параметра *Education* от параметра *Agriculture* в наборе данных *Swiss*.

Коэффициенты / параметры	Estimate	Std. Error	t value	Pr (> t)	Уровень значимости
(Intercept)	24.6952	2.68890	9.184	6.98e-12	***
Agriculture	-0.2707	0.04852	-5.580	1.30e-06	***

Теперь построим модель вида $y = a + bx$, где *Education* является объясняемой переменной, а *Fertility* регрессором с помощью команды *lm* пакета *lmtest* в языке R. Необходимо проделать те же операции, что и при построении первой модели.

С помощью команды *summary*, встроенной в R, найдем коэффициенты a , b (в таблице 1.2 они находятся в столбце *Estimate*: a – свободный коэффициент (*Intercept*), b – коэффициент *Fertility*).

Таким образом, в модели, где *Education* выступает в роли объясняемой переменной, а *Fertility* в роли регрессора значения коэффициентов будут равны: $a = 46.8178$, $b = -0.5109$.

Модель вида $y = a + bx$, где *Education* – объясняемая переменная, а *Fertility* – регрессор (подробные характеристики данной модели представлены в таблице 1.2):

$$y = 46.8178 + -0.5109 * x$$

Таблица 1.2. Характеристики модели зависимости параметра *Education* от параметра *Fertility* в наборе данных *Swiss*.

Коэффициенты / параметры	Estimate	Std. Error	t value	Pr (> t)	Уровень значимости
(Intercept)	46.8178	6.11244	7.659	1.08e-09	***
Fertility	-0.5109	0.08582	-5.954	3.66e-07	***

3. Оцените, насколько «хороша» модель по коэффициенту детерминации R^2 ?

Чтобы оценить, насколько «хороша» модель по коэффициенту детерминации R^2 , применим команду *summary*, встроенную в R, к модели вида $y = a + bx$, где *Education* является объясняемой переменной, а *Agriculture* регрессором. Здесь мы увидим, что *adjusted R-squared* = 0.4282, то есть коэффициент детерминации (R^2) приблизительно равен 43 %. Так как мы описываем *Education* всего одним параметром (*Agriculture*), то коэффициент детерминации не может быть высоким, поэтому, если брать в расчет, что *Education* описывается лишь одним параметром, то модель позволяет увидеть зависимость между уровнем образования призывников после начальной школы и мужчинами, которые занимаются сельским хозяйством. Анализируя модель далее, мы поймем, как именно *Education* зависит от *Agriculture*.

Для оценки модели вида $y = a + bx$, где *Education* является объясняемой переменной, а *Fertility* регрессором, по коэффициенту детерминации R^2 , также применим команду *summary*, встроенную в R. Как и в предыдущей модели коэффициент детерминации (R^2) будет не очень высоким, так как *Education* описывается только одним регрессором (*Fertility*). В данной модели R^2 приблизительно равен 43 %. Так как объясняемая переменная описывается только одним регрессором, то можно сказать, что модель подходит для изучения уровня образования призывников после начальной школы, с

помощью параметра *Fertility*. При дальнейшем анализе мы увидим, как именно уровень образования зависит от рождаемости в браке.

4. Оцените, есть ли взаимосвязь между объясняемой переменной и объясняющей переменной (по значению р-статистики, «количество звездочек» у регрессора в модели).

Из таблицы 1.1 видно, что уровень значимости между объясняемой переменной (*Education*) и регрессором (*Agriculture*) определено высок, ведь количество звездочек у переменных максимальное. Так как знак коэффициента *Agriculture* отрицательный, это говорит нам, что уровень образования был высок прежде всего у городского населения, то есть между *Agriculture* и *Education* скорее отрицательная связь, нежели положительная, ведь чем больше мужчин, занимающихся сельским хозяйством, тем ниже их уровень образования.

Из таблицы 1.2 видно, что уровень значимости между объясняемой переменной (*Education*) и регрессором (*Fertility*) определено высок, ведь количество звездочек у переменных максимальное, как и в случае с первой моделью. Так как знак коэффициента *Fertility* отрицательный, это говорит нам, что уровень образования был высок прежде всего у тех людей, которые не имели детей в браке, то есть между *Fertility* и *Education* скорее отрицательная связь, нежели положительная, ведь чем больше детей было в браке, тем ниже уровень образования у призывников у этой семьи.

Код решения задачи и сведения о проверенных моделях приведены в Приложении 1.

Вывод: В задаче №1 я смог оценить средние значения, дисперсии и среднеквадратичные отклонения переменных *Agriculture*, *Fertility* и *Education* из набора данных *Swiss*. В результате мною были построены две линейные регрессии, где в качестве регрессоров выступали *Agriculture* и *Fertility*, а в качестве объясняемой переменной - *Education*, которые я оценил по величине коэффициента детерминации R^2 , а также по уровню значимости регрессоров при помощи Р-статистики и положительных/отрицательных связях между объясняемой переменной и её регрессорами. По итогу первая модель оказалась информативной, так как коэффициент детерминации R^2 в первой модели, где *Education* – объясняемая переменная, а *Agriculture* – объясняющая переменная, равен приблизительно 43 %, уровень значимости объясняемой переменной от регрессора максимальный (так как количество “звездочек” максимальное – 3 штуки). При оценке коэффициентов регрессоров данных моделей выяснилось, что между *Agriculture* и *Education* прослеживается отрицательная связь (так как коэффициент у регрессора отрицательный). С помощью анализа данной модели, был сделан вывод, что уровень образования был высок прежде всего у городского населения. Вторая модель оказалась тоже информативной, так как коэффициент детерминации R^2 в первой модели, где *Education* – объясняемая переменная, а *Fertility* – объясняющая переменная, равен приблизительно 43 %, уровень значимости объясняемой переменной от регрессора максимальный (так как количество “звездочек” максимальное – 3 штуки), Как и в первой модели, коэффициент регрессора оказался отрицательным, что говорит нам об отрицательной связи между *Education* и *Fertility*. С помощью анализа данной модели, был сделан вывод, что уровень образования был высок прежде всего у тех людей, которые не имели детей в браке.

Задача №2

Необходимо загрузить данные из указанного набора и произвести следующие действия.

Набор данных: *Swiss*.

Объясняемая переменная: *Infant.Mortality*.

Регрессоры: *Education, Agriculture, Catholic*.

Чтобы приступить к решению данной задачи, следует первым делом оценить, насколько «хороша» исходная модель по коэффициенту детерминации R^2 и проанализировать, стоит ли работать с данной моделью или же добавить в условие дополнительный регрессор, от которого будет зависеть объясняемая переменная.

Итак, построим модель с помощью команды *lm* (пакета *lmtree* в языке R), где объясняемая переменная (*Infant.Mortality*) будет зависеть от регрессоров (*Education, Agriculture, Catholic*). Далее с помощью команды *summary* произведем анализ данной модели, где выявим зависимость между объясняемой переменной и ее регрессорами.

Из результата работы данной команды (Таблица 2.1) замечаем, что зависимости между объясняемой переменной и регрессорами нет. Такой вывод был сделан, исходя из коэффициента детерминации R^2 в данной модели (в нашем случае он составляет приблизительно 3 %), а также из количества звездочек (уровня значимости) у регрессоров (в нашем случае они отсутствуют).

Таблица 2.1. Характеристики модели зависимости параметра *Infant.Mortality* от параметров (*Education, Agriculture, Catholic*) в наборе данных *Swiss*.

Коэффициенты / параметры	Estimate	Std. Error	t value	Pr (> t)	Уровень значимости
(Intercept)	22.36903	1.75389	12.754	3.29e-16	***
Education	-0.08520	0.05771	-1.476	0.1472	
Agriculture	-0.04490	0.02636	-1.703	0.0957	.
Catholic	0.01904	0.01117	1.705	0.0954	.

Итак, чтобы улучшить модель, введем новый регрессор *Fertility* и посмотрим, насколько откорректированная модель будет лучше исходной.

Проделываем те же самые операции, что и при построении исходной модели. Сначала строим модель, где объясняемая переменная (*Infant.Mortality*) будет зависеть от регрессоров (*Education, Agriculture, Catholic, Fertility*), а далее с помощью команды *summary* проанализируем, какова будет зависимость между объясняемой переменной и регрессорами.

Из Таблицы 2.2, мы видим, что благодаря всего лишь одному регрессору (*Fertility*) коэффициент детерминации R^2 вырос в несколько раз (около 17 %), а у регрессора (*Fertility*) уровень значимости, в отличие от других регрессоров, намного выше (количество «звездочек» - 2).

Таблица 2.2. Характеристики модели зависимости параметра *Infant.Mortality* от параметров (*Education, Agriculture, Catholic, Fertility*) в наборе данных *Swiss*.

Коэффициенты / параметры	Estimate	Std. Error	t value	Pr (> t)	Уровень значимости
(Intercept)	22.36903	1.75389	12.754	3.29e-16	***
Education	-0.08520	0.05771	-1.476	0.1472	
Agriculture	-0.04490	0.02636	-1.703	0.0957	.
Catholic	0.01904	0.01117	1.705	0.0954	.
Fertility	0.147984	0.052400	2.824	0.00722	**

Таким образом, мы улучшили нашу исходную модель и дальнейшие ее улучшения могут положительно сказаться на коэффициенте детерминации R^2 , а также на уровне значимости регрессоров.

Приступим к решению задачи №2.1.

- 1) Проверьте, что в наборе данных нет линейной зависимости (построить зависимости между переменными, указанными в варианте, и проверить, что R^2 в каждой из них невысокий). В случае, если R^2 большой, один из таких столбцов можно исключить из рассмотрения.

Чтобы проверить, нет ли в наборе данных линейной зависимости, построим зависимости между регрессорами и проверим, каков их коэффициент детерминации R^2 . Также линейную зависимость можно проверить с помощью команды *vif* (пакета *car* в языке R).

Итак, построим 4 зависимости с помощью команды *lm*, где каждый регрессор будет выражаться через остальные и проверим их коэффициент детерминации:

- В модели, где объясняемая переменная *Agriculture*, а объясняющие (*Education, Catholic, Fertility*) коэффициент детерминации R^2 приблизительно 55 %.
- В модели, где объясняемая переменная *Catholic*, а объясняющие (*Education, Agriculture, Fertility*) коэффициент детерминации R^2 около 42 %.
- В модели, где объясняемая переменная *Fertility*, а объясняющие (*Education, Catholic, Agriculture*) коэффициент детерминации R^2 около 62 %.
- В модели, где объясняемая переменная *Education*, а объясняющие (*Agriculture, Catholic, Fertility*) коэффициент детерминации R^2 приблизительно 70 %.

Во всех моделях нетрудно заметить, что R^2 не маленький, однако самый большой он у последней модели. Это значит, что *Education* линейно зависим от остальных регрессоров. Исключим его из нашей модели и проверим, насколько изменится коэффициент детерминации R^2 , а также проверим, нет ли в наборе данных линейной зависимости уже без регрессора *Education*.

Построим модель, где объясняемой переменной будет *Infant.Mortality*, а регрессорами *Agriculture, Catholic, Fertility*, и проверим её на линейную зависимость уже без параметра *Education*.

Построим модели, где регрессор будет выражаться через оставшиеся и проверим их коэффициент детерминации.

- В модели, где объясняемая переменная *Agriculture*, а объясняющие (*Catholic*, *Fertility*) коэффициент детерминации R^2 приблизительно 16 %.
- В модели, где объясняемая переменная *Catholic*, а объясняющие (*Agriculture*, *Fertility*) коэффициент детерминации R^2 около 24 %.
- В модели, где объясняемая переменная *Fertility*, а объясняющие (*Catholic*, *Agriculture*) коэффициент детерминации R^2 около 21 %.

Коэффициент детерминации во всех моделях низок, а это значит, что ни один регрессор не может быть выражен через остальные, а из этого следует, что линейной зависимости нет.

Удостоиться в этом можно с помощью команды *vif*, упомянутой выше. При ее применении видно, что коэффициенты у каждого регрессора меньше 1,4, а это значит, что линейной зависимости между регрессорами нет.

- 2) Постройте линейную модель зависимой переменной от указанных в варианте регрессоров по методу наименьших квадратов (команда *lm* пакета *lmtest* в языке R). Оценить, насколько хороша модель, согласно: 1) R^2 , 2) p-значениям каждого коэффициента.

В корректировке модели мы уже проводили некоторые оценки исходной модели, поэтому оценим откорректированную.

Чтобы оценить линейную модель по методу наименьших квадратов, будем постепенно добавлять регрессоры в модель, чтобы увидеть растет ли коэффициент детерминации R^2 .

Построим модель, где объясняемой переменной будет *Infant.Mortality*, а регрессором *Fertility*. (*)

Узнаем коэффициент детерминации R^2 данной модели с помощью команды *summary*.

Таблица 2.3. Характеристики модели зависимости параметра *Infant.Mortality* от параметра *Fertility* в наборе данных *Swiss*.

Коэффициенты / параметры	Estimate	Std. Error	t value	Pr (> t)	Уровень значимости
(Intercept)	13.12970	2.25063	5.834	5.51e-07	***
Fertility	0.09713	0.03160	3.074	0.00359	**

Из Таблицы 2.3 видно, что уровень значимости регрессора *Fertility* высокий, так как количество звездочек у него почти максимальное, а коэффициент детерминации R^2 около 15 %.

Добавим регрессор *Agriculture* в модель (*) и оценим ее по коэффициенту детерминации и уровню значимости.

Таблица 2.4. Характеристики модели зависимости параметра *Infant.Mortality* от параметра (*Agriculture*, *Fertility*) в наборе данных *Swiss*. (**)

Коэффициенты/ параметры	Estimate	Std. Error	t value	Pr (> t)	Уровень значимости
(Intercept)	13.30130	2.20938	6.020	3.15e-07	***
Fertility	0.11669	0.03312	3.523	0.00101	**
Agriculture	-0.03047	0.01822	-1.672	0.10155	

Из Таблицы 2.4 видно, что уровень значимости регрессора *Agriculture* не такой высокий, как у *Fertility*, так как звездочки у него отсутствуют, однако коэффициент детерминации R^2 вырос и стал почти 19 %.

Добавим еще один регрессор *Catholic* в модель (**) и оценим ее по коэффициенту детерминации и уровню значимости.

Таблица 2.5. Характеристики модели зависимости параметра *Infant.Mortality* от параметра (*Agriculture*, *Fertility*, *Catholic*) в наборе данных *Swiss*.

Коэффициенты/ параметры	Estimate	Std. Error	t value	Pr (> t)	Уровень значимости
(Intercept)	13.565009	2.362940	5.741	8.7e-07	***
Fertility	0.112074	0.036105	3.104	0.00337	**
Agriculture	-0.032335	0.019207	-1.683	0.09953	.
Catholic	0.003754	0.011045	0.340	0.73560	

Из Таблицы 2.5 видно, что уровень значимости регрессора *Catholic* низкий, так как «звездочки» у него отсутствуют и коэффициент детерминации R^2 упал на 2 %.

Следовательно, регрессор *Catholic* можно убрать из рассмотрения. Тогда мы получаем модель, с которой будем в дальнейшем работать. В ней объясняемая переменная - *Infant.Mortality*, а регрессоры - *Agriculture* и *Fertility*.

- 3) Ведите в модель логарифмы регрессоров (если возможно). Сравнить модели и выбрать наилучшую.

Чтобы добавить логарифмы регрессоров в нашу модель, необходимо сначала исследовать регрессоры на линейную зависимость от исходных.

Построим модель с помощью команды *lm* (пакета *lmtest* в языке R), где объясняемой переменной будет *log (Fertility)*, а ее регрессором *Fertility*, а после проверим у нее коэффициент детерминации R^2 . После применения команды *summary* замечаем, что R^2 у этой модели высокий (около 98 %), поэтому следует заменить исходный регрессор логарифмом, чтобы избежать линейной зависимости.

Теперь проведем аналогичный анализ с другой моделью, где объясняемой переменной будет *log (Agriculture)*, а регрессором - *Agriculture*. Заметим, что и в данной модели ко-

коэффициент детерминации R^2 высокий (около 76 %), а это значит, что следует заменить исходный регрессор логарифмом, чтобы избежать линейной зависимости.

Попробуем ввести логарифмы от регрессоров в модель, где объясняемой переменной будет *Infant.Mortality*, а регрессорами (*log (Fertility)*, *Agriculture*). Чтобы команда *lm* правильно воспринимала логарифм от регрессора, необходимо при вводе обозначить ее в виде функции *I(log(Fertility))*.

Таблица 2.6. Характеристики модели зависимости параметра *Infant.Mortality* от параметра (*I(log(Fertility))*), *Agriculture*) в наборе данных *Swiss*.

Коэффициенты/параметры	Estimate	Std. Error	t value	Pr (> t)	Уровень значимости
(Intercept)	-9.82421	8.76601	-1.121	0.26849	
<i>I(log(Fertility))</i>	7.41730	2.14304	3.461	0.00121	**
<i>Agriculture</i>	-0.03219	0.01850	-1.740	0.08888	.

Из таблицы 2.6 заметно, что у нашей модели выросли стандартные ошибки из-за ввода логарифма от регрессора, а также упал коэффициент детерминации R^2 .

Попробуем ввести логарифм от регрессора *Agriculture* в модель, где объясняемой переменной будет *Infant.Mortality*.

Таблица 2.7. Характеристики модели зависимости параметра *Infant.Mortality* от параметра (*I(log(Agriculture))*), *Fertility*) в наборе данных *Swiss*.

Коэффициенты/параметры	Estimate	Std. Error	t value	Pr (> t)	Уровень значимости
(Intercept)	14.33188	2.46659	5.810	6.41e-07	***
<i>Fertility</i>	0.11592	0.03535	3.280	0.00204	**
<i>I(log(Agriculture))</i>	-0.67282	0.57605	-1.168	0.24910	

Из таблицы 2.7 заметно, что у нашей модели выросли стандартные ошибки из-за ввода логарифма от регрессора, как и в прошлом случае, а также упал коэффициент детерминации R^2 почти на 3 %.

Теперь введем логарифмы от обоих регрессоров в модель, где объясняемой переменной будет *Infant.Mortality*.

Таблица 2.8. Характеристики модели зависимости параметра *Infant.Mortality* от параметра (*I(log(Agriculture))*, *I(log(Fertility))*) в наборе данных *Swiss*.

Коэффициенты/параметры	Estimate	Std. Error	t value	Pr (> t)	Уровень значимости
(Intercept)	-9.3531	8.9682	-1.043	0.30269	
<i>I(log(Fertility))</i>	7.6459	2.3399	3.268	0.00211	**
<i>I(log(Agriculture))</i>	-0.8195	0.5986	-1.369	0.17795	

Из таблицы 2.8 заметно, что у нашей модели выросли стандартные ошибки из-за ввода логарифма от регрессора, как и в прошлых случаях, а также упал коэффициент детерминации R^2 почти на 3 %.

Делаем вывод, что логарифмы от регрессоров не улучшают нашу модель, поэтому необходимо вводить различные произведения от регрессоров, чтобы добиться ее улучшения.

- 4) Ведите в модель всевозможные произведения пар регрессоров, в том числе квадраты регрессоров. Найдите одну или несколько наилучших моделей по доле объяснённого разброса в данных R^2 .

Также, как и в случае с логарифмами, исследуем квадраты и произведение регрессоров на линейную зависимость от исходных регрессоров.

- Построим модель, где объясняемой переменной будет $(Fertility)^2$, а регрессором *Fertility* и проверим ее коэффициент детерминации R^2 через команду *summary*. В нашем случае R^2 приблизительно равен 98%.
- Построим модель, где объясняемой переменной будет $(Agriculture)^2$, а регрессором *Agriculture* и проверим ее коэффициент детерминации R^2 через команду *summary*. В нашем случае R^2 приблизительно равен 94%.
- Построим модель, где объясняемой переменной будет $(Agriculture * Fertility)$, а регрессором *I(Agriculture * Fertility)* и проверим ее коэффициент детерминации R^2 через команду *summary*. Здесь мы также видим, что подгонка слишком хорошая.

Делаем вывод, что исходные регрессоры необходимо заменить квадратами или произведением, чтобы избежать линейной зависимости.

Построим зависимости с всевозможными парами произведений регрессоров.

Первая модель:

Infant.Mortality - объясняемая переменная.

$(I(Fertility^2), Agriculture)$ - регрессоры.

Таблица 2.9. Характеристики модели зависимости параметра *Infant.Mortality* от параметра $(I(Fertility^2), Agriculture)$ в наборе данных *Swiss*.

Коэффициенты/параметры	Estimate	Std. Error	t value	Pr (> t)	Уровень значимости
(Intercept)	17.1258778	1.3318940	12.858	< 2e-16	***
$I(Fertility^2)$	0.0008341	0.0002426	3.439	0.00129	**
Agriculture	-0.0279234	0.0181042	-1.542	0.13015	

Из таблицы 2.9 видно, что уровень значимости у регрессоров не вырос и остался на прежнем уровне, при этом коэффициент детерминации R^2 приблизительно 18 %.

Вторая модель:

Infant.Mortality - объясняемая переменная.

$(Fertility, I(Agriculture^2))$ - регрессоры.

Таблица 2.10. Характеристики модели зависимости параметра $Infant.Mortality$ от параметра $(Fertility, I(Agriculture^2))$ в наборе данных *Swiss*.

Коэффициенты/ параметры	Estimate	Std. Error	t value	Pr (> t)	Уровень значимости
(Intercept)	12.8818352	2.1762479	5.919	4.43e-07	***
Fertility	0.1171277	0.0320056	3.660	0.000673	***
I(Agriculture^2)	-0.0003760	0.0001819	-2.068	0.044587	*

Из таблицы 2.10 видно, что стандартные ошибки у регрессоров стали низкими, а также количество «звездочек» у каждого регрессора увеличилось, при этом коэффициент детерминации R^2 увеличился и стал чуть выше 21 %.

Третья модель:

$Infant.Mortality$ - объясняемая переменная.

$(I(Fertility^2), I(Agriculture^2))$ - регрессоры.

Таблица 2.11. Характеристики модели зависимости параметра $Infant.Mortality$ от параметра $(I(Fertility^2), I(Agriculture^2))$ в наборе данных *Swiss*.

Коэффициенты / параметры	Estimate	Std. Error	t value	Pr (> t)	Уровень значимости
(Intercept)	1.962e+01	9.671e-01	20.290	<2e-16	***
I(Fertility * Agriculture)	8.749e-05	2.374e-04	0.369	0.714	

Как заметно из таблицы 2.11, в этой модели неадекватные коэффициенты и стандартные ошибки, а коэффициент детерминации R^2 у этой модели стал отрицательным, что говорит нам об ужасной модели.

Таким образом, делаем вывод, что лучшей моделью из всех является вторая модель, где $Infant.Mortality$ - объясняемая переменная, а $(Fertility, I(Agriculture^2))$ - регрессоры.

Теперь приступим к решению задачи №2.2.

Для зависимости, построенной при решении практического задания №2, оцените:

- 1) Доверительные интервалы для всех коэффициентов в модели, р = 95%.

Для того, чтобы найти доверительные интервалы для коэффициентов в модели, необходимо найти значение t-критерия Стьюдента. Кол-во измерений в обучающей выборке 47, из них 3 коэффициента рассчитаны. Число степеней свободы в модели $(47 - 3 = 44)$. Используем команду *qt* в языке R, чтобы посчитать критерий Стьюдента для нашей модели.

Итак, критерий Стьюдента для нашей модели приблизительно равен 2.015.

Доверительный интервал строится по формуле:

[Estimate - t * Std.Error; Estimate + t * Std.Error], где *Estimate* - значение коэффициента регрессора, *t* - критерий Стьюдента, *Std.Error* - значение стандартной ошибки регрессора.

Найдем доверительный интервал для коэффициента *k1*(коэффициента *Fertility*). *Estimate* (*Fertility*) = 0.12; *St. Error* (*Fertility*) = 0.03; *t* = 2.015;

Доверительный интервал для *k1*: [0.12 - 2.015*0.03; 0.12 + 2.015*0.03], =>

=> *k1*: [0.05955; 0.18045].

Найдем доверительный интервал для коэффициента *k2*(коэффициента *Agriculture^2*).

Estimate (*Agriculture^2*) = -0.0003; *St. Error* (*Agriculture^2*) = 0.0001;

Доверительный интервал для *k2*: [-0.0003 - 2.015*0.0001; -0.0003 + 2.015*0.0001], =>

=> *k2*: [-0.0005015; -0.0000985].

- 2) Сделайте вывод об отвержении или невозможности отвергнуть статистическую гипотезу о том, что коэффициент равен 0.

В оба доверительных интервала для коэффициентов *k1* и *k2* не попадает 0, значит коэффициенты точно не равны 0 на уровне значимости 5 %.

- 3) Доверительный интервал для одного прогноза (p = 95%, набор значений регрессоров выбираете сами).

Найдем доверительный интервал для прогноза с регрессорами *Fertility* = 25, *I(Agriculture^2)* = 900, p=95%.

```
model_best = lm(Infant.Mortality~Fertility+I(Agriculture^2), data)
new.data = data.frame(Fertility = 25, Agriculture = 30)
predict(model_best, new.data, interval = "confidence")
```

Рисунок 2.1. Код для построения доверительного интервала для прогноза с регрессорами *Fertility* = 25, *I(Agriculture^2)* = 900, p=95%.

Таблица 2.12. Результат кода для построения доверительного интервала для прогноза с регрессорами *Fertility* = 25, *I(Agriculture^2)* = 900, p=95%.

fit	lwr	upr
15.47159	12.59255	18.35063

Из значений таблицы 2.12 мы можем узнать доверительный интервал для нашего прогноза и прогноз модели для *Fertility* = 25, *Agriculture* = 30.

Прогноз модели (для *Fertility* = 25, *Agriculture* = 30) оценивается как 15.47.

Доверительный интервал: [12.59; 18.35].

Код решения задачи и сведения о проверенных моделях приведены в Приложении 2.

Вывод: В задаче №2.1 исследовалась модель, где объясняемой переменной была *Infant.Mortality*, а регрессорами - *Education*, *Agriculture*, *Catholic*. Так как модель изначально была плохая (так как коэффициент детерминации $R^2 \sim 3\%$ и уровень значимости регрессоров был низок («звездочек» не было)), пришлось добавить дополнительный регрессор *Fertility*, чтобы наша модель стала рабочей. Была выполнена проверка на линейную зависимость регрессоров: *Education*, *Agriculture*, *Catholic*, *Fertility*, в ходе которой был исключён регрессор *Education*, так как в линейной регрессии, где *Education* – объясняемая переменная, а регрессоры - *Agriculture*, *Catholic*, *Fertility*, был высокий $R^2 \sim 70\%$. После первичной проверки была произведена следующая проверка на линейную зависимость, но уже без регрессора *Education*, где выяснилось, что все объясняемые переменные линейно независимы от друг друга. После была построена модель по методу наименьших квадратов, где объясняемой переменной стал параметр - *Infant.Mortality*, а регрессорами *Agriculture*, *Fertility*, *Catholic*. В данной модели была оценена точность по коэффициенту детерминации $R^2 \sim 19\%$. С помощью команды *summary* мы узнали, что в нашей модели есть незначимый регрессор – *Catholic* (потому что его уровень значимости был минимален), поэтому я исключил его из нашей модели. R^2 стал приблизительно 17 %. Взяв за основу данную модель, я попытался ввести логарифмы от регрессоров, предварительно проверив на линейную зависимость регрессора от его логарифма. К сожалению, введение логарифмов от регрессоров не дало значительных результатов, поэтому пришлось вводить различные попарные произведения регрессоров, а также их квадраты. Проанализировав 3 модели, предварительно проверив их на линейную зависимость регрессора от его квадрата и регрессоров от их произведения, я выявил лучшую модель, которой стала модель, где *Infant.Mortality* – объясняемая переменная, а (*Fertility*, $I(Agriculture^2)$) - регрессоры. Её R^2 чуть выше 21 %, а уровень значимости высокий у всех регрессоров. Для этой модели был найден доверительный интервал, а также сделан вывод о том, может ли коэффициент быть равен 0 или нет. Доверительный интервал для коэффициента k_1 (коэффициента *Fertility*): [0.05955; 0.18045], а доверительный интервал для коэффициента k_2 (коэффициента *Agriculture^2*): [0.05955; 0.18045]. В оба доверительных интервала не попал 0, значит коэффициенты точно не равны 0 на уровне значимости 5 %. Далее был рассчитан доверительный интервал для прогноза с регрессорами *Fertility* = 25, *Agriculture* = 30: [12.59; 18.35], а также сам прогноз: 15.47.

Задача №3

Номер волны выборки РМЭЗ: 14

Подмножества для пункта 5: Женщины, не замужем; женщины, живущие в городе, разведённые

Начать данную задачу необходимо с выбора набора параметров, который необходим, для описания социально-экономического положения граждан Российской Федерации. Я выбрал следующие параметры: зарплата, пол, семейное положение, наличие высшего образования, возраст, тип населенного пункта, длительность рабочей недели, наличие подчиненных, удовлетворенность работе, является ли государство владельцем (совладельцем) предприятия, являются ли иностранные фирмы и частники владельцами (совладельцами) предприятия.

Данные параметры в исходной таблице соответственно имеют названия: *jj13.2, jh5, j_marst, j_educ, j_age, status, jj6.2, jj6, jj1.1.2, jj23, jj24*.

Выбранные данные необходимо преобразовать для дальнейшего анализа по следующим принципам:

- Из параметра, отвечающего семейному положению, сделать дамми-переменные (с помощью one-hot-encoding): 1) переменная *wed1* имеет значение 1 в случае, если респондент женат, 0 – в противном случае; 2) *wed2* = 1, если респондент разведён или вдовец; 3) *wed3* = 1, если респондент никогда не состоял в браке.
- Из параметра пол сделайте переменную *sex*, имеющую значение 1 для мужчин и равную 0 для женщин.
- Из параметра, отвечающего типу населённого пункта, создайте одну дамми-переменную *city_status* со значением 1 для города или областного центра, 0 – в противоположном случае.
- Введите один параметр *higher_educ*, характеризующий наличие полного высшего образования.
- Факторные переменные, «имеющие много значений», такие как: зарплата, длительность рабочей недели и возраст, - необходимо преобразовать в вещественные переменные и нормализовать их: вычесть среднее значение по этой переменной, разделить её значения на стандартное отклонение.
- Для всех остальных параметров сделаем отдельные дамми-переменные, которые будут иметь значение 1 для случая, когда выполняется условие в заданном вопросе и значение 0 для случая, когда не выполняется.

Тем самым мы получили рабочую таблицу *data_main* с новыми переменными, с которой и будем работать.

Приступим к решению задачи №3.

- 1) Постройте линейную регрессию зарплаты на все параметры, которые Вы выделили из данных мониторинга. Не забудьте оценить коэффициент вздутия дисперсии VIF.

Итак, построим линейную зависимость зарплаты от остальных параметров с помощью команды *lm* (пакета *lmtest* в языке R):

```
model_main = lm(salary~wed1+wed2+duration+age+sex+city_status
                 +higher_educ+satisfaction+employees+foreign_owner
                 +state_owner, data_main)
summary(model_main)
vif(model_main)
```

Рисунок 3.1. Код для построения линейной зависимости зарплаты от остальных параметров.

С помощью команды *summary* сможем определить коэффициент детерминации получившейся модели, а также выявить зависимость объясняемой переменной от регрессоров. В нашем случае коэффициент детерминации R^2 приблизительно равен 23 %, а максимальное количество «звездочек» имеют почти все регрессоры, за исключением *wed1* (не имеет «звездочек»), *wed2* (не имеет «звездочек»), *wed3* (имеет 2 «звездочки»). Значения коэффициента вздутия дисперсии варьируются от 1.03 до 2.86, при этом наибольшие значения VIF имеют *wed1*, *wed2*. С подробной характеристикой получившейся модели можно ознакомиться в таблице 3.1.

Таблица 3.1. Характеристики модели зависимости *salary ~ wed1, wed2, wed3, duration, age, sex, city_status, higher_educ, satisfaction, employees, foreign_owner, state_owner* в наборе данных 14-ой волны исследования.

Коэффициенты / параметры	Estimate	Std. Error	t value	Pr (> t)	Уровень значимости
(Intercept)	-0.553932	0.063822	-8.679	< 2e-16	***
wed1	0.008673	0.054979	0.158	0.874666	
wed2	-0.064154	0.066338	-0.967	0.333586	
wed3	-0.217328	0.067268	-3.231	0.001248	**
duration	0.114117	0.016721	6.825	1.06e-11	***
age	-0.063725	0.018323	-3.478	0.000513	***
sex	0.411687	0.034127	12.064	< 2e-16	***
city_status	0.320520	0.036710	8.731	< 2e-16	***
higher_educ	0.345263	0.039030	8.846	< 2e-16	***
satisfaction	0.281742	0.032774	8.596	< 2e-16	***
employees	0.381917	0.041143	9.283	< 2e-16	***
foreign_owner	0.520216	0.087334	5.957	2.88e-09	***

state_owner	-0.267020	0.033519	-7.966	2.31e-15	***
-------------	-----------	----------	--------	----------	-----

- 2) Поэкспериментируйте с функциями вещественных параметров: используйте логарифмы, степени (хотя бы от 0.1 до 2 с шагом 0.1), произведения вещественных регрессоров.

В нашем случае вещественными переменными являются *age* (возраст), *duration* (длительность рабочей недели). С ними и будем экспериментировать, вводя различные функции от регрессоров.

Для начала стоит проверить насколько сильно от среднего отклоняются значения *age* и *duration*. Для этого узнаем минимальное значение этих параметров и прибавим ко всем значениям переменных некоторое значение чуть большее, чем их минимальное, для того чтобы логарифмы и вещественные степени от параметров хорошо брались.

В нашем случае минимальное значение у *age* ~ -2, а у *duration* ~ -3, поэтому прибавим у обоим параметрам 5.

Далее выполним проверку на линейную зависимость регрессора от своего же логарифма, где выявляем, что наши регрессоры зависимы от своих логарифмов, поэтому заменим данные параметры их логарифмами.

```
summary(lm(salary~wed1+wed2+wed3+duration+I(log(age+5))+sex+city_status
           +satisfaction+higher_educ+employees+foreign_owner
           +state_owner, data_main))
summary(lm(salary~wed1+wed2+wed3+I(log(duration+5))+age+sex+city_status
           +satisfaction+employees+higher_educ+foreign_owner
           +state_owner, data_main))
summary(lm(salary~wed1+wed2+wed3+I(log(duration+5))+I(log(age+5))+sex
           +city_status+satisfaction+employees+higher_educ+foreign_owner
           +state_owner, data_main))
```

Рисунок 3.2. Код для построения линейной зависимости зарплаты от остальных параметров с введением логарифмов от *age* и *duration*.

С помощью команды *summary* оценим коэффициенты детерминации R^2 у данных моделей: у первой модели $R^2 = 0.2311$, у второй модели $R^2 = 0.2324$, у третьей модели $R^2 = 0.2311$. Из этого делаем вывод, что при добавлении логарифмов от регрессоров коэффициент детерминации R^2 становится ниже или равен коэффициенту детерминации R^2 исходной модели, поэтому вводить логарифмы от регрессоров не имеет смысла.

Теперь возведем вещественные регрессоры *age* и *duration*, к которым мы прибавили 5, в степень от 0.1 до 2 с шагом 0.1. Попутно будем проверять их коэффициент взаимодействия дисперсии и оценивать модели по R^2 .

При возведении в степень наших вещественных регрессоров я уловил закономерность: чем больше наша степень, тем выше коэффициент детерминации R^2 (он менялся с

0.2314 до 0.2326), а коэффициенты вздутия дисперсии находились в пределе нормы. Поэтому самой лучшей моделью, где были введены степени от вещественных регрессоров, стала модель, где *salary* – объясняемая переменная, а *wed1*, *wed2*, *wed3*, $I((duration+5)^2)$, $I((age+5)^2)$, *sex*, *city_status*, *higher_educ*, *satisfaction*, *employees*, *foreign_owner*, *state_owner* – регрессоры.

Посмотрим, что будет с коэффициентом детерминации R^2 при введении произведения двух наших вещественных регрессоров: *age* и *duration*.

С помощью команды *summary*, нетрудно заметить, что коэффициент детерминации R^2 стал ниже, чем у исходной модели на 2 % (то есть $R^2 \sim 21\%$), при этом коэффициенты вздутия дисперсии у этой модели в пределах нормы.

```
summary(lm(salary~wed1+wed2+wed3+I(duration*age)+sex+city_status+higher_educ
        +satisfaction+employees+foreign_owner+state_owner, data_main))
vif(lm(salary~wed1+wed2+wed3+I(duration*age)+sex+city_status+higher_educ
        +satisfaction+employees+foreign_owner+state_owner, data_main))
```

Рисунок 3.3. Код для построения линейной зависимости зарплаты от остальных параметров с введением произведения *age* и *duration*.

- 3) Выделите наилучшие модели из построенных: по значимости параметров, включённых в зависимости, и по объяснённому с помощью построенных зависимостей разбросу adjusted $R^2 - R^2 \text{ adj}$.

В предыдущем пункте мы экспериментировали с введением степеней, произведений и логарифмов от регрессоров. Самой лучшей моделью оказалась модель, где *salary* – объясняемая переменная, а *wed1*, *wed2*, *wed3*, $I((duration+5)^2)$, $I((age+5)^2)$, *sex*, *city_status*, *higher_educ*, *satisfaction*, *employees*, *foreign_owner*, *state_owner* – регрессоры.

Таблица 3.2. Характеристики модели зависимости *salary ~ wed1, wed2, wed3, I((duration+5)^2), I((age+5)^2), sex, city_status, higher_educ, satisfaction, employees, foreign_owner, state_owner* в наборе данных 14-ой волны исследования.

Коэффициенты / параметры	Estimate	Std. Error	t value	Pr (> t)	Уровень значимости
(Intercept)	-0.596973	0.084766	-7.043	2.34e-12	***
wed1	0.011241	0.054930	0.205	0.837871	
wed2	-0.056603	0.066285	-0.854	0.393213	
wed3	-0.221504	0.066779	-3.317	0.000921	***
$I((duration + 5)^2)$	0.008942	0.001407	6.356	2.39e-10	***
$I((age + 5)^2)$	-0.007579	0.001758	-4.311	1.68e-05	***
sex	0.420104	0.034041	12.341	< 2e-16	***
city_status	0.322969	0.036714	8.797	< 2e-16	***

higher_educ	0.342182	0.038993	8.776	< 2e-16	***
satisfaction	0.282258	0.032772	8.613	< 2e-16	***
employees	0.386924	0.041076	9.420	< 2e-16	***
foreign_owner	0.516373	0.087350	5.912	3.78e-09	***
state_owner	-0.267665	0.033482	-7.994	1.85e-15	***

Из таблицы 3.2 мы видим, что у всех регрессоров уровень значимости максимальный кроме *wed1*, *wed2*, поэтому попробуем исключить из рассмотрения.

Таблица 3.3. Характеристики модели зависимости $\text{salary} \sim \text{wed3}, I((\text{duration}+5)^2), I((\text{age}+5)^2), \text{sex}, \text{city_status}, \text{higher_educ}, \text{satisfaction}, \text{employees}, \text{foreign_owner}, \text{state_owner}$ в наборе данных 14-ой волны исследования.

Коэффициенты / параметры	Estimate	Std. Error	t value	Pr (> t)	Уровень значимости
(Intercept)	-0.595182	0.074451	-7.994	1.85e-15	***
wed3	-0.222768	0.049609	-4.490	7.38e-06	***
I((duration + 5)^2)	0.008892	0.001406	6.323	2.96e-10	***
I((age + 5)^2)	-0.007868	0.001730	-4.547	5.66e-06	***
sex	0.431686	0.033083	13.049	< 2e-16	***
city_status	0.321121	0.036681	8.754	< 2e-16	***
higher_educ	0.343326	0.038905	8.825	< 2e-16	***
satisfaction	0.281633	0.032769	8.595	< 2e-16	***
employees	0.388668	0.041053	9.468	< 2e-16	***
foreign_owner	0.518595	0.087335	5.938	3.22e-09	***
state_owner	-0.266380	0.033431	-7.968	2.28e-15	***

Из таблицы 3.3 мы видим, что уровень значимости теперь максимальный у каждого регрессора, а коэффициент детерминации $R^2 = 0.2325$ (то есть упал на 0.0001). Значит, исключить *wed1*, *wed2* из нашей модели можно и модель без этих параметров оказалась лучшей, а значит именно с ней мы дальше будем работать.

- 4) Сделайте вывод о том, какие индивиды получают наибольшую зарплату.

В ходе нашего исследования мы выбрали наилучшую модель, где $salary$ – объясняемая переменная, а $wed3$, $I((duration+5)^2)$, $I((age+5)^2)$, sex , $city_status$, $higher_educ$, $satisfaction$, $employees$, $foreign_owner$, $state_owner$ – регрессоры.

По этой модели сделаем вывод о том, какие индивиды получают наибольшую зарплату, оценив знак коэффициента каждого регрессора из таблицы 3.3.

Итак, $wed3$ – отрицательный; $I((duration + 5)^2)$ – положительный; $I((age + 5)^2)$ – отрицательный; sex – положительный; $higher_educ$ – положительный; $city_status$ – положительный; $satisfaction$ – положительный; $employees$ – положительный; $foreign_owner$ – положительный; $state_owner$ – отрицательный.

Вывод о том, какие индивиды получают большую зарплату:

Большую зарплату получают в большинстве своём мужчины с продолжительной рабочей неделей, имеющие высшее образование, проживающие в городе, удовлетворённые своей заработной платой, имеющие подчиненных. При этом, если иностранные фирмы и частники являются совладельцами или владельцами Вашего предприятия, то это положительно сказывается на уровне заработной платы. Возраст не влияет на уровень заработанной платы, а если государство являются совладельцами или владельцами Вашего предприятия, то это отрицательно сказывается на уровне заработной платы.

- 5) Оцените лучшие модели для подмножества индивидов, указанных в варианте.
Сделайте вывод о том, какие индивиды получают наибольшую зарплату.

Выделим подмножество незамужних женщин с помощью применения функции *subset* дважды. Возьмём выбранную мной лучшую модель зависимости зарплаты от других параметров, и учтём, что находимся в подмножестве незамужних женщин (в этом подмножестве переменные *sex* и *wed1* равны единице).

```
data_woman = subset(data_main, sex==0) # Женщины
data_woman_moment = subset(data_woman, wed1==0) # Женщины не замужем
model_subset1 = lm(salary~wed3+I((duration+5)^2)+I((age+5)^2)+higher_educ
+city_status+satisfaction+employees+foreign_owner
+state_owner, data_woman_moment)
summary(model_subset1)
vif(model_subset1)
```

Рисунок 3.4. Код для построения подмножества незамужних женщин и линейной зависимости зарплаты от остальных параметров в подмножестве незамужних женщин.

У данной модели $R^2 = 0.2164$. VIF хороший для всех значений. Значения коэффициентов модели, их стандартные ошибки, р-статистика и уровень значимости приведены в таблице 3.4.

Таблица 3.4. Характеристики модели зависимости $salary \sim wed3$, $I((duration+5)^2)$, $I((age+5)^2)$, $city_status$, $higher_educ$, $satisfaction$, $employees$, $foreign_owner$, $state_owner$ в наборе данных 14-ой волны исследования в подмножестве незамужних женщин.

Коэффициенты	Estimate	Std. Error	t value	Pr (> t)	Уровень
--------------	----------	------------	---------	-----------	---------

/ параметры					значимости
(Intercept)	-0.355174	0.121991	-2.911	0.00371	**
wed3	-0.096646	0.063479	-1.522	0.12832	
I((duration + 5)^2)	0.002023	0.002492	0.812	0.41725	
I((age + 5)^2)	-0.004494	0.002714	-1.656	0.09815	.
city_status	0.215763	0.059936	3.600	0.00034	***
higher_educ	0.393825	0.058697	6.709	3.89e-11	***
satisfaction	0.145094	0.051702	2.806	0.00514	**
employees	0.357771	0.068311	5.237	2.13e-07	***
foreign_owner	0.717036	0.176812	4.055	5.54e-05	***
state_owner	-0.352804	0.053983	-6.536	1.18e-10	***

Незамужние женщины получают высокую заработную плату, если имеют диплом о высшем образовании, живут в городе, имеют подчиненных и собственником их предприятия является иностранная компания или частник. Такой вывод был сделан из знаков коэффициентов регрессоров из таблицы 3.4 (если высокие и положительные включаем в вывод).

Теперь выделим подмножество женщин, живущих в городе и разведенных, с помощью применения функции *subset* трижды. Возьмём выбранную мной лучшую модель зависимости зарплаты от других параметров, и учтём, что находимся в подмножестве разведенных женщин, живущих в городе (в этом подмножестве переменные *sex*, *wed2*, *city_status* равны единице).

```
data_woman_city = subset(data_woman, city_status==0) # Женщины, живущие в городе
data_woman_city_moment = subset(data_woman_city, wed2==1) # Женщины, живущие
# в городе, разведённые
model_subset1 = lm(salary~I((duration+5)^2)+I((age+5)^2)+higher_educ
                    +satisfaction+employees+state_owner, data_woman_city_moment)
summary(model_subset1)
vif(model_subset1)
```

Рисунок 3.5. Код для построения подмножества разведенных женщин, живущих в городе, и линейной регрессии зарплаты от остальных параметров в подмножестве разведенных женщин, живущих в городе.

У данной модели $R^2 = 0.274$. VIF хороший для всех значений. Значения коэффициентов модели, их стандартные ошибки, р-статистика и уровень значимости приведены в таблице 3.5.

Таблица 3.5. Характеристики модели зависимости $salary \sim I((duration+5)^2)$, $I((age+5)^2)$, $higher_educ$, $satisfaction$, $employees$, $foreign_owner$, $state_owner$ в наборе данных 14-ой волны исследования в подмножестве разведенных женщин, живущих в городе.

Коэффициенты / параметры	Estimate	Std. Error	t value	Pr (> t)	Уровень значимости
(Intercept)	-0.643888	0.307733	-2.092	0.039537	*
$I((duration + 5)^2)$	0.004561	0.006286	0.726	0.470202	
$I((age + 5)^2)$	0.002102	0.006991	0.301	0.764440	
$higher_educ$	0.348917	0.138534	2.519	0.013750	*
$satisfaction$	0.416494	0.110680	3.763	0.000316	***
$employees$	0.310300	0.133982	2.316	0.023088	*
$state_owner$	-0.390500	0.128451	-3.040	0.003185	**

Разведенные женщины, живущие в городе, получают высокую заработную плату, если имеют диплом о высшем образовании, удовлетворены своей работой и имеют своих подчиненных. Такой вывод был сделан из знаков коэффициентов регрессоров из команды *summary* (если высокие и положительные включаем в вывод).

Код решения задачи и сведения о проверенных моделях приведены в Приложении 3.

Выход: Мной был произведен анализ данных опроса НИУ ВШЭ о материальном состоянии граждан России за 2005 год, где требовалось найти зависимость между некоторыми параметрами из данных и зарплатой людей. Я выбрал минимальный набор данных (зарплата, продолжительность рабочей недели, возраст, пол, наличие высшего образования, тип населённого пункта, семейное положение), который был дан в условии, а также дополнительные параметры (наличие подчиненных, удовлетворенность работе, является ли государство владельцем (совладельцем) предприятия, являются ли иностранные фирмы и частники владельцами (совладельцами) предприятия). Далее мной были нормализованы исходные параметры, которые были преобразованы в вещественные и дамми-переменные. После мной была построена линейная регрессия зарплаты на все нормализованные параметры $salary \sim wed1$, $wed2$, $wed3$, $duration$, age , sex , $city_status$, $higher_educ$, $satisfaction$, $employees$, $foreign_owner$, $state_owner$. Коэффициент детерминации R^2 оказался приблизительно равен 23%. VIF у всех значений оказался в диапазоне от 1.03 до 2.86. Далее я поэкспериментировал с добавлением логарифмов функций от вещественных переменных (возраст и продолжительность рабочей

недели), их степеней от 0.1 до 2 с шагом 0.1 и их произведения. Для каждой модели я проверял R², VIF и количество значимых регрессоров. В ходе исследования был сделан вывод, что логарифмы от вещественных регрессоров, а также их произведение не улучшает исходную модель, а в переборе степеней замечена закономерность (при увеличении степеней модель становится лучше, оценивая её точность по коэффициенту детерминации R²). В итоге нашлась лучшая модель $\text{salary} \sim \text{wed3}, I((\text{duration}+5)^2), I((\text{age}+5)^2), \text{sex}, \text{city_status}, \text{higher_educ}, \text{satisfaction}, \text{employees}, \text{foreign_owner}, \text{state_owner}$. Оценив коэффициенты перед переменными, был сделан вывод, какие индивиды получают большую зарплату: Большинство получают в большинстве своём мужчины с продолжительной рабочей неделей, имеющие высшее образование, проживающие в городе, удовлетворённые своей заработной платой, имеющие подчиненных. При этом, если иностранные фирмы и частники являются совладельцами или владельцами Вашего предприятия, то это положительно сказывается на уровне заработной платы. В конце задания мной были выделены 2 подмножества (женщины не замужем; разведенные женщины, живущие в городе). На данных подмножествах мной были построены модели, после оценки которых получилось сделать вывод, какие индивиды получают большую зарплату. Незамужние женщины получают высокую заработную плату, если имеют диплом о высшем образовании, живут в городе, имеют подчиненных и собственником их предприятия является иностранная компания или частник. Разведенные женщины, живущие в городе, получают высокую заработную плату, если имеют диплом о высшем образовании, удовлетворены своей работой и имеют своих подчиненных.

Задача №4

Набор данных: *Credit card customers*

Тип классификатора: *DecisionTreeClassifier (решающее дерево)*

Классификация по столбцу: *Customer Age* (выше среднего значения – класс 0, ниже или совпадает – класс 1)

- 1) Обработайте набор данных набор данных, указанный во втором столбце таблицы 4.1, подготовив его к решению задачи классификации. Выделите целевой признак, указанный в последнем столбце таблицы, и удалите его из данных, на основе которых будет обучаться классификатор. Разделите набор данных на тестовую и обучающую выборку. Постройте классификатор типа, указанного в третьем столбце, для задачи классификации по параметру, указанному в последнем столбце. Оцените точность построенного классификатора с помощью метрик precision, recall и F1 на тестовой выборке.

Для начала считаем набор данных с устройства и преобразуем его в таблицу, удалив все строки, в которых встречаются пустые значения.

Первые 5 строк набора данных *Credit card customers* представлены в таблице 4.1.

Таблица 4.1. Первые 5 строк набора данных *Credit card customers*.

CLI-ENT NU-M	Attrition_Flag	Customer_Age	Gender	Educational_Leve l	Marital_Status	Income_Category	Total_Relations hip_Count	Avg_Utilization_Ratio
7688 0538 3	Existing Customer	45	M	High School	Married	\$60K - \$80K	5	0.061
8187 7000 8	Existing Customer	49	F	Graduate	Single	Less than \$40K	6	0.105
7139 8210 8	Existing Customer	51	M	Graduate	Married	\$80K - \$120K	4	0.000
7699 1185 8	Existing Customer	40	F	High School	Un-known	Less than \$40K	3	0.760
7091 0635 8	Existing Customer	40	M	Uneducated	Married	\$60K - \$80K	5	0.000

Перейдём к подготовке решения задачи классификации.

Все не числовые признаки нужно преобразовать в числовые или, если можно, в бинарные. Столбец *Attrition_Flag* преобразуем по принципу: 1 – *Existing Customer*, 0 – *Attrited Customer*. Столбец *Gender* также преобразуем в бинарный: 1 - *M*, 0 - *F*. Признак *Education_Level*: *Doctorate* - 0, *Post-Graduate* – 1, *Graduate* – 2, *College* – 3, *High School* – 4, *Unknown* – 5, *Uneducated* – 6. Столбец *Marital_Status*: *Married* – 0, *Divorced* – 1, *Unknown* – 2, *Single* - 3. Столбец *Income_Category*: *\$120K +* - 0, 1 - *\$80K - \$120K*, *\$60K - \$80K* – 2, *\$40K - \$60K* – 3, *Less than \$40K* – 4, *Unknown* - 5. Также обработаем столбец, по которому будем строить классификацию - *Customer Age* (выше среднего значения – класс 0, ниже или совпадает – класс 1).

Для построения классификатора этот столбец необходимо удалить из данных, оставив его отдельно от остальных столбцов, чтобы наш классификатор смог обучаться.

Первые 5 строк измененной таблицы, подготовленной к решению задачи классификации, представлен в таблице 4.2.

Таблица 4.2. Первые 5 строк измененной таблицы набора данных *Credit card customers*, подготовленной к решению задачи классификации.

CLI-ENT NUM	Attrition_Flag	Gender	Educational_Level	Marital_Status	Income_Category	Total_Relationship_Count	Avg_Utilization_Ratio
768805383	1	1	4	0	2	5	0.061
818770008	1	0	2	1	4	6	0.105
713982108	1	1	2	0	1	4	0.000
769911858	1	0	4	3	4	3	0.760
709106358	1	1	6	0	2	5	0.000

Можем приступить к задаче классификации. Разделим набор данных на обучающую и тестовую выборку. Размер тестовой выборки будет 33%.

Строим классификатор *DecisionTreeClassifier* (*решающее дерево*) для обучения, выставив *random_state* на 241 и глубину (*max_depth*) на 3 (так как при увеличении глубины точность нормализуется), а после построения данной модели, она может быть использована для прогнозирования значений. Код представлен на рисунке 4.1.

```
from sklearn.tree import DecisionTreeClassifier

tree = DecisionTreeClassifier(random_state = 241, max_depth = 3)
tree = tree.fit(x_train, y_train) # Обучение
y_pred = tree.predict(x_test) # Прогноз
```

Рисунок 4.1. Код построения классификатора *DecisionTreeClassifier*.

Прогнозом будет: 0.518920697597894.

Благодаря *classification_report* узнаем точность классификатора *DecisionTreeClassifier* при *max_depth* = 3 и *random_state* = 241 с помощью метрик *accuracy*, *precision*, *recall*, *f1*. Так как классификатор обучается каждый раз по-разному, следует взять среднее значение метрик для оценки точности. Код представлен на рисунке 4.2.

```
import sklearn.metrics
from sklearn.metrics import classification_report

report = classification_report(y_test, y_pred)
print(report)
```

Рисунок 4.2. Код оценки классификатора с помощью *classification_report*.

Результатом работы метода *classification_report* на тестовой выборке стали такие средние значения метрик:

accuracy = 0.532; *recall* = 0.71;

precision = 0.53; *f1* = 0.6.

Точность нашего классификатора *DecisionTreeClassifier* оказалась низкой, потому что довольно сложно оценить возраст клиентов кредитной карты через те признаки, которые даны в наборе данных.

Визуализируем классификатор Decision Tree (Решающее дерево) и найдем границы классов в виде неравенств. Также оценим важность признаков.

Визуализацию решающего дерева произведем с помощью библиотеки *matplotlib.pyplot*.

С помощью рисунка 4.3 легко выделить границы 2-ух классов в виде неравенств:

Характеристики некоторых объектов класса 0: *Marital_Status* <= 1,5; *Income_Category* <= 1,5.

Характеристики некоторых объектов класса 1: *Marital_Status* > 1,5; 1,5 < *Education_Level* <= 4,5.

Таким образом, важными признаками для выделения классов стали *Marital_Status* (семейное положение), *Education_Level* (уровень образования), *Income_Category* (годовой доход клиента).

Теперь детализируем, кто же относится к нулевому и первому классу:

Клиенты, состоящие в браке или не состоящие в браке, у которых годовой доход более 80-ти тысяч долларов относятся к 0-му классу, а это значит, что возраст таких клиентов выше среднего.

Клиенты, кто не указал свое семейное положение, которые являются обучающимися в старшей школе, колледже или выпускниками ВУЗа относятся к 1-му классу, а это значит, что возраст таких клиентов ниже среднего.

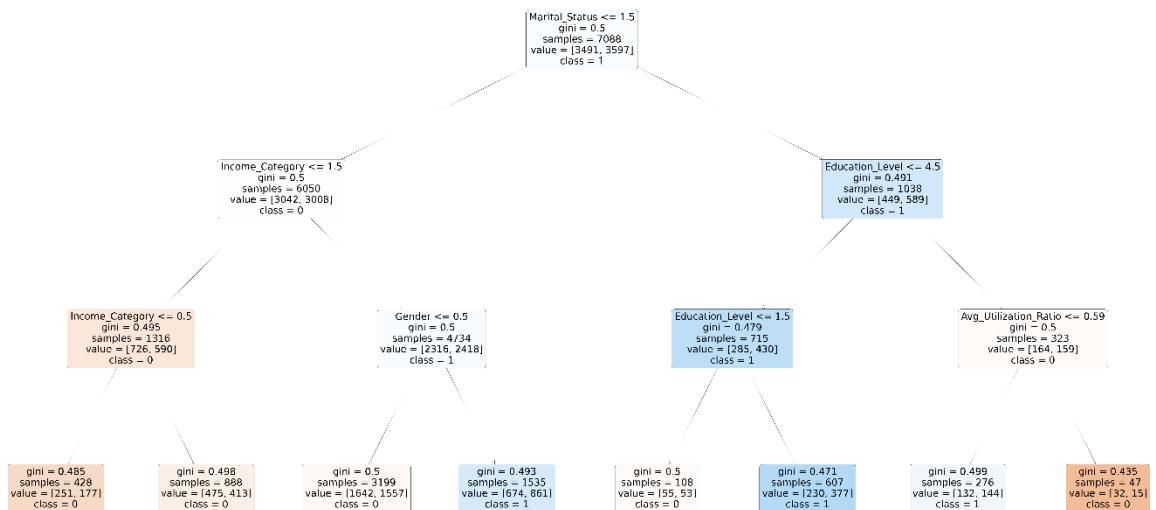


Рисунок 4.3. Визуализация классификатора *Decision Tree*.

- 2) Постройте классификатор типа Случайный Лес (*Random Forest*) для решения той же задачи классификации. Оцените его качество с помощью метрик *precision*, *recall* и *F1* на тестовой выборке. С помощью *GridSearch* переберите различные комбинации гиперпараметров: на первой итерации задайте большие шаги (50 или 100) по числу деревьев *n_estimators*. На следующих итерациях определите лучшее количество деревьев *n_estimators* с точностью до 10. Какой из классификаторов оказывается лучше?

Построим классификатор *Random Forest*, состоящий из 100, 200 и 300 так называемых решающих деревьев глубины от 1 до 20. Далее найдем лучшее количество деревьев с помощью *GridSearchCV*. Код построения *Random Forest* и поиска лучшего количества деревьев (от 100 до 300 с шагом 100) представлен на рисунке 4.4.

```

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

param_grid = {'n_estimators': list(range(100, 301, 100)), 'max_features': ['auto'], 'max_depth': list(range(1,20)), 'criterion': 'gini'}
RFC_100_300 = GridSearchCV(estimator=RandomForestClassifier(), param_grid=param_grid, cv=5, refit=True)
RFC_100_300.fit(X_train, y_train)
RFC_100_300.best_params_ # Выявление лучшего количества деревьев для Random Forest.
  
```

Рисунок 4.4. Код построения *Random Forest* и поиска лучшего количества деревьев (от 100 до 300 с шагом 100) с помощью *GridSearchCV*.

С помощью *GridSearchCV* мы узнали, что лучшим классификатором по числу деревьев (при оценке от 100 до 300 с шагом 100) является *Random Forest* из 300 решающих деревьев.

Теперь попробуем построить *Random Forest* из более, чем 300 деревьев (уменьшать смысла нет, так как у классификаторов, состоящих из менее, чем 300 деревьев хуже точность, исходя из результата *GridSearchCV*), и с помощью *GridSearchCV* выявим, какой наилучший *Random Forest* по количеству решающих деревьев. Код построения *Random Forest* и поиска лучшего количества деревьев (от 300 до 400 с шагом 10) представлен на рисунке 4.5.

```
param_grid = {'n_estimators' : list(range(300, 400, 10)), 'max_features' : ['auto'], 'max_depth': list(range(1,20)), 'criterion': 'gini'}
RFC_300_400 = GridSearchCV(estimator=RandomForestClassifier(), param_grid = param_grid, cv = 5, refit = True)
RFC_300_400.fit(X_train, y_train)
RFC_300_400.best_params_ # Выведение лучшего количества деревьев для Random Forest.
```

Рисунок 4.5. Код построения *Random Forest* и поиска лучшего количества деревьев (от 300 до 400 с шагом 10).

Исходя из результата *GridSearchCV*, лучшим классификатором по количеству решающих деревьев является *Random Forest* из 330 решающих деревьев.

Построим классификатор *Random Forest* из 330 решающих деревьев и оценим его точность с помощью среднего значения метрик *accuracy*, *precision*, *recall* и *f1*. Сравним с точностью построенного нами классификатора *Decision Tree*. Код построения *Random Forest* и оценки его точности с помощью метрик *accuracy*, *f1*, *recall*, *precision* представлен на рисунке 4.5.

```
param_grid = {'n_estimators' : [330], 'max_features' : ['auto'], 'max_depth': list(range(1,20)), 'criterion' :['gini']}
RFC_330 = GridSearchCV(estimator=RandomForestClassifier(), param_grid = param_grid, cv = 5, refit = True)
RFC_330.fit(X_train, y_train)

print("accuracy:" +str(np.average(sklearn.model_selection.cross_val_score(RFC.best_estimator_, X_test, y_test, scoring='accuracy')))
print("f1:" +str(np.average(sklearn.model_selection.cross_val_score(RFC.best_estimator_, X_test, y_test, scoring='f1'))))
print("precision:" +str(np.average(sklearn.model_selection.cross_val_score(RFC.best_estimator_, X_test, y_test, scoring='precision')))
print("recall:" +str(np.average(sklearn.model_selection.cross_val_score(RFC.best_estimator_, X_test, y_test, scoring='recall'))))
```

Рисунок 4.5. Код построения *Random Forest* и оценки его точности с помощью метрик *accuracy*, *f1*, *recall*, *precision*.

Точность *Random Forest* из 330 решающих деревьев:

accuracy = 0.5340531879534091; *precision* = 0.5357495033098759;

f1 = 0.5752483135706458; *recall* = 0.6318680043788645.

Сравним Random Forest, состоящий из 330 решающих деревьев, и наш классификатор Decision Tree.

Метрики Random Forest: *accuracy: 0.5340531879534091, f1: 0.5752483135706458, precision: 0.5357495033098759, recall: 0.6318680043788645.*

Метрики Decision Tree: *accuracy: 0.532, precision = 0.53, recall = 0.71, f1 = 0.6.*

Из данных метрик заметно, что все значения метрик выше у *Random Forest* из 330 решающих деревьев чем у *Decision Tree*. Отсюда делаем вывод, что лучшим из классификаторов оказался *Random Forest* из 330 решающих деревьев, так как его точность выше, исходя из значений метрик: *accuracy, f1, recall, precision*.

Вывод:

Мной была выполнена задача классификации для набора данных *Credit card customers* двумя разными способами: решающим деревом (*Decision Tree*) и случайнм лесом (*Random Forest*).

Требовалось разделить данные в столбце *Customer Age* по следующим критериям: выше среднего значения – класс 0, ниже или совпадает – класс 1. В начале я нормализовал все не числовые признаки в числовые или бинарные. После разбил данные на тренировочную и тестовую выборки и построил классификатор *DecisionTreeClassifier*. Оценка точности данного классификатора была произведена с помощью среднего значения метрик *accuracy* (0.532), *f1* (0.6), *precision* (0.53) и *recall* (0.71). Исходя из полученных данных, можно сделать вывод, что данный классификатор даёт правильный ответ примерно в 53 случаях из 100.

Далее я визуализировал классификатор *DecisionTreeClassifier* и нашел границы классов в виде неравенств. Характеристиками некоторых объектов класса 0 стали параметры: *Marital_Status <= 1,5; Income_Category <= 1,5*, а характеристиками некоторых объектов класса 1 стали: *Marital_Status > 1,5; 1,5 < Education_Level <= 4,5*. Важными признаками для выделения классов стали *Marital_Status* (семейное положение), *Education_Level* (уровень образования), *Income_Category* (годовой доход клиента). Далее я выявил, какие клиенты относятся к 0-му и 1-му классам: клиенты, состоящие в браке или не состоящие в браке, у которых годовой доход более 80-ти тысяч долларов относятся к 0-му классу, а это значит, что возраст таких клиентов выше среднего; клиенты, кто не указал свое семейное положение, которые являются обучающимися в старшей школе, колледже или выпускниками ВУЗа относятся к 1-му классу, а это значит, что возраст таких клиентов ниже среднего.

С помощью *GridSearchCV* я нашел лучший параметр для количества деревьев. Им стало значение 330.

Далее мною было выполнено построение классификатора *Random Forest* и оценка его точности с помощью среднего значения четырёх метрик: *accuracy = 0.5340531879534091, f1 = 0.5752483135706458, precision = 0.5357495033098759* и *recall = 0.6318680043788645*. После сравнения метрик каждого из двух классификаторов я сделал вывод, что решающее дерево работает более точно, чем случайный лес, потому что в классификаторе *Random Forest* разделить данные удалось хуже чем в *Decision*

Tree, у которого значение метрики $f1 = 60\%$. Соответственно *Decision Tree* оказался более точным классификатором нежели *Random Forest*.

Точность обоих классификаторов оказалась низкой, потому что довольно сложно оценить возраст клиентов кредитной карты из-за малого количества признаков, которые могут оценить клиента, как человека.

Задача №5

Набор данных: *Данные быстрых свиданий*.

Необходимо провести анализ набора данных и сделать его обработку, соответствующую алгоритму решения задачи.

Для начала подключим необходимые библиотеки для анализа данных: *pintpy* (для эффективной работы с данными), *pandas* (для работы с наборами данных), *matplotlib.pyplot* (библиотека для визуализации), *seaborn* (для построения графиков). Далее считаем набор данных с устройства и преобразуем его в таблицу с помощью метода *pd* из библиотеки *pandas* в переменную *data*.

Теперь перейдем к первичному анализу данных.

- 1) Сколько в наборе данных объектов и признаков? Дать описание каждому признаку, если оно есть.

Чтобы выяснить, какое количество объектов и признаков в нашем наборе данных, воспользуемся методом *shape()*. После применения данного метода узнаём, что в нашем наборе данных 8378 объектов и 195 признаков.

Так как признаков очень много, выделим некоторое подмножество с 14-ю признаками, наиболее подходящими для анализа данных. Описание данных признаков представлено ниже:

1. *gender* - Пол участника (женский = 0, мужской = 1).
2. *dec* – Решение участника (Если да - 1, если нет, то 0).
3. *age* - Возраст участника.
4. *samerace* – Принадлежность общей расе.
5. *goal* - Основная цель участия в этом мероприятии.
6. *career_c* - Род деятельности.
7. *date* - Часто ли участник ходит на свидания?
8. *like* - В целом, насколько вам нравится этот человек?
9. *attr1_1* – Привлекательность (насколько важно это качество в партнере)
10. *sinc1_1* - Искренность (насколько важно это качество в партнере)
11. *intell1_1* - Интеллектуальность (насколько важно это качество в партнере)
12. *fun1_1* – Чувство юмора (насколько важно это качество в партнере)
13. *amb1_1* - Амбициозность (насколько важно это качество в партнере)
14. *shar1_1* – Общие интересы (насколько важно это качество в партнере)

Таблица 5.1. Первые 5 строк выбранного подмножества набора данных «Данные быстрых свиданий» (данную таблицу можно получить с помощью метода *head()*).

	gen der	sa- me- race	ag e	go al	da te	ca- reer_ c	attr1 _1	sinc1 _1	in- tel1_ 1	fun 1-1	Amb 1_1	Shar 1_1	de c	li ke
0	0	0	21 .0	2. 0	7. 0	NaN	15.0	20.0	20.0	15. 0	15.0	15.0	1	7. 0
1	0	0	21 .0	2. 0	7. 0	NaN	15.0	20.0	20.0	15. 0	15.0	15.0	1	7. 0

2	0	1	21 .0	2. 0	7. 0	NaN	15.0	20.0	20.0	15. 0	15.0	15.0	1	7. 0
3	0	0	21 .0	2. 0	7. 0	NaN	15.0	20.0	20.0	15. 0	15.0	15.0	1	7. 0
4	0	0	21 .0	2. 0	7. 0	NaN	15.0	20.0	20.0	15. 0	15.0	15.0	1	6. 0

2) Сколько категориальных признаков, какие?

Чтобы узнать какие признаки являются категориальными, применим метод *describe()*.

Таблица 5.2. Результат работы метода *describe()* на выбранном подмножестве.

	gen der	sa- me- race	age	goal	date	ca- reer _c	attr 1_1	sinc 1_1	in- tel1 _1	fun 1-1	Am b1_ 1	Sha r1_ 1	dec	like
c o u n t	837 8.0 000 00	837 8.0 000 00	828 3.0 000 00	829 9.0 000 00	824 0.0 000 00	829 9.0 000 00	829 9.0 000 00	829 9.0 000 00	828 9.0 000 00	827 9.0 000 00	825 7.0 000 00	837 8.0 000 00	813 8.0 000 00	
m e a n	0.5 005 97	0.3 957 99	26. 220 358 928	2.1 5.0 63	5.2 777 91	22. 514 632	17. 396 389	20. 265 613	17. 457 043	10. 682 539	11. 845 111	0.4 199 09	6.1 340 87	
st d	0.5 000 29	0.4 890 51	3.5 667 63	1.4 071 81	1.4 445 31	3.3 095 20	12. 587 674	7.0 467 00	6.7 830 03	6.0 852 39	6.1 248 88	6.3 621 54	0.4 935 73	1.8 412 85
m i n	0.0 000 00	0.0 000 00	18. 000 000	1.0 000 00	1.0 000 00	1.0 000 00	0.0 000 00	0.0 000 00						
2 5 %	0.0 000 00	0.0 000 00	24. 000 000	1.0 000 00	4.0 000 00	2.0 000 00	15. 000 000	15. 000 000	17. 390 000	15. 000 000	5.0 200 00	9.5 200 00	0.0 000 00	5.0 000 00
5 0 %	1.0 000 00	0.0 000 00	26. 000 000	2.0 000 00	5.0 000 00	6.0 000 00	20. 000 000	18. 180 000	20. 000 000	18. 000 000	10. 000 000	10. 640 000	0.0 000 00	6.0 000 00
7 5 %	1.0 000 00	1.0 000 00	28. 000 000	2.0 000 00	6.0 000 00	7.0 000 00	25. 000 000	20. 000 000	23. 810 000	20. 000 000	15. 000 000	16. 000 000	1.0 000 00	7.0 000 00
m a	1.0 000	1.0 000	55. 000	6.0 000	7.0 000	17. 000	100 .00	60. 000	50. 000	50. 000	53. 000	30. 000	1.0 000	10. 000

x	00	00	000	00	00	000	000	000	000	000	000	000	000	00	000
---	----	----	-----	----	----	-----	-----	-----	-----	-----	-----	-----	-----	----	-----

В выбранном подмножестве категориальным признаком является *career_c*, который описывает род деятельности участника (*1 = Lawyer; 2 = Academic/Research; 3 = Psychologist; 4 = Doctor/Medicine; 5 = Engineer; 6 = Creative Arts/Entertainment; 7 = Banking/Consulting/Finance/Marketing/Business/CEO/Entrepreneur/Admin; 8 = Real Estate; 9 = International/Humanitarian Affairs; 10 = Undecided; 11 = Social Work; 12 = Speech Pathology; 13 = Politics; 14 = Pro sports/Athletics; 15 = Other; 16 = Journalism; 17 = Architecture*).

Больше категориальных признаков в выбранном подмножестве нет.

- 3) Столбец с максимальным количеством уникальных значений категориального признака?

Так как в выбранном подмножестве только один признак – *career_c*, то он и будет столбцом с максимальным количеством уникальных значений категориального признака.

Из таблицы 5.2 замечаем, что максимальным значением в этом столбце является значение 17, значит всего категорий в *career_c* = 17.

- 4) Есть ли бинарные признаки?

Также из таблицы 5.2 можно увидеть бинарные признаки. У таких признаков максимальным значением будет 1, а минимальным значением 0.

В выбранном подмножестве бинарными признаками являются признаки:

1. *gender* - Пол участника (женский = 0, мужской = 1).
2. *dec* – Решение участника (Если да - 1, если нет, то 0).
3. *samerace* – Принадлежность общей расе.

- 5) Какие числовые признаки?

Из таблицы 5.2 можно увидеть числовые признаки. В выбранном подмножестве числовые признаки — это те признаки, которые не являются ни бинарными, ни категориальными.

1. *age* - Возраст участника.
2. *goal* - Основная цель участия в этом мероприятии.
3. *date* - Часто ли участник ходит на свидания?
4. *like* - В целом, насколько вам нравится этот человек?
5. *attr1_1* – Привлекательность (насколько важно это качество в партнере)
6. *sinc1_1* - Искренность (насколько важно это качество в партнере)
7. *intell1_1* - Интеллектуальность (насколько важно это качество в партнере)
8. *fun1_1* - Чувство юмора (насколько важно это качество в партнере)
9. *amb1_1* - Амбициозность (насколько важно это качество в партнере)
10. *shar1_1* – Общие интересы (насколько важно это качество в партнере)

6) Есть ли пропуски?

Чтобы узнать, есть ли пропуски в выбранном подмножестве, воспользуемся методом `info()`.

Таблица 5.3. Результат работы метода `info()` на выбранном подмножестве.

#	Column	Non-Null Count	Dtype
0	gender	8378 non-null	int64
1	samerace	8378 non-null	int64
2	age	8283 non-null	float64
3	goal	8299 non-null	float64
4	date	8281 non-null	float64
5	career_c	8240 non-null	float64
6	attr1_1	8299 non-null	float64
7	sinc1_1	8299 non-null	float64
8	intel1_1	8299 non-null	float64
9	fun1_1	8289 non-null	float64
10	amb1_1	8279 non-null	float64
11	shar1_1	8257 non-null	float64
12	dec	8378 non-null	int64
13	like	8138 non-null	float64

Из таблицы 5.3 можно увидеть, сколько непустых значений в каждом столбце.

Так как всего объектов - 8378, в столбцах, где значение количества непустых ячеек меньше, чем 8378, содержатся пропуски.

Таким образом в выбранном подмножестве содержатся пропуски в столбцах: `age`; `goal`; `date`; `career_c`; `attr1_1`; `sinc1_1`; `intel1_1`; `fun1_1`; `amb1_1`; `shar1_1`; `like`.

7) Сколько объектов с пропусками?

В выбранном подмножестве содержатся пропуски в столбцах: `age`; `goal`; `date`; `career_c`; `attr1_1`; `sinc1_1`; `intel1_1`; `fun1_1`; `amb1_1`; `shar1_1`; `like`.

- В `age` количество пустых значений: $8378 - 8283 = 95$;
- В `goal` количество пустых значений: $8378 - 8299 = 79$;
- В `date` количество пустых значений: $8378 - 8281 = 97$;
- В `career_c` количество пустых значений: $8378 - 8240 = 138$;
- В `attr1_1` количество пустых значений: $8378 - 8299 = 79$;
- В `sinc1_1` количество пустых значений: $8378 - 8299 = 79$;
- В `intel1_1` количество пустых значений: $8378 - 8299 = 79$;
- В `fun1_1` количество пустых значений: $8378 - 8289 = 89$;
- В `amb1_1` количество пустых значений: $8378 - 8279 = 99$;
- В `shar1_1` количество пустых значений: $8378 - 8257 = 121$;
- В `like` количество пустых значений: $8378 - 8138 = 240$.

Всего объектов с пропусками: 1195.

8) Столбец с максимальным количеством пропусков?

Столбцом с максимальным количеством пропусков является столбец *like*.

9) Есть ли на ваш взгляд выбросы, аномальные значения?

Чтобы ответить на этот вопрос, необходимо для начала нормализовать переменные числовых признаков выбранного подмножества через СКО, а затем пройтись по каждому значению каждого столбца и сравнить эти значения по модулю с 3.

```
data_subset_1 = data_subset

# Заменяем пустые значения на среднее значение столбцов
for col in data_subset_1.columns:
    data_subset_1[col].fillna(data_subset_1[col].mean(), inplace=True)

# Нормализация признаков через стандартное отклонение
from sklearn.preprocessing import StandardScaler

scale_features_std = StandardScaler()
features_std = scale_features_std.fit_transform(data_subset_1[['age', 'goal', 'date', 'like',
                                                               'attr1_1', 'sinc1_1', 'intell1_1', 'funl1_1',
                                                               'amb1_1', 'shar1_1']])

# Ищем выбросы (аномальные значения). Будем считать, что значение аномально, если его стандартное отклонение >= 3 по модулю.
for j in range(0, 10):
    for i in range(0, 8378):
        if abs(features_std[i][j]) >= 3:
            print(j, i)
```

Рисунок 5.1. Нормализация переменных каждого столбца через СКО. Поиск выбросов и аномальных значений.

После применения кода на рисунке 5.1, выясняем, что аномальные значения присутствуют в столбцах: *age*, *like*, *attr1_1*, *sinc1_1*, *intell1_1*, *funl1_1*, *amb1_1*.

10) Столбец с максимальным средним значением после нормировки признаков через стандартное отклонение?

Для нахождения столбца с максимальным средним значением после нормировки признаков через стандартное отклонение пробежимся по всем строкам столбца и посчитаем сумму всех значений. Затем поделим на количество измерений, а после сравним каждое среднее арифметическое столбцов.

```
for i in range(0, 10):
    s = 0
    for j in range(0, 8378):
        s += features_std[j][i]
    s = s / 8378
    print(i, s)
```

Рисунок 5.2. Поиск столбца с максимальным средним значением после нормировки признаков через стандартное отклонение.

Столбцом с максимальным средним значением после нормировки признаков через стандартное отклонение в выбранном подмножестве оказался столбец *fun1_1*. Его среднее значение после нормировки признаков через стандартное отклонение равно 6.261403427254553e-17.

11) Столбец с целевым признаком?

Столбцом с целевым признаком является столбец *dec* - *Решение участника* (*Если да - 1, если нет, то 0*).

12) Сколько объектов попадает в тренировочную выборку при использовании train test split с параметрами test_size = 0.3, random_state = 42?

Чтобы ответить на данный вопрос, необходимо для начала удалить строки с пустыми значениями из выбранного подмножества.

Далее необходимо выделить тренировочную и тестовую выборки с использованием параметров, данных в задании (*test_size = 0.3, random_state = 42*), и посмотреть, сколько объектов попадает в тренировочную выборку.

```
target = data_subset.dec # Выделяем целевую переменную
train = data_subset.drop(['dec'], axis=1)

from sklearn.model_selection import train_test_split

# Выделяем тренировочную и тестовую выборки с параметрами test_size = 0.3, random_state = 42

x_train, x_test, y_train, y_test = train_test_split(train, target, test_size = 0.3, random_state = 42)

N_train, _ = x_train.shape
N_test, _ = x_test.shape
print(N_train)
```

Рисунок 5.3. Выделение целевой переменной, тестовой и тренировочной выборки с использованием параметров, данных в задании (*test_size = 0.3, random_state = 42*).

Из результата кода на рисунке 5.3 мы получаем количество объектов, попадающих в тренировочную выборку.

В выбранном подмножестве количество объектов, попадающих в тренировочную выборку, равно 5551.

13) Между какими признаками наблюдается линейная зависимость (корреляция)?

Выявить корреляцию между признаками позволяет метод *corr()*. Чем ниже будет наше значение, тем сильнее между признаками будет заметна отрицательная корреляция (то есть с увеличением одной переменной, другая будет уменьшаться). Чем выше будет наше значение, тем сильнее будет заметна положительная корреляция (то есть с увеличением одной переменной, другая будет тоже увеличиваться).

Примечание. Для наглядности лучше вывести температурную таблицу корреляции признаков.



Рисунок 5.4. Температурная таблица корреляции признаков.

Из рисунка 5.4 можно выявить отрицательную и положительную корреляцию между признаками.

В выбранном подмножестве отрицательная корреляция наблюдается между признаками:

- amb1_1 и gender (Из определения данного выше, мы узнаем, что с увеличением значения амбициозности, значение пола будет уменьшаться. Это значит, что женщины придают большое внимание амбициозности мужского пола и наоборот.)

- *attr1_1 u shar1_1* (Из определения данного выше, мы узнаем, что с увеличением значения привлекательности, значение общих интересов будет уменьшаться. Это значит, что участники, отдавшие большее количество баллов привлекательности, отдают малое количество баллов общим интересам и наоборот.)
- *amb1_1 u attr1_1* (Из определения данного выше, мы узнаем, что с увеличением значения амбициозности, значение привлекательности будет уменьшаться. Это значит, что участники, отдавшие большее количество баллов привлекательности, отдают малое количество баллов амбициозности и наоборот.)
- *attr1_1 u sinc1_1* (Из определения данного выше, мы узнаем, что с увеличением значения привлекательности, значение искренности будет уменьшаться. Это значит, что участники, отдавшие большее количество баллов привлекательности, отдают малое количество баллов искренности и наоборот.)

Положительная корреляция в выбранном подмножестве наблюдается между признаками: *attr1_1 u gender* (Из определения данного выше, мы узнаем, что с увеличением значения привлекательности, значение пола будет увеличиваться. Это значит, что мужчинам очень важно, чтобы противоположный пол был привлекательным).

14) Сколько признаков достаточно для объяснения 90% дисперсии после применения метода PCA?

Выделим целевую переменную *dec*, тренировочную и тестовую выборки. Применим сам метод PCA на выбранном подмножестве.

Код применения метода PCA представлен на рисунке 5.5.

```
from sklearn.decomposition import PCA
%matplotlib inline
import matplotlib.pyplot as plt

pca = PCA()
pca.fit(x_train)
X_pca = pca.transform(x_train)

for i, component in enumerate(pca.components_):
    print("{} component: {}% of initial variance".format(i + 1, round(100 * pca.explained_variance_ratio_[i], 2)))
    print(" " .join(["%.3f x %s" % (value, name) for value, name in zip(component, train.columns)]))
```

Рисунок 5.5. Код применения метода PCA на тестовую выборку с целевым признаком *dec*.

Результат работы метода:

- 1 признак объясняет 17.62% дисперсии
- 2 признака объясняют 11.25% дисперсии
- 3 признака объясняют 10.26% дисперсии
- 4 признака объясняют 10.08% дисперсии
- 5 признаков объясняют 9.24% дисперсии
- 6 признаков объясняют 8.42% дисперсии
- 7 признаков объясняют 8.02% дисперсии
- 8 признаков объясняют 7.62% дисперсии

- 9 признаков объясняют 6.97% дисперсии
- 10 признаков объясняют 6.6% дисперсии

То есть десяти признаков хватит для объяснения 90% дисперсии после применения метода PCA.

Для наглядности покажем график зависимости доли объяснённой дисперсии от числа компонент.

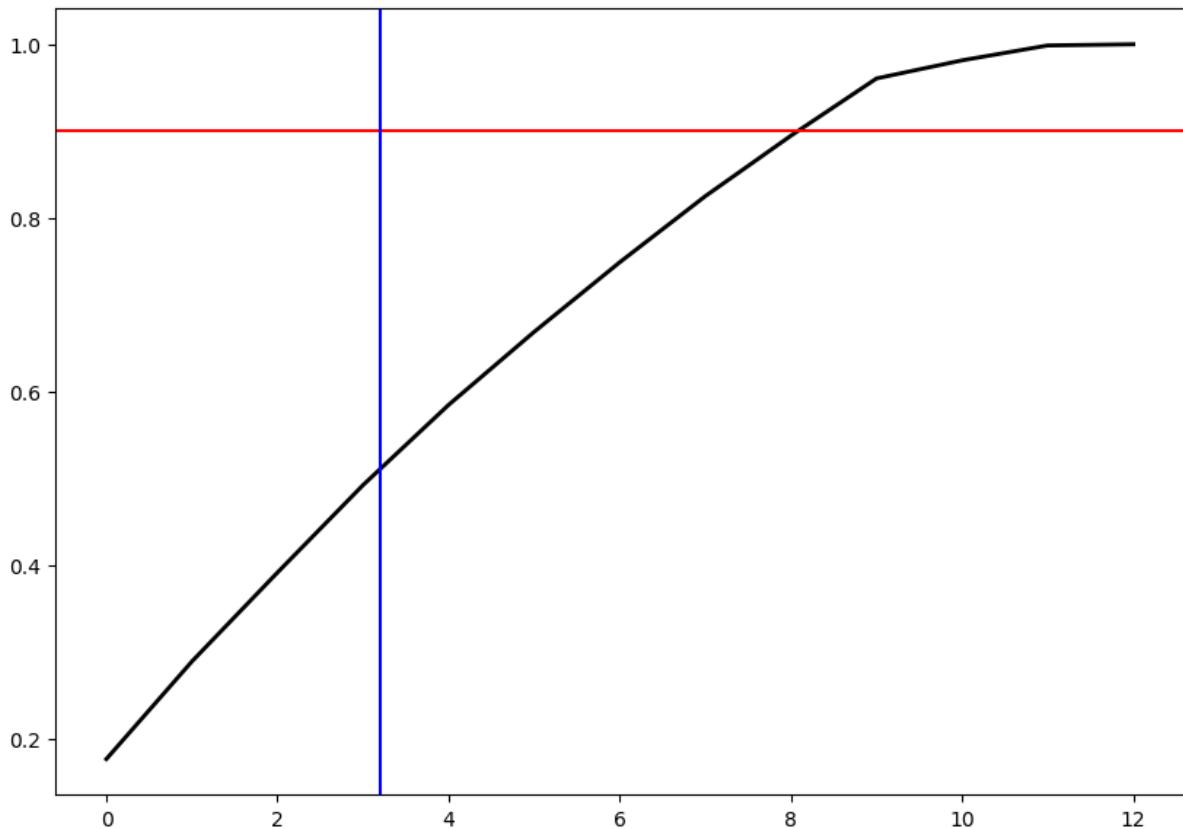


Рисунок 5.6. График зависимости доли объяснённой дисперсии от числа компонент.

15) Какой признак вносит наибольший вклад в первую компоненту?

Из результата кода на рисунке 5.5 мы узнали коэффициенты каждого признака и теперь можем оценить их вклад в первую компоненту.

Итак, посмотрим на распределение вклада признаков в первую компоненту:

$$0.148 \times gender + 0.015 \times samerace + 0.053 \times age + 0.105 \times goal + -0.242 \times date + 0.103 \times career_c + 0.641 \times attr1_1 + -0.328 \times sinc1_1 + -0.166 \times intell_1 + 0.057 \times fun1_1 + -0.436 \times amb1_1 + -0.392 \times shar1_1 + 0.041 \times like$$

Отсюда видно, что наибольшим коэффициентом является коэффициент 0.641 у признака *attr1_1*, а это значит, что привлекательность внесла наибольший вклад в первую компоненту.

- 16) Построить двухмерное представление данных с помощью алгоритма t-SNE. На сколько кластеров визуально, на ваш взгляд, разделяется выборка? Объяснить смысл кластеров.

Применим алгоритм t-SNE для двухмерного представления данных и посмотрим на сколько кластеров разделяется наша выборка.

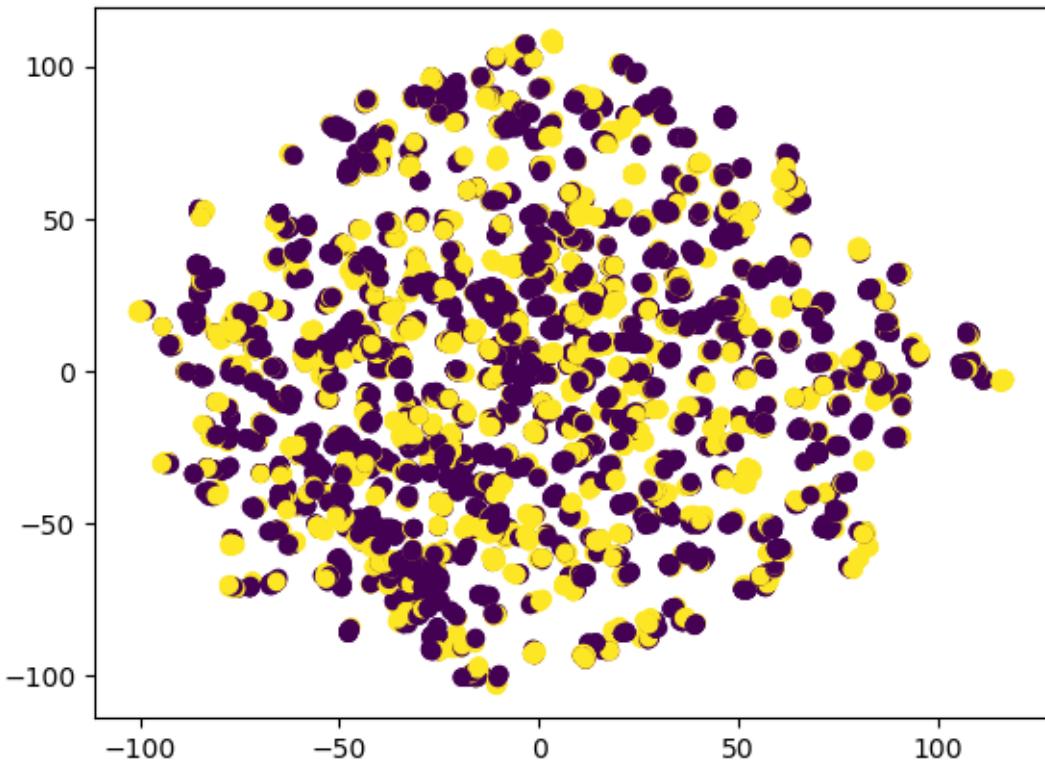


Рисунок 5.7. Двухмерное представление данных с помощью алгоритма t-SNE.

Из рисунка 5.7 видно, что кластеры не разделены на подгруппы четко, поэтому трудно сказать сколько их и объяснить их на взгляд. Для более детального рассмотрения кластеров необходимо применить метод DBSCAN или KMeans.

Вывод:

Мне нужно было провести первичный анализ набора данных – «Данные быстрых свиданий».

В начале мной было просмотрено количество объектов (8378) и признаков (195), а также их описание. Так как признаков оказалось слишком много, мне пришлось выбрать некое подмножество из 14 признаков (*gender*, *dec*, *age*, *samerace*, *goal*, *career_c*, *date*, *like*, *attr1_1*, *sinc1_1*, *intel1_1*, *fun1_1*, *amb1_1*, *shar1_1*), на которых я и проводил первичный анализ данных.

Далее мной были выявлены бинарные признаки (*gender*, *dec*, *samerace*), категориальный признак (*career_c*) и числовые признаки (*age*, *goal*, *date*, *like*, *attr1_1*, *sinc1_1*, *intel1_1*, *fun1_1*, *amb1_1*, *shar1_1*) в выбранном подмножестве. Я обнаружил, что в вы-

бранном подмножестве 1195 объектов с пропусками, а признаком с самым большим количеством пропусков является столбец *like*.

Далее мной были найдены аномальные значения в признаках *age*, *like*, *attr1_1*, *sinc1_1*, *intel1_1*, *fun1_1*, *amb1_1*. Следующим шагом был найден столбец с максимальным средним значением после нормировки признаков через стандартное отклонение в выбранном подмножестве - столбец *fun1_1*. Его среднее значение после нормировки признаков через стандартное отклонение равно 6.261403427254553e-17. После того, как я выбрал целевой признак выбранного подмножества – *dec*, мной было проведено исследование, какое количество объектов попадает в тренировочную выборку при использовании *train_test_split* с параметрами *test_size* = 0.3, *random_state* = 42.

Далее я выяснил между какими признаками наблюдается линейная зависимость (корреляция). Оказалось, что в выбранном подмножестве отрицательная корреляция наблюдается между признаками: *amb1_1 u gender*, *attr1_1 u shar1_1*, *amb1_1 u attr1_1 attr1_1 u sinc1_1*, а положительная корреляция в выбранном подмножестве наблюдается между признаками: *attr1_1 u gender*.

Также с помощью метода PCA я выяснил, сколько признаков достаточно для объяснения 90% дисперсии. Оказалось, что достаточно всего лишь 10 признаков. Было выявлено, что привлекательность внесла наибольший вклад в первую компоненту.

Далее я попробовал построить двухмерное представление данных с помощью алгоритма t-SNE и визуально оценить, на сколько кластеры разделяются выборка. К сожалению, кластеры не разделены на подгруппы четко, поэтому трудно сказать сколько их и объяснить их на взгляд. Для более детального рассмотрения кластеров необходимо применить метод DBSCAN или KMeans.

Задача №6

Набор данных: *Данные быстрых свиданий*.

Необходимо провести свободный анализ набора данных.

Перед тем, как приступить к свободному анализу данных подключим необходимые библиотеки по аналогии с задачей №5:

- *import numpy as np* (библиотека для эффективной работы с данными)
- *import pandas as pd* (библиотека для работы с наборами данных)
- *import matplotlib.pyplot as plt* (библиотека для визуализации)
- *import seaborn as sns* (еще одна библиотека для построения графиков)

Далее считаем набор данных из таблицы с помощью метода *pd* библиотеки *pandas* в переменную *data* и приступим к свободному анализу набора данных «Данные быстрых свиданий».

Конечно же, когда речь заходит о свиданиях, хочется узнать, какими качествами нужно обладать, чтобы понравиться другому человеку, а какими нет?

Поэтому начнем проводить свободный анализ набора данных о быстрых свиданиях с ответов на вопросы:

- 1) Какие наименее желательные качества женщины находят в мужчине?
- 2) Какие наиболее желательные качества женщины находят в мужчине?
- 3) Какие наименее желательные качества мужчины находят в женщине?
- 4) Какие наиболее желательные качества мужчины находят в женщине?

Для ответа на эти вопросы сначала необходимо выбрать подмножество из набора данных, которое будет содержать переменную, которая отвечает за пол участника – *gender* и переменные, отвечающие за качества человека.

Просмотрев описание признаков набора данных из документа Word, я сделал вывод, что существует 5 качеств, по которым участники эксперимента оценивали, какие качества для них более/менее важны в партнере.

Признаки, по которым была осуществлена оценка партнеров в разные этапы эксперимента:

- *attr* - привлекательность
- *sinc* - искренность
- *intel* - интеллект
- *fun* - веселье
- *amb* - амбициозность
- *shar* - общие интересы/увлечения

Замечание. Данные признаки в наборе данных с разными коэффициентами. Нас будут интересовать только те, которые описывают партнеров!

Это коэффициенты 1_3, 1_2, 1_s, 1_1.

Итак, после выбора признаков для подмножества необходимо удалить все строки с пустыми значениями, а после этого уже приступать к ответам на вопросы, сформулированные выше.

Для ответа на вопросы: **Какие наименее/наиболее желательные качества женщины находят в мужчине?** можно использовать группировку данных по признаку *gender* и вычислить средние значения остальных признаков. Затем можно отсортировать признаки по убыванию средних значений для женщин и выбрать наименьшие.

Такой подход будет верным, так как значения признаков присваиваются участниками баллах. То есть чем меньше среднее по признаку, тем менее важен этот признак партнеру.

Код данного алгоритма показан на рисунке 6.1.

```
# Подмножество женщин
woman_subset = data_quality.loc[data_quality['gender'] == 0]

# Вычисляем средние значения остальных признаков
mean_values = woman_subset.mean()

# Сортируем признаки по убыванию средних значений
sorted_values = mean_values.sort_values()

# Выбираем наименьшие значения
least_desirable_qualities = sorted_values[1:]

print(least_desirable_qualities)
```

Рисунок 6.1. Код выбора наименее желательных качеств для женщин в мужчинах.

Таблица 6.1. Результат работы кода из рисунка 6.1.

amb1_s	12.904916
amb1_2	13.056389
amb1_1	13.138389
amb1_3	13.153589
shar1_1	13.646916
shar1_2	13.955095
shar1_s	13.997211
shar1_3	14.908358
fun1_s	14.915411
fun1_3	15.439368
fun1_2	15.992211
fun1_1	16.773516

sinc1_s	17.110874
sinc1_3	17.597947
attr1_1	17.881221
sinc1_2	18.165884
sinc1_1	18.653137
intel1_s	18.752853
intel1_3	18.995579
attr1_s	19.211747
intel1_2	19.260200
attr1_3	19.750463
intel1_1	19.907705
attr1_2	20.030105

Из таблицы 6.1 видим, что девушки (женщины) считают наименее желательным качеством - амбициозность в парнях (мужчинах). Следующим менее важным (желательным) качеством в парне (мужчине) девушки (женщины) считают **общие интересы и увлечения**.

Важными качествами для девушки (женщины) в мужчине является **уровень интеллекта и привлекательность**.

Выполним такой же алгоритм для ответа на вопросы: **Какие наименее/наиболее желательные качества мужчины находят в женщине?**

Код этого алгоритма показан на рисунке 6.2.

```
# Подмножество мужчин
male_subset = data_quality.loc[data_quality['gender'] == 1]

# Вычисляем средние значения остальных признаков
mean_values = male_subset.mean()

# Сортируем признаки по убыванию средних значений
sorted_values = mean_values.sort_values()

# Выбираем наименьшие значения
least_desirable_qualities = sorted_values[1:]

print(least_desirable_qualities)
```

Рисунок 6.2. Код выбора наименее желательных качеств для мужчин в женщинах.

Таблица 6.2. Результат работы кода из рисунка 6.2.

amb1_2	9.360463
--------	----------

amb1_1	9.785046
amb1_s	9.867292
amb1_3	10.367222
shar1_1	10.552755
shar1_s	10.647894
shar1_2	10.863218
shar1_3	11.467326
sinc1_s	14.664664
sinc1_2	16.426354
intel1_s	16.487373
fun1_3	16.843299
fun1_s	16.874410
sinc1_1	17.036100
sinc1_3	17.383148
fun1_2	18.297141
intel1_3	18.337928
intel1_2	18.419618
intel1_1	18.716771
fun1_1	19.360127
attr1_1	24.550012
attr1_s	25.206447
attr1_2	26.321262
attr1_3	26.505417

Из таблицы 6.2 видим, что парни (мужчины) считают наименее желательным качеством в девушках (женщинах) - **амбициозность**. Следующим менее важным (желательным) качеством в девушках (женщинах) парни (мужчины) считают **общие интересы и увлечения**, также как и девушки (женщины) в мужчине.

Делаем вывод, что женское мнения от мужского по части наименее желательных качеств в партнере не отличается.

А важными качествами для парня (мужчины) в девушке является **привлекательность и чувство юмора**.

Следующий вопрос, который возник при анализе набора данных «Данные быстрых свиданий»: Являются ли общие интересы более важными, чем общее расовое происхождение?

Для ответа на этот вопрос выделим новое подмножество из исходного набора данных, в которое возьмем признаки:

- *dec* – целевой признак, означающий за решение участника.
- *samerace* – признак, отвечающий за принадлежность общей расе.
- *shar1_1, shar1_s, shar1_2, shar1_3* – признаки, отвечающие за общие интересы в разные волны экспериментов.

После выбора признаков для подмножества необходимо удалить все строки с пустыми значениями.

Проведем исследование, построив линейную регрессию, где целевая переменная будет объясняемой переменной, а ее регрессорами остальные признаки в подмножестве.

Код выделения тестовой и тренировочной выборки, построения линейной регрессии, просмотра ее коэффициентов и выявления точности данной модели показан на рисунке 6.3.

```
from sklearn.model_selection import train_test_split
target = data_race_or_shar['dec']
data_sel = data_race_or_shar.drop(['dec'], axis = 1)

x_train, x_test, y_train, y_test = train_test_split(data_sel, target, test_size= 0.3, random_state=4477)

from sklearn.linear_model import LinearRegression
linmodel = LinearRegression()
model = linmodel.fit(x_train,y_train)

# Так как значения целевой переменной 0 и 1, а линейная регрессия,
# не предназначена для разбиения на два класса, делаем это самостоятельно.
# Все, что больше 0.5 = 1, что меньше = 0.

from sklearn import metrics

def linear_scorer(estimator, x, y):
    scorer_predictions = estimator.predict(x)

    scorer_predictions[scorer_predictions > 0.5] = 1
    scorer_predictions[scorer_predictions <= 0.5] = 0

    return metrics.accuracy_score(y, scorer_predictions)

linear_scorer(linmodel, x_test, y_test) # Выявление точности модели
print(model.coef_) # Просмотр коэффициентов линейной регрессии
```

Рисунок 6.3. Код построения линейной регрессии, оценки ее точности по коэффициенту детерминации R^2 (accuracy), просмотра коэффициентов линейной регрессии.

Результатом выполнения кода на рисунке 6.3 является массив из коэффициентов линейной регрессии:

[-0.01810194, -0.00292448, -0.00015589, 0.00585048, 0.00377909], где первый коэффициент – коэффициент samerace, а остальные коэффициенты – коэффициенты переменных, отвечающих за общие интересы и увлечения.

Точность у данной линейной регрессии хорошая для данного подмножества, а именно 53 %, значит зависимость между объясняемой переменной и регрессорами присутствует.

Оценивая коэффициенты данной модели можем сказать, что принадлежность разной расе (участника и его партнера) вследствие низкого отрицательного коэффициента очень влияет на решение участника (в отрицательную сторону) в отличие от общих интересов партнеров.

Делаем вывод, что общее расовое происхождение важнее участникам, чем их общие интересы.

Теперь произведем **кластеризацию** на всем наборе данных, но для начала приведем его к рабочему виду.

К сожалению, при удалении всех строк с пустыми значениями в нашем наборе данных не останется ни одной переменной. С помощью метода *info()* я узнал, что в выборке существуют столбцы, в которых более 4000 пропусков (более половины от всех объектов в столбце), поэтому мной было принято решение удалить все столбцы, в которых более 4000 пропусков, а потом уже удалять строки с пустыми значениями.

Замечание. При замене пустых значений на их среднее по столбцу набор данных будет невозможно разбить на кластеры, поэтому заменять пустые значения на их среднее по столбцу нельзя!

Код по обработке набора данных показан на рисунке 6.4.

```
data = data.drop(['match', 'dec_o', 'iid', 'id', 'field', 'undergra', 'mn_sat', 'tuition', 'from',
                  'zipcode', 'income', 'career'], axis=1)

# подсчет количества пропущенных значений в каждом столбце
missing_values_count = data.isna().sum()

# получение списка столбцов, в которых количество пропущенных значений больше 4000
columns_to_drop = list(missing_values_count[missing_values_count > 4000].index)

# удаление столбцов из таблицы
data = data.drop(columns=columns_to_drop)

# удаление всех пустых значений в каждом столбце
data = data.dropna()

# вывод обновленной таблицы
print(data)
```

Рисунок 6.4. Код обработки набора данных «Данные быстрых свиданий».

После обработки набора данных попробуем воспользоваться методом **PCA** для снижения размерности. Код построения метода *PCA* и выявление, сколькими признаками описывается 90% дисперсии показан на рисунке 6.5.

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

for i in range(data.shape[1] + 1):
    pca2 = PCA(n_components = i)
    pca2.fit(data)
    print(i, sum(pca2.explained_variance_ratio_))
```

Рисунок 6.5. Код построения метода *PCA* для снижения размерности.

Таблица 6.3. Первые 13 результатов работы кода из рисунка 6.5.

0	0
1	0.7403371097105373
2	0.8020180173327598
3	0.8278988959196985
4	0.8432684421386915
5	0.8560703590291434
6	0.8686468486050883
7	0.8798351627094184
8	0.8906416686456051
9	0.8989993922768336
10	0.906297769748573
11	0.9119444626572055
12	0.9174113989612865

Из таблицы 6.3 заметим, что 90% дисперсии объясняется 10-ю компонентами.

Визуализация метода *PCA* для снижения размерности на нашем наборе данных показана на рисунке 6.6.

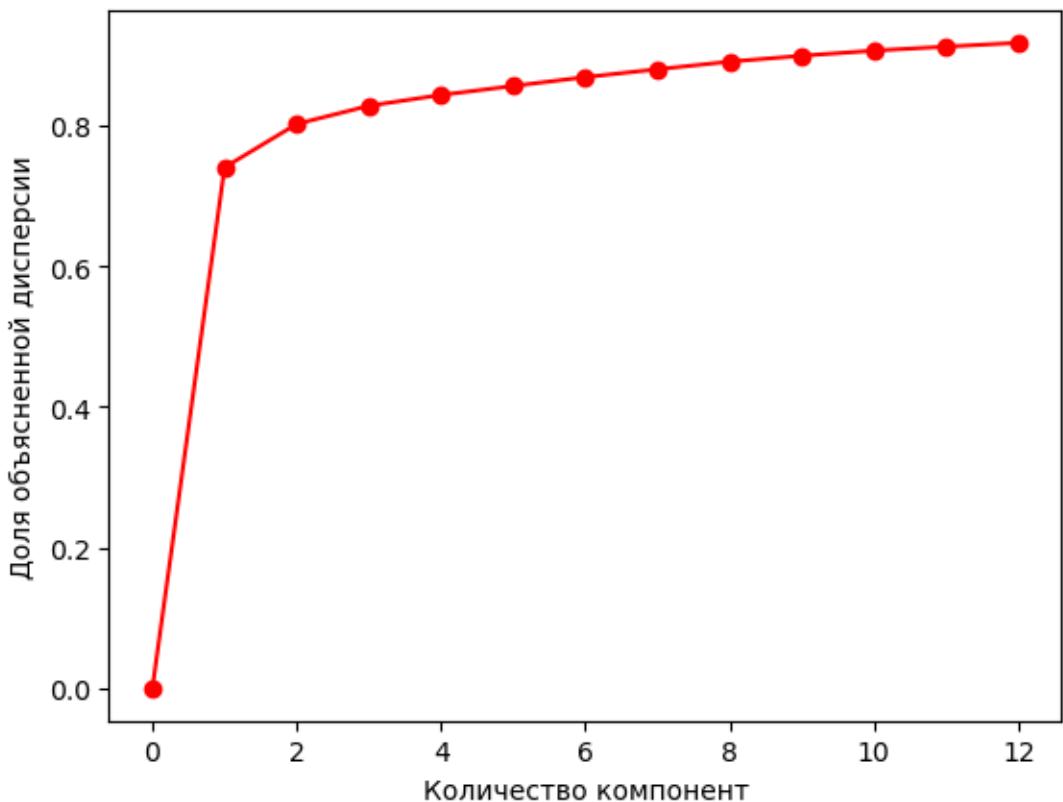


Рисунок 6.6. Визуализация метода *PCA* для снижения размерности на наборе данных «Данные быстрых свиданий».

Итак, теперь мы знаем, что можем снизить размерность до 10 компонент.

Теперь применим алгоритмы кластеризации для анализа набора данных «Данные быстрых свиданий».

Чтобы приступить к кластеризации, необходимо для начала нормализовать объекты данных через СКО, чтобы снизить масштаб разброса объектов для устойчивой и сбалансированной кластеризации.

После того, как мы нормализовали объекты данных, можем приступать к задаче кластеризации.

Выполним кластеризацию данных с использованием алгоритма *DBSCAN* и визуализируем полученные кластеры с помощью метода *t-SNE*:

- Для начала снизим размерность данных до 10 компонент, так как для описания 90% дисперсии больше компонент не понадобится.
- Далее создадим экземпляр класса *DBSCAN* с указанием размера окрестности, в которой ищутся соседние точки – $eps = 1$ и минимального количества точек, которые должны находиться в этой окрестности для образования кластера – $min_samples = 5$.
- После снизим размерность с помощью *t-SNE* и визуализируем кластеры.

Код выявления кластеров с использованием алгоритма *DBSCAN* и визуализации полученных кластеров с помощью метода *t-SNE* показан на рисунке 6.7.

```
from sklearn.cluster import DBSCAN
from sklearn.manifold import TSNE

#снизим размерность данных при помощи метода главных компонент
pca = PCA(n_components=10).fit(data)
pdata = pca.transform(data)

#используем dbscan для обнаружения кластеров
dbscan = DBSCAN(eps = 1, min_samples=5)
dbscan.fit(pdata)
labels = dbscan.labels_

#используем tsne для снижения размерности
tsne = TSNE()
pca_2d = tsne.fit_transform(pdata)

#визуализируем кластеры
for label in set(labels):
    if label == -1:
        plt.scatter(pca_2d[labels == label, 0], pca_2d[labels == label, 1], color='blue')
    else:
        plt.scatter(pca_2d[labels == label, 0], pca_2d[labels == label, 1], color=plt.cm.jet(label / np.max(labels + 1)))

plt.title('DBSCAN нашел {} кластеров и {} шумовых точек'.format(len(set(labels)) - (1 if -1 in labels else 0),
                                                               np.sum(labels == -1)))
plt.show()
```

Рисунок 6.7. Код выявления кластеров с использованием алгоритма *DBSCAN* и визуализации полученных кластеров с помощью метода *t-SNE*.



Рисунок 6.8. Визуализация кластеров при помощи метода *t-SNE* с использованием алгоритма *DBSCAN*.

Из рисунка 6.8 мы видим, что *DBSCAN* нашел 11 кластеров и 2654 шумовые точки, однако на графике не наблюдается четкого выделения кластеров.

Попробуем выполнить кластеризацию данных с использованием алгоритма *KMeans* и визуализируем полученные кластеры с помощью метода t-SNE.

```
from sklearn.cluster import KMeans
from sklearn.manifold import TSNE

kmeans = KMeans(n_init = 'auto', random_state = 157)
kmeans.fit(data)
labels = kmeans.labels_

tsne = TSNE()
pca_2d = tsne.fit_transform(data)

for label in set(labels):
    if label == -1:
        plt.scatter(pca_2d[labels == label, 0], pca_2d[labels == label, 1], color='blue')
    else:
        plt.scatter(pca_2d[labels == label, 0], pca_2d[labels == label, 1], color=plt.cm.jet(label / np.max(labels + 1)))

plt.title('kmeans нашел {} кластеров и {} шумовых точек'.format(len(set(labels)) - (1 if -1 in labels else 0),
                                                               np.sum(labels == -1)))
plt.show()
```

Рисунок 6.9. Код выявления кластеров с использованием алгоритма *KMeans* и визуализации полученных кластеров с помощью *t-SNE*.

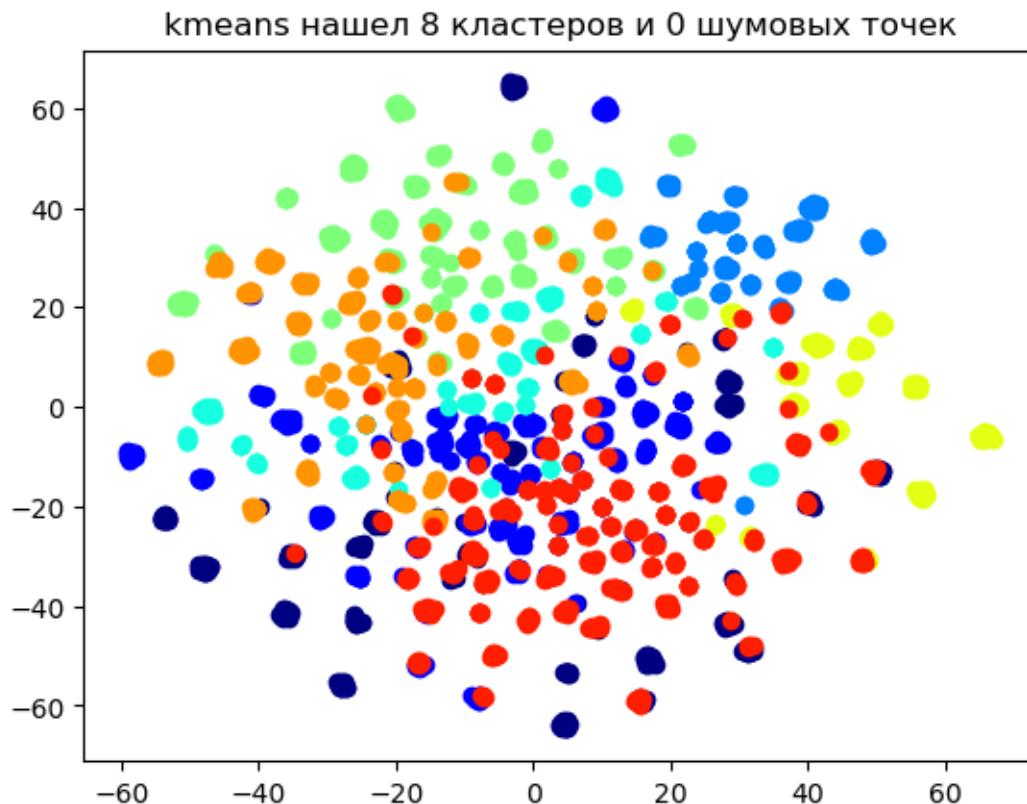


Рисунок 6.10. Визуализация кластеров при помощи метода t-SNE с использованием алгоритма *KMeans*.

Из рисунка 6.10 мы видим, что алгоритм кластеризации *K-means* справился лучше с выделением кластеров в данных, чем алгоритм *DBSCAN*. *K-means* обнаружил 8 кластеров и показал их более четко на графике. С другой стороны, *DBSCAN* обнаружил 11 кластеров, но на графике не наблюдается явного выделения кластеров, и также было обнаружено 2654 шумовых точек.

В целом, *K-means* показал более точные результаты в задаче кластеризации данного набора данных по сравнению с *DBSCAN*.

Однако четкого разделения на группы ни *KMeans*, ни *DBSCAN* в нашем наборе данных не выделил. Скорее всего кластеры четко не выделяются, потому что в нашем наборе данных слишком много пустых значений (так как каждый участник имел хотя бы одно пустое значение). Поэтому при удалении столбцов со многими пропусками, участников стало сложно группировать по каким-либо группам.

Попробуем применить классификатор – Decision Tree (решающее дерево), чтобы построить модель предсказания на основе дерева решений.

Так как для применения классификатора *Decision Tree (решающее дерево)*, нормализацию переменных делать не нужно, стоит вернуться к обработанному набору данных без нормализации объектов.

Далее выделим тренировочную и тестовую выборки, а затем найдем лучшую высоту дерева. Код поиска лучшей высоты дерева представлен на рисунке 6.11.

```
1 from sklearn.model_selection import train_test_split
2
3 target = data.dec
4 data_sel = data.drop('dec', axis = 1)
5
6 x_train, x_test, y_train, y_test = train_test_split(data_sel, target, test_size= 0.3, random_state=4477)
```

Найдем лучшую высоту дерева.

```
1 from sklearn.model_selection import train_test_split
2 from sklearn.tree import DecisionTreeClassifier
3
4 #используем решающее дерево
5 for i in range(1, 20):
6     T = DecisionTreeClassifier(random_state=338, max_depth = i)
7     T = T.fit(x_train, y_train)
8     print(str(i) + ": " + str(T.score(x_test, y_test)))
```

Рисунок 6.11. Код выделения тестовой и тренировочной выборки, поиска лучшей высоты решающего дерева.

В результате кода на рисунке 6.11 лучшей высотой дерева стало *Decision Tree* с высотой 3, так как последующие увеличения высоты ухудшают точность классификатора по метрике *accuracy*.

Далее посмотрим на точность нашего классификатора, оценив его по метрикам *accuracy*, *recall*, *precision*, *f1*, и сделаем вывод, можем ли мы доверять данному классификатору или нет.

Таблица 6.4. Точность классификатора *DecisionTree* (*Решающее дерево*) с высотой 3.

	precision	recall	f1-score	support
0	0.79	0.85	0.82	465
1	0.78	0.70	0.74	352
accuracy			0.78	817
macro avg	0.78	0.77	0.78	817
weighted avg	0.78	0.78	0.78	817

Так как значения всех метрик высокие, делаем вывод, что решающее дерево довольно точно предсказывает, на чем основано решение участников, и разделяет нашу выборку на 2 класса.

С помощью визуального просмотра классификатора и анализа вопросов, которые задает дерево, оценим, какие признаки влияют в большей мере на решение участников.

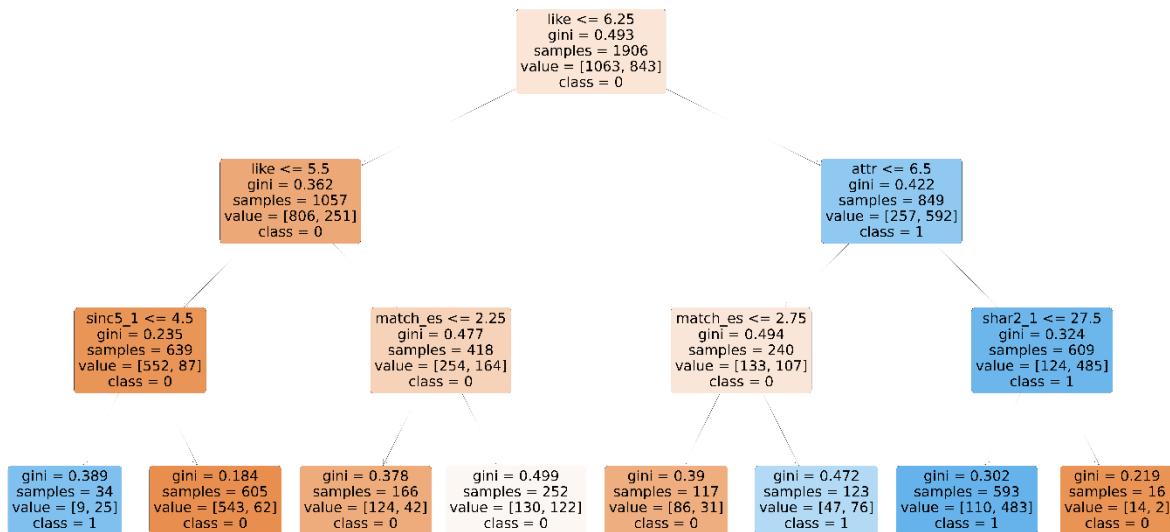


Рисунок 6.12. Визуализация классификатора *Decision Tree* (*Решающее дерево*) с высотой 3.

Опишем характеристики некоторых объектов класса 1:

like > 6.25, attr > 6.5, shar2_1 <= 27.5

Опишем характеристики некоторых объектов класса 0:

like <= 6.25, sinc5_1 > 4.5

Таким образом, участники, которые получили положительный ответ от своего партнера, должны нравиться своему партнеру (потому что признак *like*, который имеет значения от 0 до 10, должен иметь оценку выше среднего), должны быть привлекательными (потому что большое внимание уделяется признаку *attr*), а также их общие интересы могут совпадать (но при этом данному признаку уделяется не так много внимания).

Участники, которые получили отрицательный ответ от своего партнера, должны не нравиться своему партнеру (потому что признак *like*, который имеет значения от 0 до 10, должен иметь оценку ниже среднего), а также должны требовать от партнера быть искренними (потому что большое внимание уделяется признаку *sinc*).

Вывод:

Мне нужно было провести свободный анализ набора данных – «Данные быстрых свиданий».

Я задался вопросами: Какие наименее/наиболее желательные качества женщины/мужчины находят в мужчине/женщине? Для ответа на данные вопросы мне пришлось выбрать новое подмножество, состоящее из качественных признаков человека, а также признака – *gender*. Так как качественные признаки были оценены в баллах, для ответа на данные вопросы мне нужно было всего лишь взять их среднее и сравнить их среднее по столбцам. После анализа данных, мною было выявлено, что девушки (женщины) считают наименее желательными качествами – **амбициозность** в парнях (мужчинах). Также менее важным (желательным) качеством в парне (мужчине) девушки (женщины) считают **общие интересы и увлечения**. Наоборот, важными качествами для девушки (женщины) в мужчине является **уровень интеллекта и привлекательность**. Парни (мужчины) же считают наименее желательным качеством в девушках (женщинах) - **амбициозность**. Также наименее важным (желательным) качеством в девушках (женщинах) парни (мужчины) считают **общие интересы и увлечения**, также как и девушки (женщины) в мужчине, а важными качествами для парня (мужчины) в девушке является **привлекательность и чувство юмора**.

Следующий вопрос, которым я задался при свободном анализе данных «Данные быстрых свиданий»: Являются ли общие интересы более важными, чем общее расовое происхождение? Для ответа на данный вопрос, мною была сделана линейная регрессия, где объясняемой переменной стала целевая переменная – *dec*, а регрессорами – признаки, описывающие общие интересы и увлечения в разные волны эксперимента, а также принадлежность общей расе. Была оценена точность этой модели (53 %), по которой был сделан вывод, что данной модели можно доверять, а также выявлены коэффициенты линейной регрессии. Самым большим по модулю стал коэффициент перед регрессором - *samerace* (общее расовое происхождение), а, значит, он является более важным для описывания целевой переменной – *dec*. Вследствие низкого отрицательного коэффициента был сделан вывод, что принадлежность разной расе (участника и его партнера) влияет на решение участника (в отрицательную сторону).

Далее была произведена кластеризация с помощью алгоритмов DBSCAN и KMeans, в ходе которой применялся также и метод снижения размерности (PCA). В результате работы метода снижения размерности (PCA) было выявлено, что 90 % дисперсии опи-

сывается с помощью 10 компонент, поэтому при выявлении кластеров размерность была снижена до 10. При решении задачи кластеризации было выявлено, что алгоритм кластеризации *K-means* справился лучше с выделением кластеров в данных, чем алгоритм *DBSCAN*. *K-means* обнаружил 8 кластеров и показал их более четко на графике. С другой стороны, *DBSCAN* обнаружил 11 кластеров, но на графике не наблюдается явного выделения кластеров, и также было обнаружено 2654 шумовых точек. Однако четкого разделения на группы ни *KMeans*, ни *DBSCAN* в нашем наборе данных не выделил. Скорее всего кластеры четко не выделяются, потому что в нашем наборе данных слишком много пустых значений (так как каждый участник имел хотя бы одно пустое значение). Поэтому при удалении столбцов со многими пропусками, участников стало сложно группировать по каким-либо группам.

Следующим шагом, который я сделал в свободном анализе набора данных «Данные быстрых свиданий», было построение классификатора *Decision Tree (Решающее дерево)*. Сначала я выявил лучшую высоту дерева, с помощью сравнения классификатора с разным количеством деревьев по метрике *accuracy*. Самой лучшей высотой решающего дерева стала высота – 3. Далее мной была проверена точность классификатора по метрикам *accuracy, f1, recall, precision*. Все значения данных метрик были больше 0.7, поэтому я сделал вывод, что классификатору можно верить.

Далее я визуализировал классификатор и проанализировал вопросы, которые задает дерево, а также оценил, какие признаки влияют в большей мере на решение участников. Мной был сделан вывод, что участники, которые получили положительный ответ от своего партнера, должны нравиться своему партнеру (потому что признак *like*, который имеет значения от 0 до 10, должен иметь оценку выше среднего), должны быть привлекательными (потому что большое внимание уделяется признаку *attr*), а также их общие интересы могут совпадать (но при этом данному признаку уделяется не так много внимания). Участники, которые получили отрицательный ответ от своего партнера, должны не нравиться своему партнеру (потому что признак *like*, который имеет значения от 0 до 10, должен иметь оценку ниже среднего), а также должны требовать от партнера быть искренними (потому что большое внимание уделяется признаку *sinc*).

Заключение

1. В задаче №1 мной были оценены средние значения, дисперсии и СКО переменных из набора данных Swiss. Также мной были построены две математические модели с одним регрессором и построены их графики. Для каждой из линейных регрессий была оценена точность по коэффициенту детерминации R^2 , зависимость объясняемой переменной от регрессора по количеству «звездочек», а также выявлены положительные/отрицательные связи между объясняемой переменной и её регрессорами.
2. В задаче №2 мной было построено большое количество линейных регрессий. Была выполнена проверка на отсутствие связи между регрессорами, с помощью коэффициента взаимности дисперсии и построения линейных регрессий, где объясняемой переменной стал один из регрессоров, а его объясняющими переменными остальные регрессоры. Далее в первоначальную модель были добавлены логарифмы регрессоров, степени от регрессоров и их попарное произведение. Одновременно с вводом функций в модель была выполнена проверка на линейную независимость от регрессоров. Мной была выделена лучшая линейная регрессия по доле объяснения данных набора Swiss. Для неё я определил доверительные интервалы для всех коэффициентов, а после рассчитал доверительный интервал для одного прогноза.
3. В задаче №3 мной был выбран список регрессоров, через которые мне было необходимо оценить, какие индивиды получают большую зарплату. Перед построением моделей мной были обработаны переменные, которые я взял для анализа заработной платы индивидов: из некоторых я сделал дамми-переменные, некоторые преобразовал в категориальные признаки, а все остальные нормализовал через СКО. В ходе решения задачи были построены более 40-ка линейных регрессий, в которые я вводил различные функции от регрессоров (логарифмы регрессоров, степени от регрессоров и их попарное произведение), убирал переменные, от которых ничего не зависит. После того как я нашёл самую лучшую модель по коэффициенту детерминации R^2 , а также зависимости объясняемой переменной от регрессоров (по количеству «звездочек»), мне пришлось оценить коэффициенты перед объясняющими переменными данной регрессии. Выводом стал список параметров о том, какие индивиды получают большую зарплату. После мной были выделены два подмножества, для них я описал, какие индивиды получают большую зарплату.
4. В задаче №4 мной были обработаны переменные типа object, был построен классификатор (Решающее дерево) для задачи классификации и была произведена оценка его точности с помощью метрик (accuracy, F1, recall и precision). С помощью данного классификатора я выделил границы классов, на которые Решающее дерево разделяет набор данных. Также мной был построен классификатор Случайный Лес для той же задачи и была произведена оценка его точности с использованием тех же метрик. В конце я сравнил два этих разных классификатора, и сделал вывод, какой из них лучший по точности разделения объектов на 2 класса (Решающее дерево оказалось лучше, так как все его метрики были выше, чем у Случайного леса).
5. В задаче №5 я провел достаточно подробный первичный анализ данных набора «Данные быстрых свиданий». Мной была выполнена проверка данного набора данных на наличие пропусков, я выявил количество признаков и изучил их описание, а также

нашел целевую переменную. Так как количество признаков, представленных в наборе данных было огромно, мной было выделено подмножество из признаков, которые описывали целевую переменную лучше всего, а после нашёл столбцы с аномальными выбросами и проверил признаки на корреляцию между собой. После обработки набора данных я провел исследование, сколько объектов попадает в тренировочную выборку при заданных параметрах. Конечно, мной был испробован метод РСА для снижения размерности и выявления главных компонент, в результате которого я снизил размерность до 10. Также я попробовал выделить кластеры с помощью алгоритма t-SNE, который, к сожалению, не смог преобразовать данные по группам. Я сделал вывод, что для выделения кластеров стоит попробовать другие алгоритмы.

6. В задаче №6 мной был выполнен свободный анализ данных набора «Данные быстрых свиданий». Для начала я ответил на интересующие меня вопросы, построив линейную регрессию и оценив коэффициенты каждого регрессора, а также выявил желательные и нежелательные качества в обоих полах с помощью выделения подмножеств и выявления средних значений баллов, которые участники эксперимента присуждали качествам, интересующих их больше всего. Далее был обработан весь набор данных, из которого я исключил признаки, имеющие более половины пустых значений. Из результата предыдущей задачи был сделан вывод, что алгоритм t-SNE не может выделить кластеры в нашем наборе данных, поэтому я применил алгоритмы KMeans и DBSCAN, предварительно снизив размерность с помощью РСА. К сожалению, данные алгоритмы также не дали желаемого результата, поэтому был сделан вывод, что в нашем наборе данных объекты не могут быть выделены в пределах кластеров. Далее мной был построен классификатор Решающее дерево, с помощью которого я выделил границы классов и описал важнейшие признаки, влияющие на решение участника (целевую переменную).

Список литературы

1. Introduction to Econometrics with R/Christoph Hanck, Martin Arnold, Alexander Gerber, Martin Schmelzer. - Essen, Germany: University of Duisburg-Essen, 2021.
2. Айвазян, С.А. Основы эконометрики/С.А. Айвазян, В.С. Мхитарян – Москва: Изд. объединение «ЮНИТИ», 1998. – 1005 с.
3. Вербик, М. Путеводитель по современной эконометрике/М. Вербик – Москва: «Научная книга», 2008. – 616 с.
4. Доугерти, К. Введение в эконометрику/К. Доугерти – Москва: ИНФРА-М, 2009. – 465 с.
5. Магнус, Я.Р. Эконометрика. Начальный курс/Я.Р. Магнус, П.К. Катышев, А.А. Пересецкий – Москва: Изд-во «ДЕЛО», 2004. – 576 с.

Приложения

Приложение 1

```
library ("lmtest")

data = swiss
help(swiss)

# Высчитывание среднего значения переменных Agriculture, Fertility, Education
data["Education1"] = (sum(data$Education)/47)
data["Fertility1"] = mean(data$Fertility)
data["Agriculture1"] = mean(data$Agriculture)

# Высчитывание дисперсии переменных Agriculture, Fertility, Education
data["Education2"] = var(data$Education)
data["Fertility2"] = var(data$Fertility)
data["Agriculture2"] = var(data$Agriculture)

# Высчитывание СКО переменных Agriculture, Fertility, Education
data["Education3"] = sqrt(var(data$Education))
data["Fertility2"] = sqrt(var(data$Fertility))
data["Agriculture2"] = sqrt(var(data$Agriculture))

# Построение модели вида  $y = a + bx$ , где  $y$  - Education,  $x$  - Agriculture
model1 = lm(Education~Agriculture, data)
model1
summary(model1)

# Команда summary(model) позволяет нам увидеть коэффициенты  $a$ ,  $b$ , для того
# чтобы составить модель вида  $y = a + bx$ . Здесь  $a = 24.69$  (Intercept),
#  $b = -0.27$ (Agriculture).  $y = 24.69 - 0.27x$ 
```

```

plot(swiss) + abline(a = 24.69, b = -0.27, col = "red")

# Оценка модели  $y = a + bx$ , где  $y$  - Education,  $x$  - Agriculture по коэффициенту детерминации:

# Так как мы описываем Education, всего одним параметром (Agriculture), то коэффициент детерминации очевидно вряд ли может быть высоким. В нашем случае  $R^2 =$  около 40 %. Если брать в расчет, что Education описывается лишь одним параметром, то модель не такая уж и плохая.

# Существует ли взаимосвязь между объясняемой переменной (Education) и объясняющей (Agriculture)?

# Определенно есть, ведь по вероятности, а именно по большому кол-ву звездочек у #регрессора, мы можем сказать, что уровень образования довольно сильно зависит от #образа жизни (Agriculture).

#-----#
# Построение модели вида  $y = a + bx$ , где  $y$  - Education,  $x$  - Fertility

model2 = lm(Education~Fertility, data)
model2
summary(model2)

# Команда summary(model2) позволяет нам увидеть коэффициенты  $a$ ,  $b$ , для того
# чтобы составить модель вида  $y = a + bx$ . Здесь  $a = 46.82$  (Intercept),
#  $b = -0.51$ (Fertility).  $y = 46.82 - 0.51x$ 

plot(swiss) + abline(a = 46.82, b = -0.51, col = "green")

# Оценка модели  $y = a + bx$ , где  $y$  - Education,  $x$  - Fertility по коэффициенту детерминации:

# Так как мы описываем Education, всего одним параметром (Fertility), то коэффициент #детерминации очевидно вряд ли может быть высоким. В нашем случае  $R^2 =$  около

```

#43 - 44 %. Если брать в расчет, что Education описывается лишь одним параметром, то модель не такая уж и плохая.

Существует ли взаимосвязь между объясняемой переменной (Education) и объясняющей (Fertility)?

Определенно есть, ведь по вероятности, а именно по большому кол-ву звездочек у регрессора, мы можем сказать, что уровень образования довольно сильно зависит от коэффициента рождаемости (Fertility).

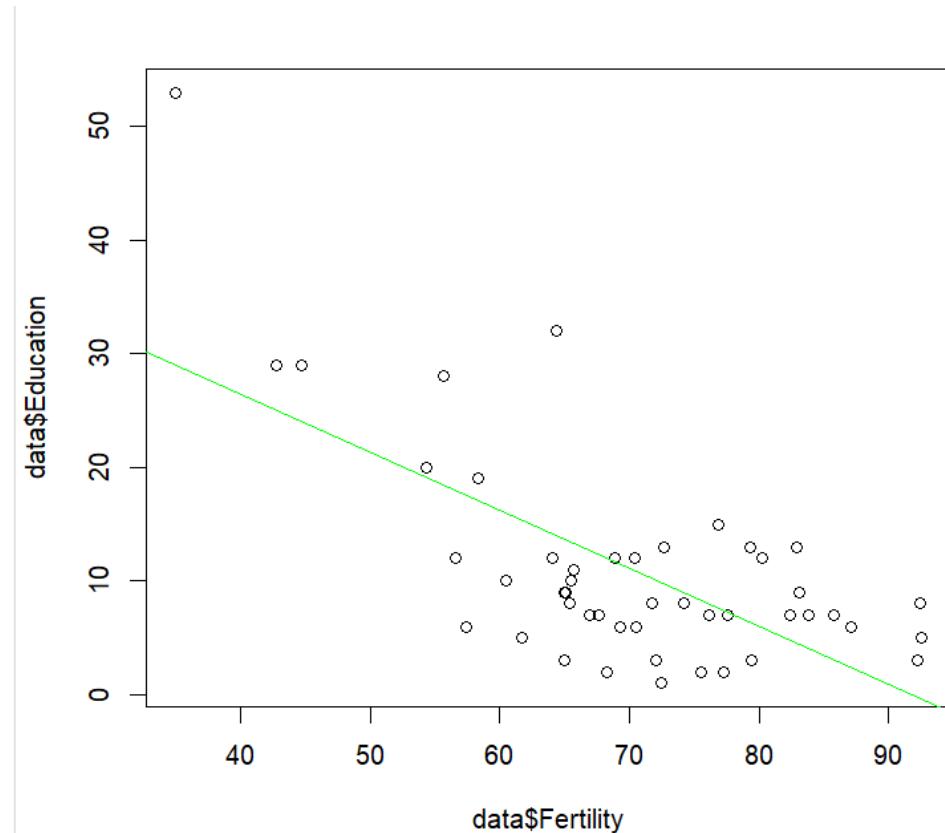


Рисунок 1.1. График линейной регрессии, где объясняемая переменная - Education, а регрессор – Agriculture.

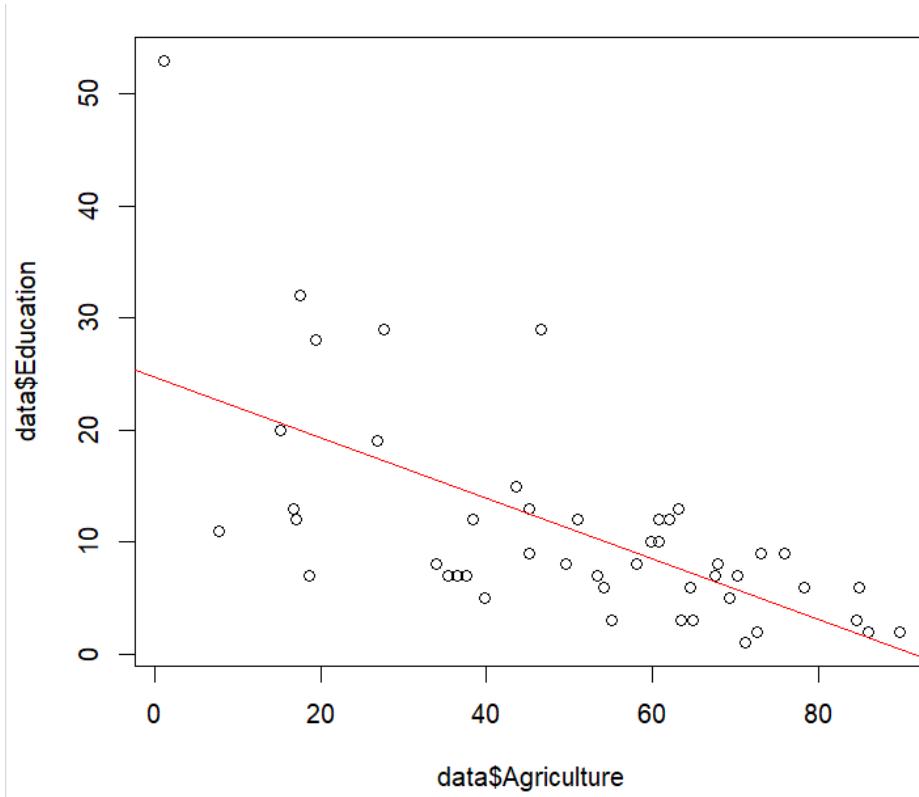


Рисунок 1.2. График линейной регрессии, где объясняемая переменная - Education, а регрессор - Fertility.

Приложение 2

```
library("lmtest")
```

```
library("GGally")
```

```
library("car")
```

```
data = na.omit(swiss)
```

```
data
```

```
summary(data)
```

```
ggpairs(data)
```

```
#-----Корректировка исходной зависимости-----
```

```
# Исходная зависимость
```

```
model_main = lm(Infant.Mortality~Education+Agriculture+Catholic, data)
```

```
summary(model_main)
```

```
# Мы видим, что в исходной модели очень низкий коэффициент детерминации( $R^2$ ), а  
#также нет зависимости между регрессорами и объясняемой переменной, поэтому  
# дальнийшие улучшения модели не приведут к желаемому результату.
```

```
# Чтобы улучшить модель введем новый регрессор Fertility.
```

```
model_corrected = lm(Infant.Mortality~Education+Agriculture+Catholic+Fertility, data)
```

```
summary(model_corrected)
```

```
# Наша модель стала намного лучше. Это заметно, ведь коэффициент #детермина-  
ции( $R^2$ ) вырос в несколько раз, став около 17%, а также появились #звездочки, что  
свидетельствует о зависимости между объясняемой и объясняющими #переменными.
```

```
#-----Проверка регрессоров на линейную зависимость-----
```

```
# Проверим, нет ли в наборе данных линейной зависимости.
```

```
# Для этого построим зависимости между регрессорами и увидим,
```

```
# есть ли между ними зависимость с помощью коэффициента детерминации( $R^2$ ).
```

```
model01 = lm(Agriculture~Catholic+Fertility+Education, data)
```

```
model02 = lm(Catholic~Agriculture+Fertility+Education, data)
```

```
model03 = lm(Fertility~Agriculture+Catholic+Education, data)
```

```
model04 = lm(Education~Agriculture+Catholic+Fertility, data)
```

```
summary(model01)
```

```
# Коэффициент детерминации ( $R^2$ ) около 55 %.
```

```
summary(model02)
```

```
#  $R^2$  около 42 %.
```

```
summary(model03)
```

```
# R^2 около 62 %.
```

```
summary(model04)
```

```
# R^2 около 70 %.
```

```
# Мы видим, что R^2 у всех регрессоров огромен. Однако, больше всего он наблюдается у регрессора Education.
```

```
# Его R^2 = 70%, а кол-во звездочек у каждого регрессора большое,
```

```
# а значит он линейно зависим от остальных регрессоров.
```

```
# Исключим его из откорректированной модели и посмотрим на R^2.
```

```
model_working = lm(Infant.Mortality~Agriculture+Catholic+Fertility, data)
```

```
summary(model_working)
```

```
# Видим, что R^2 у нашей модели стал даже лучше без регрессора Education, поэтому будем работать с этой моделью.
```

```
# Проверим, нет ли в наборе данных линейной зависимости уже без регрессора Education.
```

```
# Для этого построим зависимости между регрессорами и увидим,
```

```
# есть ли между ними зависимость с помощью коэффициента детерминации(R^2).
```

```
# Также проверим это с помощью VIF.
```

```
model1 = lm(Agriculture~Catholic+Fertility, data)
```

```
model2 = lm(Catholic~Agriculture+Fertility, data)
```

```
model3 = lm(Fertility~Agriculture+Catholic, data)
```

```
summary(model1)
```

```
# Коэффициент детерминации (R^2) около 16 %, что свидетельствует о линейной независимости регрессора
```

```
# Agriculture от других регрессоров.
```

```
summary(model2)
```

```
# R^2 около 24 %, что свидетельствует о линейной независимости регрессора
```

```
# Catholic от других регрессоров.  
summary(model3)  
# R^2 около 21 %, что свидетельствует о линейной независимости регрессора  
# Fertility от других регрессоров.  
  
# Теперь с помощью VIF удостоверимся в верности наших рассуждений.  
vif(model_working)  
  
# Видим, что наши регрессоры действительно линейно независимы друг от друга, так  
как значения из команды VIF у всех регрессоров меньше 1,4.  
  
#-----Проверка модели по методу наименьших квадратов-----  
# В корректировке модели мы уже проводили некоторые оценки исходной модели, по-  
этому оценим откорректированную рабочую модель.  
  
# Посмотрим, как будет меняться R^2 при добавлении регрессоров в модель.  
  
model_working1 = lm(Infant.Mortality~Fertility, data)  
summary(model_working1)  
  
# Видим, что R^2 у модели будет около 15 %, а кол-во звезд у регрессора будет нема-  
лое.  
  
model_working2 = lm(Infant.Mortality~Fertility+Agriculture, data)  
summary(model_working2)  
  
# Видим, что R^2 у модели увеличился и стал почти 19 %, однако звезд у добавляемого  
регрессора вообще нет.  
  
model_working3 = lm(Infant.Mortality~Fertility+Agriculture+Catholic, data)  
summary(model_working3)
```

```
# Видим, что R^2 у модели при добавлении регрессора Catholic упал на 2 %, а значит  
#стоит исключить его из нашей модели, ведь объясняемая переменная никак не #зависит от данного регрессора.
```

```
# Получаем хорошую модель
```

```
model_good = lm(Infant.Mortality~Fertility+Agriculture, data)
```

```
#-----Ввод логарифмов от регрессоров в модель-----
```

```
# Для начала исследуем регрессоры на линейную зависимость от исходных.
```

```
model001 = lm(log(Fertility)~Fertility, data)
```

```
model002 = lm(log(Agriculture)~Agriculture, data)
```

```
summary(model001)
```

```
# Видим, что R^2 около 98%.
```

```
summary(model002)
```

```
# Видим, что R^2 около 76%.
```

```
# Делаем вывод, что исходные регрессоры необходимо заменить логарифмами, чтобы  
избежать линейной зависимости.
```

```
# Построим зависимости с логарифмами от регрессоров
```

```
model_log1 = lm(Infant.Mortality~I(log(Fertility))+Agriculture, data)
```

```
summary(model_log1)
```

```
# В данной модели незначительно уменьшился R^2, выросли стандартные ошибки.
```

```
model_log2 = lm(Infant.Mortality~Fertility+I(log(Agriculture)), data)
```

```
summary(model_log2)
```

```
# В данной модели уменьшился R^2 почти на 3 %.
```

```
model_log3 = lm(Infant.Mortality~I(log(Fertility))+I(log(Agriculture)), data)
summary(model_log3)
```

```
# В данной модели тоже уменьшился R^2 на 3 %, а к тому же выросли стандартные ошибки.
```

```
# Делаем вывод, что логарифмы от регрессоров не улучшают нашу модель, поэтому #стоит вводить различные произведения регрессоров, чтобы добиться ее улучшения.
```

```
#-----Ввод различных произведений пар регрессоров-----
```

```
# Также, как и в случае с логарифмами исследуем квадраты регрессоров на линейную зависимость от исходных регрессоров.
```

```
model0001 = lm((Fertility)^2~Fertility, data)
model0002 = lm((Agriculture)^2~Agriculture, data)
model0003 = lm(Agriculture*Fertility~I(Agriculture*Fertility), data)
```

```
summary(model0001)
```

```
# Видим, что R^2 около 98%.
```

```
summary(model0002)
```

```
# Видим, что R^2 около 94%.
```

```
summary(model0003)
```

```
# Здесь, мы также видим, что подгонка слишком хорошая.
```

```
# Делаем вывод, что исходные регрессоры необходимо заменить квадратами или произведением, чтобы избежать линейной зависимости.
```

```
# Построим зависимости с всевозможными парами произведений регрессоров
```

```

model_a = lm(Infant.Mortality~I(Fertility^2)+Agriculture, data)
model_b = lm(Infant.Mortality~Fertility+I(Agriculture^2), data)
model_c = lm(Infant.Mortality~I(Fertility*Agriculture), data)

summary(model_a)
# Видим, что R^2 около 18%.
summary(model_b)
# Видим, что R^2 около 21%.
summary(model_c)
# Здесь, мы видим, что R^2 стал отрицательным, что свидетельствует о плохой зависимости.

# Вывод: из всех моделей лучшая - model_b.
# Ее R^2 самый лучший из всех моделей, стандартные ошибки маленькие, есть звездочки у каждого регрессора.

model_best = lm(Infant.Mortality~Fertility+I(Agriculture^2), data)
summary(model_best)

#-----Построение доверительных интервалов для всех коэффициентов в модели-----
# Для того, чтобы найти доверительные интервалы для коэффициентов в модели,
# необходимо найти значение t-критерия Стьюдента. Кол-во измерений в обучающей
# выборке 47, из них 3 коэффициента рассчитаны.

# Число степеней свободы в модели 47 - 3 = 44. Для p = 95% критерий Стьюдента будет:
t_critical = qt(0.975, df = 44)
t_critical

# Значение t-критерия Стьюдента приблизительно 2.015

```

```

# Доверительный интервал строится по формуле [Estimate - t * Std.Error; Estimate + t * Std.Error]

# Найдем доверительный интервал для коэффициента k1(коэффициента Fertility).
# Estimate(Fertility) = 0.12; Std.Error(Fertility) = 0.03;
# Доверительный интервал для k1: [0.12 - 2.015*0.03; 0.12 + 2.015*0.03], k1: [0.05955; 0.18045]
# В этот интервал не попадает 0, значит коэффициент точно не равен 0 на уровне значимости 5 %.

# Найдем доверительный интервал для коэффициента k1(коэффициента Agriculture^2).
# Estimate(Agriculture^2) = -0.0003; Std.Error(Agriculture^2) = 0.0001;
# Доверительный интервал для k1: [-0.0003 - 2.015*0.0001; -0.0003 + 2.015*0.0001], k1: [-0.0005015; -0.0000985]
# В этот интервал не попадает 0, значит коэффициент точно не равен 0 на уровне значимости 5 %.

# Найдем доверительный интервал для прогноза с регрессорами Fertility=25, I(Agriculture^2)=900, p=95%
new.data = data.frame(Fertility = 25, Agriculture = 30)
predict(model_best, new.data, interval = "confidence")

# Прогноз модели (для Fertility = 25, Agriculture = 30) оценивается как 15.47
# Доверительный интервал: [12.59; 18.35]

```

Приложение 3

```

library("lmtest")
library("rlms")
library("dplyr")
library("GGally")
library(car)

```

```
library(sandwich)
```

```
data <- read.csv("C:\\\\Users\\\\Admin\\\\Documents\\\\r14i_os26b.csv")  
glimpse(data)
```

```
data_main = select(data, jj13.2, jh5, j_marst, j_educ, j_age, status, jj6.2, jj6, jj1.1.2, jj23, jj24)  
data_main
```

В data_main - набор данных, где записаны параметры из задачи по порядку:

jj13.2 - зарплата; jh5 - пол; j_marst - семейное положение; j_educ - наличие высшего обращования;

j_age - возраст; status - тип населенного пункта; jj6.2 - длительность рабочей недели;

jj6 - есть ли подчиненные; jj1.1.2 - удовлетворенность работе; jj23 – государство (собственник или владелец);

jj24 - иностранное фирмы и частники (совладельцы или владельцы);

```
#-----Семейное положение(wed)-----
```

```
data_main["wed"] = data_main$j_marst
```

Переменная wed1 имеет значение 1 в случае, если респондент женат, 0 – в противном случае

```
data_main["wed1"] = lapply(data_main["wed"], as.character) # Приведение данных к одному и тому же типу
```

```
data_main$wed1 = 0 # В противном случае
```

```
data_main$wed1[which(data_main$wed=='2')] <- 1 # Ответ: состояте в зарегистрированном браке
```

```
data_main$wed1[which(data_main$wed=='6')] <- 1 # Ответ: официально зарегистрированы, но вместе не проживают
```

```
data_main$wed1 = as.numeric(data_main$wed1)
```

wed2 = 1, если респондент разведён или вдовец

```
data_main["wed2"] = lapply(data_main["wed"], as.character)
data_main$wed2 = 0 # По умолчанию
data_main$wed2[which(data_main$wed=='4')] <- 1 # Ответ: Разведены и в браке не состоят
data_main$wed2[which(data_main$wed=='5')] <- 1 # Ответ: Вдовец(вдова)
data_main$wed2 = as.numeric(data_main$wed2)
```

wed3 = 1, если респондент никогда не состоял в браке

```
data_main["wed3"] = lapply(data_main["wed"], as.character)
data_main$wed3 = 0 # По умолчанию
data_main$wed3[which(data_main$wed=='1')] <- 1 # Ответ: Никогда в браке не состояли
data_main$wed3 = as.numeric(data_main$wed3)
```

#-----Проверка мультиколлинеарности(wed)-----

```
model0 = lm(jj13.2 ~ wed1 + wed2 + wed3, data_main)
model0
vif(model0) # По коэффициенту вздутия дисперсии видно, что мультиколлинерности нет (все коэф. меньше 3)
```

#-----Пол(sex)-----

Из параметра пол сделайте переменную sex, имеющую значение 1 для мужчин и равную 0 для женщин.

```
data_main["sex"] = data_main$jh5
data_main$sex[which(data_main$sex=='2')] <- 0
data_main$sex[which(data_main$sex=='1')] <- 1
data_main$sex = as.numeric(data_main$sex)
```

```
#-----Тип населенного пункта (city status) -----  
--
```

```
# Из параметра, отвечающего типу населённого пункта, создайте одну дамми-  
переменную city_status со значением 1 для города
```

```
# или областного центра, 0 – в противоположном случае.
```

```
data_main["city_status"] = data_main$status  
data_main$city_status = 0 # По умолчанию  
data_main$city_status[which(data_main$status=='1')] <- 1 # Областной центр  
data_main$city_status[which(data_main$status=='2')] <- 1 # Город  
data_main$city_status = as.numeric(data_main$city_status)
```

```
#-----Полное высшее образование(higher_educ) -----
```

```
# Введите один параметр higher_educ, характеризующий наличие полного высшего об-  
разования.
```

```
data_main["higher_educ"] = data_main$j_educ  
data_main$higher_educ = 0 # По умолчанию  
data_main$higher_educ[which(data_main$j_educ=='21')] <- 1 # Диплом о высшем образо-  
вании  
data_main$higher_educ[which(data_main$j_educ=='22')] <- 1 # Аспирантура и т.п. без  
диплома  
data_main$higher_educ[which(data_main$j_educ=='23')] <- 1 # Аспирантура и т.п. с ди-  
пломом  
data_main$higher_educ = as.numeric(data_main$higher_educ)
```

```
#-----Нормализация факторных переменных-----
```

```
# Преобразование заработной платы
```

```
data_main$jj13.2[which(data_main$jj13.2>99999990)] <- NA # Исключение респондентов, затруднившихся к ответу
```

```
data_main = na.omit(data_main)
```

```
salary = data_main$jj13.2
```

```
mean(salary) # Средняя заработная плата
```

```
data_main["salary"] = (salary - mean(salary)) / sqrt(var(salary))
```

```
#data_main["salary"]
```

```
# Преобразование возраста
```

```
age = data_main$j_age
```

```
data_main["age"] = (age - mean(age)) / sqrt(var(age))
```

```
mean(age) # Средний возраст
```

```
#data_minimal["age"]
```

```
# Преобразование длительности рабочей недели
```

```
data_main$jj6.2[which(data_main$jj6.2>99999990)] <- NA # Исключение респондентов, затруднившихся к ответу
```

```
data_main = na.omit(data_main)
```

```
duration = data_main$jj6.2
```

```
mean(duration) # Средняя длительность рабочей недели
```

```
data_main["duration"] = (duration - mean(duration)) / sqrt(var(duration))
```

```
#data_main["duration"]
```

```
-----Нормализация факторных переменных (не из минимального набора данных) -----
```

```
-
```

```
# Есть ли подчиненные?
```

```
data_main["employees"] = data_main$jj6
```

```

data_main$employees = 0 # По умолчанию
data_main$employees[which(data_main$jj6=='1')] <- 1 # Есть подчиненные
data_main$employees = as.numeric(data_main$employees)
#data_main$employees

# Удовлетворенность работе
data_main["satisfaction"] = data_main$jj1.1.2
data_main$satisfaction = 0 # По умолчанию
data_main$satisfaction[which(data_main$jj1.1.2=='1')] <- 1 # Полностью удовлетворены
data_main$satisfaction[which(data_main$jj1.1.2=='2')] <- 1 # Частично удовлетворены
data_main$satisfaction = as.numeric(data_main$satisfaction)

# Государство (совладелец или владелец)
data_main["state_owner"] = data_main$jj23
data_main$state_owner = 0 # По умолчанию
data_main$state_owner[which(data_main$jj23=='1')] <- 1 # Да
data_main$state_owner = as.numeric(data_main$state_owner)

# Иностранные фирмы и частники (совладельцы или владельцы)
data_main["foreign_owner"] = data_main$jj24
data_main$foreign_owner = 0 # По умолчанию
data_main$foreign_owner[which(data_main$jj24=='1')] <- 1 # Да
data_main$foreign_owner = as.numeric(data_main$foreign_owner)

#-----Построение линейной регрессии зарплаты на остальные параметры---

model_main = lm(salary~wed1+wed2+wed3+duration+age+sex+city_status+higher_educ+satisfaction+employees+foreign_owner+state_owner, data_main)
summary(model_main)

```

```
# Наша модель имеет довольно высокий коэффициент детерминации( $R^2$ ) = 0.2324.  
Хотя вся зарплата зависит от наших регрессоров  
# прямым образом, так как кол-во звездочек у всех регрессоров высокое.
```

```
vif(model_main)
```

```
# Значения коэффициента вздутия дисперсии небольшие, значит можно сказать, что  
наши регрессоры линейно независимые
```

```
# друг от друга
```

```
#-----Введение функций от регрессоров-----
```

```
model_main  
lm(salary~wed1+wed2+wed3+duration+age+sex+city_status+higher_educ+satisfaction+emp  
loyees+foreign_owner+state_owner, data_main)  
summary(model_main)
```

```
# -----Вводим логарифмы от регрессоров
```

```
# Логарифмы следует вводить только от вещественных переменных, так как смысла  
вводить их в дамми-переменные нет
```

```
# Проверим насколько сильно отклоняются вещественные переменные от среднего и  
добавим данное значение к регрессорам
```

```
min(data_main$age) # min(age) = -2.023109  
min(data_main$duration) # min(duration) = -3.32842
```

```
# Добавим значение 5 к каждому регрессору и возьмем от них логарифмы
```

```
# Проверим на линейную зависимость исходного регрессора от его логарифма
```

```
summary(lm((age+5)~I(log(age+5)), data_main)) # R^2 ~ 0.9872
```

```
summary(lm((duration+5)~I(log(duration+5)), data_main)) # R^2 ~ 0.9502
```

R^2 высокий в обоих случаях, поэтому исходные регрессоры нужно заменить логарифмами

Зависимость salary от вещественных переменных (age, duration) с введением логарифмов

sum-

```
mary(lm(salary~wed1+wed2+wed3+duration+I(log(age+5))+sex+city_status+satisfaction+higher_educ+employees+foreign_owner+state_owner, data_main)) # R^2 ~ 0.2311
```

sum-

```
mary(lm(salary~wed1+wed2+wed3+I(log(duration+5))+age+sex+city_status+satisfaction+employees+higher_educ+foreign_owner+state_owner, data_main)) # R^2 ~ 0.2324
```

sum-

```
mary(lm(salary~wed1+wed2+wed3+I(log(duration+5))+I(log(age+5))+sex+city_status+satisfaction+employees+higher_educ+foreign_owner+state_owner, data_main)) # R^2 ~ 0.2312
```

Мы видим, что R^2 при добавлении регрессоров становится немного ниже или равен исходной модели, поэтому добавлять логарифмы в нашу модель не имеет смысла

----- Вводим степени от регрессоров (от 0.1 до 2 с шагом 0.1)

Степени следует вводить только от вещественных переменных, так как смысла вводить их в дамми-переменные нет

Добавим значение 5 к каждому регрессору и возведем их в степени от 0.1 до 2 с шагом 0.1

Степень 0.1

mod-

```
el_degree_01=lm(salary~wed1+wed2+wed3+I((duration+5)^0.1)+I((age+5)^0.1)+sex+higher_educ+city_status+satisfaction+employees+foreign_owner+state_owner, data_main)
```

```
summary(model_degree_01) # R^2 ~ 0.2314
```

```
vif(model_degree_01) # Коэффициенты VIF хорошие
```

```
# Степень 0.2
```

```
mod-
```

```
el_degree_02=lm(salary~wed1+wed2+wed3+I((duration+5)^0.2)+I((age+5)^0.2)+sex+higher_education+city_status+satisfaction+employees+foreign_owner+state_owner, data_main)
```

```
summary(model_degree_02) # R^2 ~ 0.2315
```

```
vif(model_degree_02) # Коэффициенты VIF хорошие
```

```
# Степень 0.3
```

```
mod-
```

```
el_degree_03=lm(salary~wed1+wed2+wed3+I((duration+5)^0.3)+I((age+5)^0.3)+sex+higher_education+city_status+satisfaction+employees+foreign_owner+state_owner, data_main)
```

```
summary(model_degree_03) # R^2 ~ 0.2317
```

```
vif(model_degree_03) # Коэффициенты VIF хорошие
```

```
# Степень 0.4
```

```
mod-
```

```
el_degree_04=lm(salary~wed1+wed2+wed3+I((duration+5)^0.4)+I((age+5)^0.4)+sex+higher_education+city_status+satisfaction+employees+foreign_owner+state_owner, data_main)
```

```
summary(model_degree_04) # R^2 ~ 0.2318
```

```
vif(model_degree_04) # Коэффициенты VIF хорошие
```

```
# Степень 0.5
```

```
mod-
```

```
el_degree_05=lm(salary~wed1+wed2+wed3+I((duration+5)^0.5)+I((age+5)^0.5)+sex+higher_education+city_status+satisfaction+employees+foreign_owner+state_owner, data_main)
```

```
summary(model_degree_05) # R^2 ~ 0.2319
```

```
vif(model_degree_05) # Коэффициенты VIF хорошие
```

```
# Степень 0.6
```

```
model_degree_06
```

```
=  
lm(salary~wed1+wed2+wed3+I((duration+5)^0.6)+I((age+5)^0.6)+sex+city_status+higher_education+satisfaction+employees+foreign_owner+state_owner, data_main)
```

```
summary(model_degree_06) # R^2 ~ 0.232
```

```
vif(model_degree_06) # Коэффициенты VIF хорошие
```

```
# Степень 0.7  
mod-  
el_degree_07=lm(salary~wed1+wed2+wed3+I((duration+5)^0.7)+I((age+5)^0.7)+sex+high  
er_educ+city_status+satisfaction+employees+foreign_owner+state_owner, data_main)  
summary(model_degree_07) # R^2 ~ 0.2321  
vif(model_degree_07) # Коэффициенты VIF хорошие
```

```
# Степень 0.8  
mod-  
el_degree_08=lm(salary~wed1+wed2+wed3+I((duration+5)^0.8)+I((age+5)^0.8)+sex+high  
er_educ+city_status+satisfaction+employees+foreign_owner+state_owner, data_main)  
summary(model_degree_08) # R^2 ~ 0.2322  
vif(model_degree_08) # Коэффициенты VIF хорошие
```

```
# Степень 0.9  
mod-  
el_degree_09=lm(salary~wed1+wed2+wed3+I((duration+5)^0.9)+I((age+5)^0.9)+sex+high  
er_educ+city_status+satisfaction+employees+foreign_owner+state_owner, data_main)  
summary(model_degree_09) # R^2 ~ 0.2323  
vif(model_degree_09) # Коэффициенты VIF хорошие
```

```
# Степень 1 тоже самое, что и исходная модель, поэтому пропускаем
```

```
# Степень 1.1  
mod-  
el_degree_11=lm(salary~wed1+wed2+wed3+I((duration+5)^1.1)+I((age+5)^1.1)+sex+high  
er_educ+city_status+satisfaction+employees+foreign_owner+state_owner, data_main)  
summary(model_degree_11) # R^2 ~ 0.2324  
vif(model_degree_11) # Коэффициенты VIF хорошие
```

```
# Степень 1.2
```

```
mod-
el_degree_12=lm(salary~wed1+wed2+wed3+I((duration+5)^1.2)+I((age+5)^1.2)+sex+high
r_educ+city_status+satisfaction+employees+foreign_owner+state_owner, data_main)
```

```
summary(model_degree_12) # R^2 ~ 0.2325
```

```
vif(model_degree_12) # Коэффициенты VIF хорошие
```

Степень 1.3

```
mod-
el_degree_13=lm(salary~wed1+wed2+wed3+I((duration+5)^1.3)+I((age+5)^1.3)+sex+high
r_educ+city_status+satisfaction+employees+foreign_owner+state_owner, data_main)
```

```
summary(model_degree_13) # R^2 ~ 0.2325
```

```
vif(model_degree_13) # Коэффициенты VIF хорошие
```

Степень 1.4

```
mod-
el_degree_14=lm(salary~wed1+wed2+wed3+I((duration+5)^1.4)+I((age+5)^1.4)+sex+high
r_educ+city_status+satisfaction+employees+foreign_owner+state_owner, data_main)
```

```
summary(model_degree_14) # R^2 ~ 0.2325
```

```
vif(model_degree_14) # Коэффициенты VIF хорошие
```

Степень 1.5

```
mod-
el_degree_15=lm(salary~wed1+wed2+wed3+I((duration+5)^1.5)+I((age+5)^1.5)+sex+high
r_educ+city_status+satisfaction+employees+foreign_owner+state_owner, data_main)
```

```
summary(model_degree_15) # R^2 ~ 0.2326
```

```
vif(model_degree_15) # Коэффициенты VIF хорошие
```

Степень 1.6

```
mod-
el_degree_16=lm(salary~wed1+wed2+wed3+I((duration+5)^1.6)+I((age+5)^1.6)+sex+high
r_educ+city_status+satisfaction+employees+foreign_owner+state_owner, data_main)
```

```
summary(model_degree_16) # R^2 ~ 0.2326
```

```
vif(model_degree_16) # Коэффициенты VIF хорошие
```

```
# Степень 1.7
```

```
mod-
```

```
el_degree_17=lm(salary~wed1+wed2+wed3+I((duration+5)^1.7)+I((age+5)^1.7)+sex+higher_education+city_status+satisfaction+employees+foreign_owner+state_owner, data_main)
```

```
summary(model_degree_17) # R^2 ~ 0.2326
```

```
vif(model_degree_17) # Коэффициенты VIF хорошие
```

```
# Степень 1.8
```

```
mod-
```

```
el_degree_18=lm(salary~wed1+wed2+wed3+I((duration+5)^1.8)+I((age+5)^1.8)+sex+higher_education+city_status+satisfaction+employees+foreign_owner+state_owner, data_main)
```

```
summary(model_degree_18) # R^2 ~ 0.2326
```

```
vif(model_degree_18) # Коэффициенты VIF хорошие
```

```
# Степень 1.9
```

```
mod-
```

```
el_degree_19=lm(salary~wed1+wed2+wed3+I((duration+5)^1.9)+I((age+5)^1.9)+sex+higher_education+city_status+satisfaction+employees+foreign_owner+state_owner, data_main)
```

```
summary(model_degree_19) # R^2 ~ 0.2326
```

```
vif(model_degree_19) # Коэффициенты VIF хорошие
```

```
# Степень 2
```

```
mod-
```

```
el_degree_20=lm(salary~wed1+wed2+wed3+I((duration+5)^2)+I((age+5)^2)+sex+higher_education+city_status+satisfaction+employees+foreign_owner+state_owner, data_main)
```

```
summary(model_degree_20) # R^2 ~ 0.2326
```

```
vif(model_degree_20) # Коэффициенты VIF хорошие
```

```
# Проанализировав ввод степеней в модель, мы приходим к выводу, что зависимость становится лучше при росте степени.
```

```
# Коэффициент VIF у всех моделей хорошие, поэтому приходим к выводу, что линейной зависимости между регрессорами нет.
```

```
# Самая лучшая модель по коэффициенту детерминации R^2 model_degree_20.
```

```
#-----Ввод произведения вещественных переменных
```

```
sum-
mary(lm(salary~wed1+wed2+wed3+I(duration*age)+sex+city_status+higher_educ+satisfaction+employees+foreign_owner+state_owner, data_main))

vif(lm(salary~wed1+wed2+wed3+I(duration*age)+sex+city_status+satisfaction+employees+foreign_owner+state_owner, data_main))
```

```
# В модели при введении произведения регрессоров коэффициент детерминации R^2 стал ниже, при этом коэффициенты взаимодействия дисперсии хорошие.
```

```
# !!!!!!!
```

```
# Лучшей моделью стала model_degree_20. Эта модель имеет самый высокий коэффициент детерминации R^2 и множество звездочек (значимость регрессоров хорошая).
```

```
model_ = lm(salary~wed1+wed2+wed3+I((duration+5)^2)+I((age+5)^2)+sex+higher_educ+city_status+satisfaction+employees+foreign_owner+state_owner, data_main)

summary(model_)
```

```
# Попробуем исключить wed1, wed2 из нашей модели
```

```
model_corrected = lm(salary~wed3+I((duration+5)^2)+I((age+5)^2)+sex+higher_educ+city_status+satisfaction+employees+foreign_owner+state_owner, data_main)
```

```
# R^2 упал незначительно, а максимальное кол-во звездочек теперь у всех регрессоров, значит wed1, wed2 можно исключить из нашей модели.
```

```
# model_corrected - модель, с которой будем работать в дальнейшем
```

```
#-----Вывод о том, какие индивиды получают наибольшую зарплату-----
```

```
summary(model_corrected)
```

Из команды summary(model_corrected) мы видим, что коэффициент регрессора:

wed3 - отрицательный

I((duration + 5)^2) - положительный

I((age + 5)^2) - отрицательный

sex - положительный

higher_educ - положительный

city_status - положительный

satisfaction - положительный

employees - положительный

foreign_owner - положительный

state_owner - отрицательный

Вывод о том, какие индивиды получают большую зарплату: большую зарплату получают в большинстве своём мужчины с продолжительной рабочей неделей, имеющие высшее

образование, проживающие в городе, удовлетворённые своей заработной платой, имеющие подчиненных. При этом, если иностранные фирмы и частники являются

совладельцами или владельцами Вашего предприятия, то это положительно сказывается на уровне заработной платы. Возраст не влияет на уровень заработанной платы, а если

государство являются совладельцами или владельцами Вашего предприятия, то это отрицательно сказывается на уровне заработной платы.

-----Оцените лучшие модели для подмножества индивидов, указанных в варианте. Сделайте вывод о том, какие индивиды получают наибольшую зарплату. -----

Женщины не замужем-----

```
data_woman = subset(data_main, sex==0) # Женщины
```

```
data_woman_moment = subset(data_woman, wed1==0) # Женщины не замужем
```

```
#data_woman_moment
```

```
# Возьмём выбранную мной откорректированную модель зависимости зарплаты от других параметров, и учтём, что находимся в подмножестве
```

```
# женщин не замужем (в этом подмножестве переменные sex и wed1 равны единице).
```

```
model_subset1 =  
lm(salary~wed3+I((duration+5)^2)+I((age+5)^2)+higher_educ+city_status+satisfaction+employees+foreign_owner+state_owner, data_woman_moment)
```

```
summary(model_subset1) # R^2 ~ 21%
```

```
vif(model_subset1) # Коэффициенты VIF хорошие(невысокие) для всех регрессоров
```

```
# Женщины не замужем получают высокую заработную плату, если имеют диплом о высшем образовании, живут в городе, имеют подчиненных и собственником их предприятия
```

```
# является иностранная компания или частник. Такой вывод был сделан из коэффициентов регрессоров из команды summary (если высокие и положительные включаем в вывод).
```

```
# Женщины, живущие в городе, разведённые-----
```

```
data_woman_city = subset(data_woman, city_status==0) # Женщины, живущие в городе
```

```
data_woman_city_moment = subset(data_woman_city, wed2==1) # Женщины, живущие в городе, разведённые
```

```
# Возьмём выбранную мной откорректированную модель зависимости зарплаты от других параметров, и учтём, что находимся в подмножестве
```

```
# женщин, живущих в городе, разведённых (в этом подмножестве переменные sex, wed2, city_status равны единице).
```

```
model_subset1 =  
lm(salary~I((duration+5)^2)+I((age+5)^2)+higher_educ+satisfaction+employees+state_owner, data_woman_city_moment)
```

```
summary(model_subset1) # R^2 ~ 27%
```

```
vif(model_subset1) # Коэффициенты VIF хорошие(невысокие) для всех регрессоров
```

```
# Разведенные женщины, живущие в городе, получают высокую заработную плату, если имеют диплом о высшем образовании, удовлетворены своей работой и имеют своих подчиненных.
```

```
# Такой вывод был сделан из коэффициентов регрессоров из команды summary (если высокие и положительные включаем в вывод).
```

Приложение 4

```
import pandas  
import numpy as np  
from sklearn.tree import DecisionTreeClassifier  
  
data = pandas.read_csv("C:\\\\Users\\\\Admin\\\\Desktop\\\\BankChurners.csv", index_col = "CLIE-  
ENTNUM")  
data_sel = data.loc[:,data.columns.isin(['Attrition_Flag','Customer_Age','Gender',  
'Education_Level', 'Marital_Status','Income_Category','Total_Relationship_Count',  
'Avg_Utilization_Ratio'])]  
data_sel = data_sel.dropna() # Исключим пустые строки.  
  
# Нормализуем переменные.  
  
# Обрабатаем переменную "Attrition_Flag" (активность клиента), сделав из неё дамми-  
переменную.  
# Пусть существующий клиент ("Existing Customer") класс 1, а привлеченный клиент  
("Attrited Customer") класс 0.  
  
data_sel['Attrition_Flag'] = np.where(data_sel['Attrition_Flag'] == 'Existing Customer', 1, 0)  
#print(data_sel['Attrition_Flag'])  
  
# Обрабатаем переменную "Gender" (гендер), сделав из неё дамми-переменную.  
# Пусть мужчина("M") класс 1, а женщина("F") класс 0.
```

```
data_sel['Gender'] = np.where(data_sel['Gender'] == 'M', 1, 0)
#print(data_sel['Gender'])

# Обработаем переменную "Education_Level" (уровень образования), приведя её к числовому виду.

# Клиенты с докторской степенью --> 0
data_sel['Education_Level'] = np.where(data_sel['Education_Level'] == 'Doctorate', 0, data_sel['Education_Level'])

# Клиенты с послевузовским профессиональным образованием --> 1
data_sel['Education_Level'] = np.where(data_sel['Education_Level'] == 'Post-Graduate', 1, data_sel['Education_Level'])

# Клиенты - аспиранты --> 2
data_sel['Education_Level'] = np.where(data_sel['Education_Level'] == 'Graduate', 2, data_sel['Education_Level'])

# Клиенты, учащиеся в колледже --> 3
data_sel['Education_Level'] = np.where(data_sel['Education_Level'] == 'College', 3, data_sel['Education_Level'])

# Клиенты, учащиеся в старшей средней школе --> 4
data_sel['Education_Level'] = np.where(data_sel['Education_Level'] == 'High School', 4, data_sel['Education_Level'])

# Unknownm --> 5
data_sel['Education_Level'] = np.where(data_sel['Education_Level'] == 'Unknown', 5, data_sel['Education_Level'])

# Uneducated --> 6
data_sel['Education_Level'] = np.where(data_sel['Education_Level'] == 'Uneducated', 6, data_sel['Education_Level'])

#print(data_sel['Education_Level'])

# Обработаем переменную "Marital_Status" (семейное положение), приведя её к числовому виду.

# Клиенты, состоящие в браке --> 0
```

```

data_sel['Marital_Status'] = np.where(data_sel['Marital_Status'] == 'Married', 0, data_sel['Marital_Status'])

# Клиенты, не состоящие в браке --> 1

data_sel['Marital_Status'] = np.where(data_sel['Marital_Status'] == 'Single', 1, data_sel['Marital_Status'])

# Разведенные клиенты --> 2

data_sel['Marital_Status'] = np.where(data_sel['Marital_Status'] == 'Divorced', 2, data_sel['Marital_Status'])

# Unknown --> 3

data_sel['Marital_Status'] = np.where(data_sel['Marital_Status'] == 'Unknown', 3, data_sel['Marital_Status'])

#print(data_sel['Marital_Status'])

# Обрабатываем переменную "Income_Category" (категория годового дохода), приведя её к числовому виду.

# Клиенты с годовым доходом более 120 тысяч долларов --> 0

data_sel['Income_Category'] = np.where(data_sel['Income_Category'] == '$120K +', 0, data_sel['Income_Category'])

# Клиенты с годовым доходом от 80 до 120 тысяч долларов --> 1

data_sel['Income_Category'] = np.where(data_sel['Income_Category'] == '$80K - $120K', 1, data_sel['Income_Category'])

# Клиенты с годовым доходом от 60 до 80 тысяч долларов --> 2

data_sel['Income_Category'] = np.where(data_sel['Income_Category'] == '$60K - $80K', 2, data_sel['Income_Category'])

# Клиенты с годовым доходом от 40 до 60 тысяч долларов --> 3

data_sel['Income_Category'] = np.where(data_sel['Income_Category'] == '$40K - $60K', 3, data_sel['Income_Category'])

# Клиенты с годовым доходом менее 40 тысяч долларов --> 4

data_sel['Income_Category'] = np.where(data_sel['Income_Category'] == 'Less than $40K', 4, data_sel['Income_Category'])

# Unknown --> 5

```

```

data_sel['Income_Category'] = np.where(data_sel['Income_Category'] == 'Unknown', 5, data_sel['Income_Category'])

#print(data_sel['Income_Category'])

# Обрабатаем переменную "Customer Age" (возраст клиента) в соответствии с условием задачи.

# Если возраст клиента выше среднего значения – класс 0, ниже или совпадает – класс 1.

mean_age = sum(data_sel['Customer_Age']) // 10000
data_sel['Customer_Age'] = np.where(data_sel['Customer_Age'] > mean_age, 0, 1)
Customer_Age = data_sel.loc[:, data_sel.columns.isin(['Customer_Age'])]

#print(Customer_Age)

data2 = data_sel.loc[:,data_sel.columns.isin(['Attrition_Flag', 'Gender', 'Education_Level', 'Marital_Status', 'Income_Category','Total_Relationship_Count', 'Avg_Utilization_Ratio'])]
target = data_sel.Customer_Age

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(data2, target, test_size = 0.3)
from sklearn.tree import DecisionTreeClassifier

tree = DecisionTreeClassifier(random_state = 241, max_depth = 3)
y_pred = tree.predict(x_test) # Прогноз

import sklearn.metrics
from sklearn.metrics import classification_report

report = classification_report(y_test, y_pred)

```

```

print(report)

from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

plt.figure(figsize = (200, 100))
plot_tree(tree, feature_names = x_train.columns, class_names = ["0", "1"], filled = True,
rounded = True)
plt.show()

# Построение классификатора Случайный Лес (Random Forest) из 100, 200, 300 решающих деревьев и его оценка с помощью метрик
# accuracy, recall, f1, precision.

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

param_grid = {'n_estimators' : list(range(100, 301, 100)), 'max_features' : ['auto'],
'max_depth': list(range(1,20)), 'criterion' :['gini']}
RFC_100_300 = GridSearchCV(estimator=RandomForestClassifier(), param_grid = param_grid, cv = 5, refit = True)
RFC_100_300.fit(x_train, y_train)
RFC_100_300.best_params_ # Выявление лучшего количества деревьев для Random Forest.

# С помощью GridSearchCV мы узнали, что лучшим классификатором по числу деревьев (при оценке от 100 до 300 с шагом 100) является Random Forest из 300 решающих деревьев.

# Теперь попробуем построить Random Forest из больше чем 300 деревьев (уменьшать смысла нет, так как у классификаторов, состоящих из менее, чем 300 деревьев хуже точность, исходя из результата GridSearchCV) и с помощью GridSearchCV выявим, какой наилучший по количеству решающих деревьев Random Forest.

```

```
param_grid = {'n_estimators': list(range(300, 400, 10)), 'max_features': ['auto'], 'max_depth': list(range(1,20)), 'criterion': ['gini']}

RFC_300_400 = GridSearchCV(estimator=RandomForestClassifier(), param_grid = param_grid, cv = 5, refit = True)

RFC_300_400.fit(X_train, y_train)

RFC_300_400.best_params_
```

Исходя из результата GridSearchCV, лучшим классификатором по количеству решающих деревьев является Random Forest из 330 решающих деревьев.

```
param_grid = {'n_estimators': [330], 'max_features': ['auto'], 'max_depth': list(range(1,20)), 'criterion': ['gini']}

RFC_330 = GridSearchCV(estimator=RandomForestClassifier(), param_grid = param_grid, cv = 5, refit = True)

RFC_330.fit(x_train, y_train)
```

```
print("accuracy:"+str(np.average(sklearn.model_selection.cross_val_score(RFC.best_estimator_, x_test, y_test, scoring='accuracy'))))

print("f1:" +str(np.average(sklearn.model_selection.cross_val_score(RFC.best_estimator_, x_test, y_test, scoring='f1'))))

print("precision:"+str(np.average(sklearn.model_selection.cross_val_score(RFC.best_estimator_, x_test, y_test, scoring='precision'))))

print("recall:" +str(np.average(sklearn.model_selection.cross_val_score(RFC.best_estimator_, x_test, y_test, scoring='recall'))))
```

Точность Random Forest из 330 решающих деревьев:

```
# accuracy = 0.5340531879534091; precision = 0.5357495033098759;
# f1 = 0.5752483135706458; recall = 0.6318680043788645.
```

Приложение 5

```
import numpy as np # библиотека для эффективной работы с данными
import pandas as pd # библиотека для работы с наборами данных
import matplotlib.pyplot as plt # библиотека для визуализации
```

```
import seaborn as sns # еще одна библиотека для построения графиков
data = pd.read_csv('C:\\\\Users\\\\Admin\\\\Desktop\\\\Speed_Dating_Data.csv', encoding = "ISO-8859-1")
data.shape
data.head() # первые 5 записей
data.tail() # последние 5 записей

data_subset = data.loc[:,data.columns.isin(['gender', 'dec', 'age', 'samerace','goal', 'career_c', 'date','like','attr1_1', 'sinc1_1', 'intel1_1', 'fun1_1', 'amb1_1', 'shar1_1'])]

data_subset.head()
data_subset.info()
data_subset.describe() # статистический анализ числовых столбцов

data_set = data_subset

# Нормализация признаков через стандартное отклонение

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

data_set[['age', 'goal', 'career_c', 'date','like','attr1_1', 'sinc1_1', 'intel1_1', 'fun1_1', 'amb1_1', 'shar1_1']] = pd.DataFrame(scaler.fit_transform(data_set[['age', 'goal', 'career_c', 'date','like','attr1_1', 'sinc1_1', 'intel1_1', 'fun1_1', 'amb1_1', 'shar1_1']]), columns = data_set[['age', 'goal', 'career_c', 'date','like','attr1_1', 'sinc1_1', 'intel1_1', 'fun1_1', 'amb1_1', 'shar1_1']].columns)

data_set = data_set.dropna() # Удаление строк с пустыми значениями

data_set.info()
data_subset_1 = data_subset

# Заменяем пустые значения на среднее значение столбцов
```

```

for col in data_subset_1.columns:
    data_subset_1[col].fillna(data_subset_1[col].mean(), inplace=True)

# Нормализация признаков через стандартное отклонение
from sklearn.preprocessing import StandardScaler

scale_features_std = StandardScaler()
features_std = scale_features_std.fit_transform(data_subset_1[['age', 'goal', 'date','like',
    'attr1_1', 'sinc1_1', 'intel1_1', 'fun1_1',
    'amb1_1', 'shar1_1']])

```

Ищем выбросы (аномальные значения). Будем считать, что значение аномально, если его стандартное отклонение ≥ 3 по модулю.

```

for j in range(0, 10):
    for i in range(0, 8378):
        if abs(features_std[i][j]) >= 3:
            print(j, i)

for i in range(0, 10):
    s = 0
    for j in range(0, 8378):
        s += features_std[j][i]
    s = s / 8378
    print(i, s)

```

```

target = data_set.dec # Выделяем целевую переменную
train = data_set.drop(['dec'], axis=1)

from sklearn.model_selection import train_test_split

```

```
# Выделяем тренировочную и тестовую выборки с параметрами test_size = 0.3,
random_state = 42
```

```
x_train, x_test, y_train, y_test = train_test_split(train, target, test_size = 0.3, random_state = 42)
```

```
N_train, _ = x_train.shape
```

```
N_test, _ = x_test.shape
```

```
print(N_train)
```

```
data_subset.corr()
```

```
plt.figure(figsize=(12,10), dpi= 70)
```

```
sns.heatmap(data_subset.corr(),           xticklabels=data_subset.corr().columns,           ytick-
```

```
labels=data_subset.corr().columns,
```

```
cmap='RdYlGn', center=0, annot=True)
```

```
plt.xticks(fontsize=10)
```

```
plt.yticks(fontsize=10)
```

```
plt.show()
```

```
from sklearn.decomposition import PCA
```

```
%matplotlib inline
```

```
import matplotlib.pyplot as plt
```

```
pca = PCA()
```

```
pca.fit(x_train)
```

```
X_pca = pca.transform(x_train)
```

```
for i, component in enumerate(pca.components_):
```

```
    print("{} component: {}% of initial variance".format(i + 1, round(100 *  
pca.explained_variance_ratio_[i], 2)))
```

```

print(" + ".join("% .3f x %s" % (value, name) for value, name in
zip(component,train.columns)))

plt.figure(figsize=(10,7))
plt.plot(np.cumsum(pca.explained_variance_ratio_), color='k', lw=2)
plt.axhline(0.9, c='r')
plt.axvline(3.2, c='b')

from sklearn.manifold import TSNE
tsne1 = TSNE(learning_rate=200, random_state=130)
transformed1 = tsne1.fit_transform(train)
x_axist1 = transformed1[:, 0]
y_axist1 = transformed1[:, 1]
plt.scatter(x_axist1, y_axist1, c=target)
plt.show()

```

Приложение 6

```

import numpy as np # библиотека для эффективной работы с данными
import pandas as pd # библиотека для работы с наборами данных
import matplotlib.pyplot as plt # библиотека для визуализации
import seaborn as sns # еще одна библиотека для построения графиков

data = pd.read_csv('C:\\\\Users\\\\Admin\\\\Desktop\\\\Speed_Dating_Data.csv', encoding = "ISO-8859-1")

data.info()

data_quality = data.loc[:,data.columns.isin(['gender',
                                              'attr1_1', 'sinc1_1', 'intel1_1', 'fun1_1','amb1_1', 'shar1_1',
                                              'attr1_2', 'sinc1_2', 'intel1_2', 'fun1_2','amb1_2', 'shar1_2',
                                              'attr1_3', 'sinc1_3', 'intel1_3', 'fun1_3','amb1_3', 'shar1_3',
                                              'attr2_1', 'sinc2_1', 'intel2_1', 'fun2_1','amb2_1', 'shar2_1',
                                              'attr2_2', 'sinc2_2', 'intel2_2', 'fun2_2','amb2_2', 'shar2_2',
                                              'attr2_3', 'sinc2_3', 'intel2_3', 'fun2_3','amb2_3', 'shar2_3',
                                              'attr3_1', 'sinc3_1', 'intel3_1', 'fun3_1','amb3_1', 'shar3_1',
                                              'attr3_2', 'sinc3_2', 'intel3_2', 'fun3_2','amb3_2', 'shar3_2',
                                              'attr3_3', 'sinc3_3', 'intel3_3', 'fun3_3','amb3_3', 'shar3_3',
                                              'attr4_1', 'sinc4_1', 'intel4_1', 'fun4_1','amb4_1', 'shar4_1',
                                              'attr4_2', 'sinc4_2', 'intel4_2', 'fun4_2','amb4_2', 'shar4_2',
                                              'attr4_3', 'sinc4_3', 'intel4_3', 'fun4_3','amb4_3', 'shar4_3',
                                              'attr5_1', 'sinc5_1', 'intel5_1', 'fun5_1','amb5_1', 'shar5_1',
                                              'attr5_2', 'sinc5_2', 'intel5_2', 'fun5_2','amb5_2', 'shar5_2',
                                              'attr5_3', 'sinc5_3', 'intel5_3', 'fun5_3','amb5_3', 'shar5_3',
                                              'attr6_1', 'sinc6_1', 'intel6_1', 'fun6_1','amb6_1', 'shar6_1',
                                              'attr6_2', 'sinc6_2', 'intel6_2', 'fun6_2','amb6_2', 'shar6_2',
                                              'attr6_3', 'sinc6_3', 'intel6_3', 'fun6_3','amb6_3', 'shar6_3',
                                              'attr7_1', 'sinc7_1', 'intel7_1', 'fun7_1','amb7_1', 'shar7_1',
                                              'attr7_2', 'sinc7_2', 'intel7_2', 'fun7_2','amb7_2', 'shar7_2',
                                              'attr7_3', 'sinc7_3', 'intel7_3', 'fun7_3','amb7_3', 'shar7_3',
                                              'attr8_1', 'sinc8_1', 'intel8_1', 'fun8_1','amb8_1', 'shar8_1',
                                              'attr8_2', 'sinc8_2', 'intel8_2', 'fun8_2','amb8_2', 'shar8_2',
                                              'attr8_3', 'sinc8_3', 'intel8_3', 'fun8_3','amb8_3', 'shar8_3',
                                              'attr9_1', 'sinc9_1', 'intel9_1', 'fun9_1','amb9_1', 'shar9_1',
                                              'attr9_2', 'sinc9_2', 'intel9_2', 'fun9_2','amb9_2', 'shar9_2',
                                              'attr9_3', 'sinc9_3', 'intel9_3', 'fun9_3','amb9_3', 'shar9_3',
                                              'attr10_1', 'sinc10_1', 'intel10_1', 'fun10_1','amb10_1', 'shar10_1',
                                              'attr10_2', 'sinc10_2', 'intel10_2', 'fun10_2','amb10_2', 'shar10_2',
                                              'attr10_3', 'sinc10_3', 'intel10_3', 'fun10_3','amb10_3', 'shar10_3',
                                              'attr11_1', 'sinc11_1', 'intel11_1', 'fun11_1','amb11_1', 'shar11_1',
                                              'attr11_2', 'sinc11_2', 'intel11_2', 'fun11_2','amb11_2', 'shar11_2',
                                              'attr11_3', 'sinc11_3', 'intel11_3', 'fun11_3','amb11_3', 'shar11_3',
                                              'attr12_1', 'sinc12_1', 'intel12_1', 'fun12_1','amb12_1', 'shar12_1',
                                              'attr12_2', 'sinc12_2', 'intel12_2', 'fun12_2','amb12_2', 'shar12_2',
                                              'attr12_3', 'sinc12_3', 'intel12_3', 'fun12_3','amb12_3', 'shar12_3',
                                              'attr13_1', 'sinc13_1', 'intel13_1', 'fun13_1','amb13_1', 'shar13_1',
                                              'attr13_2', 'sinc13_2', 'intel13_2', 'fun13_2','amb13_2', 'shar13_2',
                                              'attr13_3', 'sinc13_3', 'intel13_3', 'fun13_3','amb13_3', 'shar13_3',
                                              'attr14_1', 'sinc14_1', 'intel14_1', 'fun14_1','amb14_1', 'shar14_1',
                                              'attr14_2', 'sinc14_2', 'intel14_2', 'fun14_2','amb14_2', 'shar14_2',
                                              'attr14_3', 'sinc14_3', 'intel14_3', 'fun14_3','amb14_3', 'shar14_3',
                                              'attr15_1', 'sinc15_1', 'intel15_1', 'fun15_1','amb15_1', 'shar15_1',
                                              'attr15_2', 'sinc15_2', 'intel15_2', 'fun15_2','amb15_2', 'shar15_2',
                                              'attr15_3', 'sinc15_3', 'intel15_3', 'fun15_3','amb15_3', 'shar15_3',
                                              'attr16_1', 'sinc16_1', 'intel16_1', 'fun16_1','amb16_1', 'shar16_1',
                                              'attr16_2', 'sinc16_2', 'intel16_2', 'fun16_2','amb16_2', 'shar16_2',
                                              'attr16_3', 'sinc16_3', 'intel16_3', 'fun16_3','amb16_3', 'shar16_3',
                                              'attr17_1', 'sinc17_1', 'intel17_1', 'fun17_1','amb17_1', 'shar17_1',
                                              'attr17_2', 'sinc17_2', 'intel17_2', 'fun17_2','amb17_2', 'shar17_2',
                                              'attr17_3', 'sinc17_3', 'intel17_3', 'fun17_3','amb17_3', 'shar17_3',
                                              'attr18_1', 'sinc18_1', 'intel18_1', 'fun18_1','amb18_1', 'shar18_1',
                                              'attr18_2', 'sinc18_2', 'intel18_2', 'fun18_2','amb18_2', 'shar18_2',
                                              'attr18_3', 'sinc18_3', 'intel18_3', 'fun18_3','amb18_3', 'shar18_3',
                                              'attr19_1', 'sinc19_1', 'intel19_1', 'fun19_1','amb19_1', 'shar19_1',
                                              'attr19_2', 'sinc19_2', 'intel19_2', 'fun19_2','amb19_2', 'shar19_2',
                                              'attr19_3', 'sinc19_3', 'intel19_3', 'fun19_3','amb19_3', 'shar19_3',
                                              'attr20_1', 'sinc20_1', 'intel20_1', 'fun20_1','amb20_1', 'shar20_1',
                                              'attr20_2', 'sinc20_2', 'intel20_2', 'fun20_2','amb20_2', 'shar20_2',
                                              'attr20_3', 'sinc20_3', 'intel20_3', 'fun20_3','amb20_3', 'shar20_3',
                                              'attr21_1', 'sinc21_1', 'intel21_1', 'fun21_1','amb21_1', 'shar21_1',
                                              'attr21_2', 'sinc21_2', 'intel21_2', 'fun21_2','amb21_2', 'shar21_2',
                                              'attr21_3', 'sinc21_3', 'intel21_3', 'fun21_3','amb21_3', 'shar21_3',
                                              'attr22_1', 'sinc22_1', 'intel22_1', 'fun22_1','amb22_1', 'shar22_1',
                                              'attr22_2', 'sinc22_2', 'intel22_2', 'fun22_2','amb22_2', 'shar22_2',
                                              'attr22_3', 'sinc22_3', 'intel22_3', 'fun22_3','amb22_3', 'shar22_3',
                                              'attr23_1', 'sinc23_1', 'intel23_1', 'fun23_1','amb23_1', 'shar23_1',
                                              'attr23_2', 'sinc23_2', 'intel23_2', 'fun23_2','amb23_2', 'shar23_2',
                                              'attr23_3', 'sinc23_3', 'intel23_3', 'fun23_3','amb23_3', 'shar23_3',
                                              'attr24_1', 'sinc24_1', 'intel24_1', 'fun24_1','amb24_1', 'shar24_1',
                                              'attr24_2', 'sinc24_2', 'intel24_2', 'fun24_2','amb24_2', 'shar24_2',
                                              'attr24_3', 'sinc24_3', 'intel24_3', 'fun24_3','amb24_3', 'shar24_3',
                                              'attr25_1', 'sinc25_1', 'intel25_1', 'fun25_1','amb25_1', 'shar25_1',
                                              'attr25_2', 'sinc25_2', 'intel25_2', 'fun25_2','amb25_2', 'shar25_2',
                                              'attr25_3', 'sinc25_3', 'intel25_3', 'fun25_3','amb25_3', 'shar25_3',
                                              'attr26_1', 'sinc26_1', 'intel26_1', 'fun26_1','amb26_1', 'shar26_1',
                                              'attr26_2', 'sinc26_2', 'intel26_2', 'fun26_2','amb26_2', 'shar26_2',
                                              'attr26_3', 'sinc26_3', 'intel26_3', 'fun26_3','amb26_3', 'shar26_3',
                                              'attr27_1', 'sinc27_1', 'intel27_1', 'fun27_1','amb27_1', 'shar27_1',
                                              'attr27_2', 'sinc27_2', 'intel27_2', 'fun27_2','amb27_2', 'shar27_2',
                                              'attr27_3', 'sinc27_3', 'intel27_3', 'fun27_3','amb27_3', 'shar27_3',
                                              'attr28_1', 'sinc28_1', 'intel28_1', 'fun28_1','amb28_1', 'shar28_1',
                                              'attr28_2', 'sinc28_2', 'intel28_2', 'fun28_2','amb28_2', 'shar28_2',
                                              'attr28_3', 'sinc28_3', 'intel28_3', 'fun28_3','amb28_3', 'shar28_3',
                                              'attr29_1', 'sinc29_1', 'intel29_1', 'fun29_1','amb29_1', 'shar29_1',
                                              'attr29_2', 'sinc29_2', 'intel29_2', 'fun29_2','amb29_2', 'shar29_2',
                                              'attr29_3', 'sinc29_3', 'intel29_3', 'fun29_3','amb29_3', 'shar29_3',
                                              'attr30_1', 'sinc30_1', 'intel30_1', 'fun30_1','amb30_1', 'shar30_1',
                                              'attr30_2', 'sinc30_2', 'intel30_2', 'fun30_2','amb30_2', 'shar30_2',
                                              'attr30_3', 'sinc30_3', 'intel30_3', 'fun30_3','amb30_3', 'shar30_3',
                                              'attr31_1', 'sinc31_1', 'intel31_1', 'fun31_1','amb31_1', 'shar31_1',
                                              'attr31_2', 'sinc31_2', 'intel31_2', 'fun31_2','amb31_2', 'shar31_2',
                                              'attr31_3', 'sinc31_3', 'intel31_3', 'fun31_3','amb31_3', 'shar31_3',
                                              'attr32_1', 'sinc32_1', 'intel32_1', 'fun32_1','amb32_1', 'shar32_1',
                                              'attr32_2', 'sinc32_2', 'intel32_2', 'fun32_2','amb32_2', 'shar32_2',
                                              'attr32_3', 'sinc32_3', 'intel32_3', 'fun32_3','amb32_3', 'shar32_3',
                                              'attr33_1', 'sinc33_1', 'intel33_1', 'fun33_1','amb33_1', 'shar33_1',
                                              'attr33_2', 'sinc33_2', 'intel33_2', 'fun33_2','amb33_2', 'shar33_2',
                                              'attr33_3', 'sinc33_3', 'intel33_3', 'fun33_3','amb33_3', 'shar33_3',
                                              'attr34_1', 'sinc34_1', 'intel34_1', 'fun34_1','amb34_1', 'shar34_1',
                                              'attr34_2', 'sinc34_2', 'intel34_2', 'fun34_2','amb34_2', 'shar34_2',
                                              'attr34_3', 'sinc34_3', 'intel34_3', 'fun34_3','amb34_3', 'shar34_3',
                                              'attr35_1', 'sinc35_1', 'intel35_1', 'fun35_1','amb35_1', 'shar35_1',
                                              'attr35_2', 'sinc35_2', 'intel35_2', 'fun35_2','amb35_2', 'shar35_2',
                                              'attr35_3', 'sinc35_3', 'intel35_3', 'fun35_3','amb35_3', 'shar35_3',
                                              'attr36_1', 'sinc36_1', 'intel36_1', 'fun36_1','amb36_1', 'shar36_1',
                                              'attr36_2', 'sinc36_2', 'intel36_2', 'fun36_2','amb36_2', 'shar36_2',
                                              'attr36_3', 'sinc36_3', 'intel36_3', 'fun36_3','amb36_3', 'shar36_3',
                                              'attr37_1', 'sinc37_1', 'intel37_1', 'fun37_1','amb37_1', 'shar37_1',
                                              'attr37_2', 'sinc37_2', 'intel37_2', 'fun37_2','amb37_2', 'shar37_2',
                                              'attr37_3', 'sinc37_3', 'intel37_3', 'fun37_3','amb37_3', 'shar37_3',
                                              'attr38_1', 'sinc38_1', 'intel38_1', 'fun38_1','amb38_1', 'shar38_1',
                                              'attr38_2', 'sinc38_2', 'intel38_2', 'fun38_2','amb38_2', 'shar38_2',
                                              'attr38_3', 'sinc38_3', 'intel38_3', 'fun38_3','amb38_3', 'shar38_3',
                                              'attr39_1', 'sinc39_1', 'intel39_1', 'fun39_1','amb39_1', 'shar39_1',
                                              'attr39_2', 'sinc39_2', 'intel39_2', 'fun39_2','amb39_2', 'shar39_2',
                                              'attr39_3', 'sinc39_3', 'intel39_3', 'fun39_3','amb39_3', 'shar39_3',
                                              'attr40_1', 'sinc40_1', 'intel40_1', 'fun40_1','amb40_1', 'shar40_1',
                                              'attr40_2', 'sinc40_2', 'intel40_2', 'fun40_2','amb40_2', 'shar40_2',
                                              'attr40_3', 'sinc40_3', 'intel40_3', 'fun40_3','amb40_3', 'shar40_3',
                                              'attr41_1', 'sinc41_1', 'intel41_1', 'fun41_1','amb41_1', 'shar41_1',
                                              'attr41_2', 'sinc41_2', 'intel41_2', 'fun41_2','amb41_2', 'shar41_2',
                                              'attr41_3', 'sinc41_3', 'intel41_3', 'fun41_3','amb41_3', 'shar41_3',
                                              'attr42_1', 'sinc42_1', 'intel42_1', 'fun42_1','amb42_1', 'shar42_1',
                                              'attr42_2', 'sinc42_2', 'intel42_2', 'fun42_2','amb42_2', 'shar42_2',
                                              'attr42_3', 'sinc42_3', 'intel42_3', 'fun42_3','amb42_3', 'shar42_3',
                                              'attr43_1', 'sinc43_1', 'intel43_1', 'fun43_1','amb43_1', 'shar43_1',
                                              'attr43_2', 'sinc43_2', 'intel43_2', 'fun43_2','amb43_2', 'shar43_2',
                                              'attr43_3', 'sinc43_3', 'intel43_3', 'fun43_3','amb43_3', 'shar43_3',
                                              'attr44_1', 'sinc44_1', 'intel44_1', 'fun44_1','amb44_1', 'shar44_1',
                                              'attr44_2', 'sinc44_2', 'intel44_2', 'fun44_2','amb44_2', 'shar44_2',
                                              'attr44_3', 'sinc44_3', 'intel44_3', 'fun44_3','amb44_3', 'shar44_3',
                                              'attr45_1', 'sinc45_1', 'intel45_1', 'fun45_1','amb45_1', 'shar45_1',
                                              'attr45_2', 'sinc45_2', 'intel45_2', 'fun45_2','amb45_2', 'shar45_2',
                                              'attr45_3', 'sinc45_3', 'intel45_3', 'fun45_3','amb45_3', 'shar45_3',
                                              'attr46_1', 'sinc46_1', 'intel46_1', 'fun46_1','amb46_1', 'shar46_1',
                                              'attr46_2', 'sinc46_2', 'intel46_2', 'fun46_2','amb46_2', 'shar46_2',
                                              'attr46_3', 'sinc46_3', 'intel46_3', 'fun46_3','amb46_3', 'shar46_3',
                                              'attr47_1', 'sinc47_1', 'intel47_1', 'fun47_1','amb47_1', 'shar47_1',
                                              'attr47_2', 'sinc47_2', 'intel47_2', 'fun47_2','amb47_2', 'shar47_2',
                                              'attr47_3', 'sinc47_3', 'intel47_3', 'fun47_3','amb47_3', 'shar47_3',
                                              'attr48_1', 'sinc48_1', 'intel48_1', 'fun48_1','amb48_1', 'shar48_1',
                                              'attr48_2', 'sinc48_2', 'intel48_2', 'fun48_2','amb48_2', 'shar48_2',
                                              'attr48_3', 'sinc48_3', 'intel48_3', 'fun48_3','amb48_3', 'shar48_3',
                                              'attr49_1', 'sinc49_1', 'intel49_1', 'fun49_1','amb49_1', 'shar49_1',
                                              'attr49_2', 'sinc49_2', 'intel49_2', 'fun49_2','amb49_2', 'shar49_2',
                                              'attr49_3', 'sinc49_3', 'intel49_3', 'fun49_3','amb49_3', 'shar49_3',
                                              'attr50_1', 'sinc50_1', 'intel50_1', 'fun50_1','amb50_1', 'shar50_1',
                                              'attr50_2', 'sinc50_2', 'intel50_2', 'fun50_2','amb50_2', 'shar50_2',
                                              'attr50_3', 'sinc50_3', 'intel50_3', 'fun50_3','amb50_3', 'shar50_3',
                                              'attr51_1', 'sinc51_1', 'intel51_1', 'fun51_1','amb51_1', 'shar51_1',
                                              'attr51_2', 'sinc51_2', 'intel51_2', 'fun51_2','amb51_2', 'shar51_2',
                                              'attr51_3', 'sinc51_3', 'intel51_3', 'fun51_3','amb51_3', 'shar51_3',
                                              'attr52_1', 'sinc52_1', 'intel52_1', 'fun52_1','amb52_1', 'shar52_1',
                                              'attr52_2', 'sinc52_2', 'intel52_2', 'fun52_2','amb52_2', 'shar52_2',
                                              'attr52_3', 'sinc52_3', 'intel52_3', 'fun52_3','amb52_3', 'shar52_3',
                                              'attr53_1', 'sinc53_1', 'intel53_1', 'fun53_1','amb53_1', 'shar53_1',
                                              'attr53_2', 'sinc53_2', 'intel53_2', 'fun53_2','amb53_2', 'shar53_2',
                                              'attr53_3', 'sinc53_3', 'intel53_3', 'fun53_3','amb53_3', 'shar53_3',
                                              'attr54_1', 'sinc54_1', 'intel54_1', 'fun54_1','amb54_1', 'shar54_1',
                                              'attr54_2', 'sinc54_2', 'intel54_2', 'fun54_2','amb54_2', 'shar54_2',
                                              'attr54_3', 'sinc54_3', 'intel54_3', 'fun54_3','amb54_3', 'shar54_3',
                                              'attr55_1', 'sinc55_1', 'intel55_1', 'fun55_1','amb55_1', 'shar55_1',
                                              'attr55_2', 'sinc55_2', 'intel55_2', 'fun55_2','amb55_2', 'shar55_2',
                                              'attr55_3', 'sinc55_3', 'intel55_3', 'fun55_3','amb55_3', 'shar55_3',
                                              'attr56_1', 'sinc56_1', 'intel56_1', 'fun56_1','amb56_1', 'shar56_1',
                                              'attr56_2', 'sinc56_2', 'intel56_2', 'fun56_2','amb56_2', 'shar56_2',
                                              'attr56_3', 'sinc56_3', 'intel56_3', 'fun56_3','amb56_3', 'shar56_3',
                                              'attr57_1', 'sinc57_1', 'intel57_1', 'fun57_1','amb57_1', 'shar57_1',
                                              'attr57_2', 'sinc57_2', 'intel57_2', 'fun57_2','amb57_2', 'shar57_2',
                                              'attr57_3', 'sinc57_3', 'intel57_3', 'fun57_3','amb57_3', 'shar57_3',
                                              'attr58_1', 'sinc58_1', 'intel58_1', 'fun58_1','amb58_1', 'shar58_1',
                                              'attr58_2', 'sinc58_2', 'intel58_2', 'fun58_2','amb58_2', 'shar58_2',
                                              'attr58_3', 'sinc58_3', 'intel58_3', 'fun58_3','amb58_3', 'shar58_3',
                                              'attr59_1', 'sinc59_1', 'intel59_1', 'fun59_1','amb59_1', 'shar59_1',
                                              'attr59_2', 'sinc59_2', 'intel59_2', 'fun59_2','amb59_2', 'shar59_2',
                                              'attr59_3', 'sinc59_3', 'intel59_3', 'fun59_3','amb59_3', 'shar59_3',
                                              'attr60_1', 'sinc60_1', 'intel60_1', 'fun60_1','amb60_1', 'shar60_1',
                                              'attr60_2', 'sinc60_2', 'intel60_2', 'fun60_2','amb60_2', 'shar60_2',
                                              'attr60_3', 'sinc60_3', 'intel60_3', 'fun60_3','amb60_3', 'shar60_3',
                                              'attr61_1', 'sinc61_1', 'intel61_1', 'fun61_1','amb61_1', 'shar61_1',
                                              'attr61_2', 'sinc61_2', 'intel61_2', 'fun61_2','amb61_2', 'shar61_2',
                                              'attr61_3', 'sinc61_3', 'intel61_3', 'fun61_3','amb61_3', 'shar61_3',
                                              'attr62_1', 'sinc62_1', 'intel62_1', 'fun62_1','amb62_1', 'shar62_1',
                                              'attr62_2', 'sinc62_2', 'intel62_2', 'fun62_2','amb62_2', 'shar62_2',
                                              'attr62_3', 'sinc62_3', 'intel62_3', 'fun62_3','amb62_3', 'shar62_3',
                                              'attr63_1', 'sinc63_1', 'intel63_1', 'fun63_1','amb63_1', 'shar63_1',
                                              'attr63_2', 'sinc63_2', 'intel63_2', 'fun63_2','amb63_2', 'shar63_2',
                                              'attr63_3', 'sinc63_3', 'intel63_3', 'fun63_3','amb63_3', 'shar63_3',
                                              'attr64_1', 'sinc64_1', 'intel64_1', 'fun64_1','amb64_1', 'shar64_1',
                                              'attr64_2', 'sinc64_2', 'intel64_2', 'fun64_2','amb64_2', 'shar64_2',
                                              'attr64_3', 'sinc64_3', 'intel64_3', 'fun64_3','amb64_3', 'shar64_3',
                                              'attr65_1', 'sinc65_1', 'intel65_1', 'fun65_1','amb65_1', 'shar65_1',
                                              'attr65_2', 'sinc65_2', 'intel65_2', 'fun65_2','amb65_2', 'shar65_2',
                                              'attr65_3', 'sinc65_3', 'intel65_3', 'fun65_3','amb65_3', 'shar65_3',
                                              'attr66_1', 'sinc66_1', 'intel66_1', 'fun66_1','amb66_1', 'shar66_1',
                                              'attr66_2', 'sinc66_2', 'intel66_2', 'fun66_2','amb66_2', 'shar66_2',
                                              'attr66_3', 'sinc66_3', 'intel66_3', 'fun66_3','amb66_3', 'shar66_3',
                                              'attr67_1', 'sinc67_1', 'intel67_1', 'fun67_1','amb67_1', 'shar67_1',
                                              'attr67_2', 'sinc67_2', 'intel67_2', 'fun67_2','amb67_2', 'shar67_2',
                                              'attr67_3', 'sinc67_3', 'intel67_3', 'fun67_3','amb67_3', 'shar67_3',
                                              'attr68_1', 'sinc68_1', 'intel68_1', 'fun68_1','amb68_1', 'shar68_1',
                                              'attr68_2', 'sinc68_2', 'intel68_2', 'fun68_2','amb68_2', 'shar68_2',
                                              'attr68_3', 'sinc68_3', 'intel68_3', 'fun68_3','amb68_3', 'shar68_3',
                                              'attr69_1', 'sinc69_1', 'intel69_1', 'fun69_1','amb69_1', 'shar69_1',
                                              'attr69_2', 'sinc69_2', 'intel69_2', 'fun69_2','amb69_2', 'shar69_2',
                                              'attr69_3', 'sinc69_3', 'intel69_3', 'fun69_3','amb69_3', 'shar69_3',
                                              'attr70_1', 'sinc70_1', 'intel70_1', 'fun70_1','amb70_1', 'shar70_1',
                                              'attr70_2', 'sinc70_2', 'intel70_2', 'fun70_2','amb70_2', 'shar70_2',
                                              'attr70_3', 'sinc70_3', 'intel70_3', 'fun70_3','amb70_3', 'shar70_3',
                                              'attr71_1', 'sinc71_1', 'intel71_1', 'fun71_1','amb71_1', 'shar71_1',
                                              'attr71_2', 'sinc71_2', 'intel71_2', 'fun71_2','amb71_2', 'shar71_2',
                                              'attr71_3', 'sinc71_3', 'intel71_3', 'fun71_3','amb71_3', 'shar71_3',
                                              'attr72_1', 'sinc72_1', 'intel72_1', 'fun72_1','amb72_1', 'shar72_1',
                                              'attr72_2', 'sinc72_2', 'intel72_2', 'fun72_2','amb72_2', 'shar72_2',
                                              'attr72_3', 'sinc72_3', 'intel72_3', 'fun72_3','amb72_3', 'shar72_3',
                                              'attr73_1', 'sinc73_1', 'intel73_1', 'fun73_1','amb73_1', 'shar73_1',
                                              'attr73_2', 'sinc73_2', 'intel73_2', 'fun73_2','amb73_2', 'shar73_2',
                                              'attr73_3', 'sinc73_3', 'intel73_3', 'fun73_3','amb73_3', 'shar73_3',
                                              'attr74_1', 'sinc74_1', 'intel74_1', 'fun74_1','amb74_1', 'shar74_1',
                                              'attr74_2', 'sinc74_2', 'intel74_2', 'fun74_2','amb74_2', 'shar74_2',
                                              'attr74_3', 'sinc74_3', 'intel74_3', 'fun74_3','amb74_3', 'shar74_3',
                                              'attr75_1', 'sinc75_1', 'intel75_1', 'fun75_1','amb75_1', 'shar75_1',
                                              'attr75_2', 'sinc75_2', 'intel75_2', 'fun75_2','amb75_2', 'shar75_2',
                                              'attr75_3', 'sinc75_3', 'intel75_3', 'fun75_3','amb75_3', 'shar75_3',
                                              'attr76_1', 'sinc76_1', 'intel76_1', 'fun76_1','amb76_1', 'shar76_1',
                                              'attr76_2', 'sinc76_2', 'intel76_2', 'fun76_2','amb76_2', 'shar76_2',
                                              'attr76_3', 'sinc76_3', 'intel76_3', 'fun76_3','amb76_3', 'shar76_3',
                                              'attr77_1', 'sinc77_1', 'intel77_1', 'fun77_1','amb77_1', 'shar77_1',
                                              'attr77_2', 'sinc77_2', 'intel77_2', 'fun77_2','amb77_2', 'shar77_2',
                                              'attr77_3', 'sinc77_3', 'intel77_3', 'fun77_3','amb77_3', 'shar77_3',
                                              'attr78_1', 'sinc78_1', 'intel78_1', 'fun78_1','amb78_1', 'shar78_1',
                                              'attr78_2', 'sinc78_2', 'intel78_2', 'fun78_2','amb78_2', 'shar78_2',
                                              'attr78_3', 'sinc78_3', 'intel78_3', 'fun78_3','amb78_3', 'shar78_3',
                                              'attr79_1', 'sinc79_1', 'intel79_1', 'fun79_1','amb79_1', 'shar79_1',
                                              'attr79_2', 'sinc79_2', 'intel79_2', 'fun79_2','amb79_2', 'shar79_2',
                                              'attr79_3', 'sinc79_3', 'intel79_3', 'fun79_3','amb79_3', 'shar79_3',
                                              'attr80_1', 'sinc80_1', 'intel80_1', 'fun80_1','amb80_1', 'shar80_1',
                                              'attr80_2', 'sinc80_2', 'intel80_2', 'fun80_2','amb80_2', 'shar80_2',
                                              'attr80_3', 'sinc80_3', 'intel80_3', 'fun80_3','amb80_3', 'shar80_3',
                                              'attr81_1', 'sinc81_1', 'intel81_1', 'fun81_1','amb81_1', 'shar81_1',
                                              'attr81_2', 'sinc81_2', 'intel81_2', 'fun81_2','amb81_2', 'shar81_2',
                                              'attr81_3', 'sinc81_3', 'intel81_3', 'fun81_3','amb81_3', 'shar81_3',
                                              'attr82_1', 'sinc82_1', 'intel82_1', 'fun82_1','amb82_1', 'shar82_1',
                                              'attr82_2', 'sinc82_2', 'intel82_2', 'fun82_2','amb82_2', 'shar82_2',
                                              'attr82_3', 'sinc82_3', 'intel82_3', 'fun82_3','amb82_3', 'shar82_3',
                                              'attr83_1', 'sinc83_1', 'intel83_1', 'fun83_1','amb83_1', 'shar83_1',
                                              'attr83_2', 'sinc83_2', 'intel83_2', 'fun83_2','amb83_2', 'shar83_2',
                                              'attr83_3', 'sinc83_3', 'intel83_3', 'fun83_3','amb
```

```
'attr1_s', 'sinc1_s', 'intel1_s', 'fun1_s','amb1_s', 'shar1_s'])]

data_quality = data_quality.dropna() # Исключим строки с пустыми значениями

data_quality.head() # Просмотр подмножества с преобразованными пустыми значениями

# Подмножество женщин
woman_subset = data_quality.loc[data_quality['gender'] == 0]

# Вычисляем средние значения остальных признаков
mean_values = woman_subset.mean()

# Сортируем признаки по убыванию средних значений
sorted_values = mean_values.sort_values()

# Выбираем наименьшие значения
least_desirable_qualities = sorted_values[1:]

print(least_desirable_qualities)

# Подмножество мужчин
male_subset = data_quality.loc[data_quality['gender'] == 1]

# Вычисляем средние значения остальных признаков
mean_values = male_subset.mean()

# Сортируем признаки по убыванию средних значений
sorted_values = mean_values.sort_values()

# Выбираем наименьшие значения
```

```

least_desirable_qualities = sorted_values[1:]

print(least_desirable_qualities)

data_race_or_shar = data.loc[:,data.columns.isin(['dec', 'samerace',
'shar1_1','shar1_2','shar1_3','shar1_s'])]

data_race_or_shar = data_race_or_shar.dropna() # Исключим строки с пустыми значениями

data_race_or_shar.head() # Просмотр подмножества с преобразованными пустыми значениями

from sklearn.model_selection import train_test_split

target = data_race_or_shar.dec
data_sel = data_race_or_shar.drop('dec', axis = 1)

x_train, x_test, y_train, y_test = train_test_split(data_sel, target, test_size= 0.3, random_state=4477)

from sklearn.linear_model import LinearRegression

linmodel = LinearRegression()
model = linmodel.fit(x_train,y_train)

# Так как значения целевой переменной 0 и 1, а линейная регрессия,
# не предназначена для разбиения на два класса, делаем это самостоятельно.
# Все, что больше 0.5 = 1, что меньше = 0.

from sklearn import metrics

```

```
def linear_scorer(estimator, x, y):
    scorer_predictions = estimator.predict(x)

    scorer_predictions[scorer_predictions > 0.5] = 1
    scorer_predictions[scorer_predictions <= 0.5] = 0

    return metrics.accuracy_score(y, scorer_predictions)

linear_scorer(linmodel, x_test, y_test) # Выявление точности модели
print(model.coef_) # Просмотр коэффициентов линейной регрессии

data = data.drop(['match', 'dec_o', 'iid', 'id', 'field', 'undergra', 'mn_sat', 'tuition', 'from',
                  'zipcode', 'income', 'career'], axis=1)

# подсчет количества пропущенных значений в каждом столбце
missing_values_count = data.isna().sum()

# получение списка столбцов, в которых количество пропущенных значений больше
# 4000
columns_to_drop = list(missing_values_count[missing_values_count > 4000].index)

# удаление столбцов из таблицы
data = data.drop(columns=columns_to_drop)

# удаление всех пустых значений в каждом столбце
data = data.dropna()

# вывод обновленной таблицы
print(data)

from sklearn.decomposition import PCA
```

```

from sklearn.preprocessing import StandardScaler

for i in range(data.shape[1] + 1):
    pca2 = PCA(n_components = i)
    pca2.fit(data)
    print(i, sum(pca2.explained_variance_ratio_))

import matplotlib.pyplot as plt

x = range(13)
y = [0, 0.7403156919379744, 0.8019949480534895, 0.8278750970521416,
     0.8432442238303203, 0.856045823306342, 0.8686220381111169, 0.8798100562670698,
     0.8906162631507597, 0.8989738598212641, 0.9062720998116164, 0.9119188483557351,
     0.9173855501314454]

plt.plot(x, y, 'ro-')

plt.xlabel('Количество компонент')
plt.ylabel('Доля объясненной дисперсии')

plt.show()

# Нормализуем данные

scaler = StandardScaler()
data = scaler.fit_transform(data)

from sklearn.cluster import DBSCAN
from sklearn.manifold import TSNE

```

```

#снизим размерность данных при помощи метода главных компонент
pca = PCA(n_components=10).fit(data)
pdata = pca.transform(data)

#используем dbSCAN для обнаружения кластеров
dbSCAN = DBSCAN(eps = 1, min_samples=5)
dbSCAN.fit(pdata)
labels = dbSCAN.labels_

#используем t-SNE для снижения размерности
tsne = TSNE()
pca_2d = tsne.fit_transform(pdata)

#визуализируем кластеры
for label in set(labels):
    if label == -1:
        plt.scatter(pca_2d[labels == label, 0], pca_2d[labels == label, 1], color='blue')
    else:
        plt.scatter(pca_2d[labels == label, 0], pca_2d[labels == label, 1], color=plt.cm.jet(label / np.max(labels + 1)))

plt.title('DBSCAN нашел {} кластеров и {} шумовых точек'.format(len(set(labels)) - (1 if -1 in labels else 0),
                                                               np.sum(labels == -1)))
plt.show()

from sklearn.cluster import KMeans
from sklearn.manifold import TSNE

kmeans = KMeans(n_init = 'auto', random_state = 157)

```

```

kmeans.fit(data)
labels = kmeans.labels_

tsne = TSNE()
pca_2d = tsne.fit_transform(data)

for label in set(labels):
    if label == -1:
        plt.scatter(pca_2d[labels == label, 0], pca_2d[labels == label, 1], color='blue')
    else:
        plt.scatter(pca_2d[labels == label, 0], pca_2d[labels == label, 1], color=plt.cm.jet(label / np.max(labels + 1)))

plt.title('kmeans нашел {} кластеров и {} шумовых точек'.format(len(set(labels)) - (1 if -1 in labels else 0),
                                                               np.sum(labels == -1)))
plt.show()

from sklearn.model_selection import train_test_split

target = data['dec']
data_sel = data.drop(['dec'], axis=1)

x_train, x_test, y_train, y_test = train_test_split(data_sel, target, test_size=0.3, random_state=4477)

from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

#используем решающее дерево
for i in range(1, 20):

```

```
T = DecisionTreeClassifier(random_state=338, max_depth = i)
T = T.fit(x_train, y_train)
print(str(i) + ":" + str(T.score(x_test, y_test)))
```

```
from sklearn.metrics import classification_report
```

```
T = DecisionTreeClassifier(random_state=338, max_depth = 3)
T = T.fit(x_train, y_train)
```

```
pred = T.predict(x_test)
report = classification_report(y_test, pred)
print(report)
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
```

```
# Визуализация решающего дерева
plt.figure(figsize=(200, 100))
plot_tree(T, feature_names=x_train.columns, class_names=["0", "1"], filled=True, rounded=True)
plt.show()
```