



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: Предсказание стоимости медицинской страховки (kaggle.com)
приказ университета о направлении на практику
от «9» февраля 2022 г. № 1038 - С

Отчет представлен к
рассмотрению:

Студент группы КМБО-01-
21

Сиразетдинов Р.Д.
(расшифровка подписи)
«__» _____ 2022 г.

Отчет утвержден.
Допущен к защите:

Руководитель практики от
кафедры

Петрусевич Д.А.
(расшифровка подписи)
«__» _____ 2022 г.

Москва 2022



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-01-21 института искусственного интеллекта
Сиразетдинову Рустему Дамировичу

(фамилия, имя и отчество)

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

Время НИР: с «09» февраля 2022 по «31» мая 2022

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», «k 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 («k ближайших соседей»); построением модели линейной регрессии

3. ДОПОЛНИТЕЛЬНОЕ ЗАДАНИЕ: предсказание стоимости медицинской страховки (kaggle.com)

4. ОГРАНИЗАЦИОННО-МЕТОДИЧЕСКИЕ УКАЗАНИЯ: построить модель предсказания, оценить вклад каждой компоненты. Есть ли выбросы среди данных, какие? Применить алгоритмы кластеризации. Что общего между объектами в каждом кластере?

Заведующий кафедрой

высшей математики

«09» февраля 2022 г.

Ю.И.Худак

СОГЛАСОВАНО

Руководитель практики от кафедры:

«09» февраля 2022 г.

(подпись)

(Петрусеви́ч Д.А.)

(фамилия и инициалы)

Задание получил:

«09» февраля 2022 г.

(подпись)

(Сиразетдинов Р.Д.)

(фамилия и инициалы)

ИНСТРУКТАЖ ПРОВЕДЕН:

Вид мероприятия	ФИО ответственного, подпись, дата	ФИО студента, подпись, дата
Охрана труда	<u>Петрусеви́ч Д.А.</u> «09» февраля 2022 г.	<u>Сиразетдинов Р.Д.</u> «09» февраля 2022 г.
Техника безопасности	<u>Петрусеви́ч Д.А.</u> «09» февраля 2022 г.	<u>Сиразетдинов Р.Д.</u> «09» февраля 2022 г.
Пожарная безопасность	<u>Петрусеви́ч Д.А.</u> «09» февраля 2022 г.	<u>Сиразетдинов Р.Д.</u> «09» февраля 2022 г.
Правила внутреннего распорядка	<u>Петрусеви́ч Д.А.</u> «09» февраля 2022 г.	<u>Сиразетдинов Р.Д.</u> «09» февраля 2022 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 Сиразетдинова Р.Д. 1 курса группы КМБО-01-21 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	09.02.2022	Выбор темы НИР. Пройти инструктаж по технике безопасности	
1	09.02.2022	Вводная установочная лекция	
2	14.02.2022	Построение и оценка парной регрессии с помощью языка R	
3	21.02.2022	Построение и оценка множественной регрессии с помощью языка R	
4	28.02.2022	Построение доверительных интервалов. Обработка факторных переменных. Мультиколлинеарность	
5	07.03.2022	Гетероскедастичность	
6	14.03.2022	Классификация	
7	21.03.2022	Кластеризация. Предобработка данных	
8	28.03.2022	Метод главных компонент	
9	04.04.2022	Ансамбли классификаторов.	

		Беггинг. Бустинг	
16	29.05.2022	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 практики от кафедры _____ / ФИО / Петрусевич Д.А.

Обучающийся _____ / ФИО / Сиразетдинов Р.Д.

Содержание

1	Практическая работа №1	4
1.1	Решение	4
1.1.1	Оценки	4
1.1.2	Линейная зависимость	5
1.1.3	Оценка R^2	6
1.1.4	Оценка, взаимосвязи между объясняемой пере- менной и объясняющей переменной - регрессором	7
1.2	Вывод	8
2	Практическая работа №2	9
2.1	Решение	9
2.1.1	Проверка отсутствия линейной зависимости меж- ду регрессорами	9
2.1.2	Построение линейных моделей зависимости пе- ременной от указанных регрессоров	10
2.1.3	Введение в модель логарифмов регрессоров	13
2.2	Вывод	14
2.2.1	Построение доверительных интервалов для всех коэффициентов модели	15
2.3	β -гипотеза	17
2.4	Построение доверительного интервала	18
3	Практическая работа №3	20
3.1	Решение	20
3.1.1	Начальная обработка данных	20
3.1.2	Линейная регрессия	20
3.1.3	Преобразование регрессоров построенной модели	22
3.2	Вывод	22
3.3	Оценка регрессий на предложенных подмножествах . .	24
4	Практическая работа №4	26
4.1	Описание данных	26
4.2	Введение	26

4.3	Решение	26
5	Практическая работа №5	32
5.1	Решение	33
5.1.1	Введение в данные	33
5.1.2	Представление признаков	34
5.1.3	Обработка пропущенных значений	34
5.1.4	Определение и обработка аномальных значений и выбросов	35
5.1.5	Стандартизация значений признаков	36
5.1.6	Определение целевого признака	38
5.1.7	Разбиение данных на выборки	39
5.1.8	Определение зависимости между описывающими признаками	39
5.1.9	Построение линейной зависимости	41
6	Практическая работа №6	43
6.1	Решение	43
7	Заключение	53
	Список литературы	54
8	Приложение	55

1 Практическая работа №1

1. Набор данных: **Swiss**
2. Объясняемая переменная: **Catholic**
3. Регрессоры: **Agriculture, Examination**

Пояснение:

Catholic - количество людей в процентах, относящихся к католической церкви.

Agriculture - часть мужского населения в процентах, занятая в сельскохозяйственном секторе.

Examination - количество людей в процентах, которые умеют уровень образования выше начальной школы.

Необходимо загрузить данные из указанного набора и произвести следующие действия.

1.1 Решение

1.1.1 Оценка среднего значения, дисперсии и СКО объясняемой переменной и регрессоров

```
1      # среднее значение столбцов Catholic, Agriculture,
      Examination
2      mean(data$Catholic)      # 41.14383
3      mean(data$Agriculture)  # 50.65957
4      mean(data$Examination) # 16.48936
5
6      # дисперсия(сумма квадратов отклонений всех значений от средне-
      го) столбцов
7      #Catholic, Agriculture, Examination
8      var(data$Catholic)      # 1739.295
9      var(data$Agriculture)  # 515.7994
10     var(data$Examination) # 63.64662
11
12
13     # СКО(среднеквадратическое отклонение столбцов Catholic,
      Agriculture, Examination
```



```

14 sd(data$Catholic)      # 41.70485
15 sd(data$Agriculture)   # 22.71122
16 sd(data$Examination)   # 7.977883
17

```

Листинг 1: Оценка среднего значения, дисперсии и СКО объясняемой переменной и регрессоров.

1.1.2 Построение линейной зависимости вида $y = a + b * x$, где y – объясняемая переменная, x – регрессор

Всего будет построено две модели:

1. Модель - *model_Agr*
 - Объясняемая переменная - *Catholic*
 - Регрессор - *Agriculture*
2. Модель - *model_Ex*
 - Объясняемая переменная - *Catholic*
 - Регрессор - *Examination*

```

1 # model_Agr: y- Catholic ~ x_1- Agriculture
2 model_Agr = lm(Catholic~Agriculture, data)
3 model_Agr
4 summary(model_Agr)
5

```

Листинг 2: Построение модели *model_Agr*.

Построена зависимость между y - *Catholic* - количеством католиков и x_1 - *Agriculture* - мужчинами, занятыми в сельском хозяйстве:

$$y = 3.8313 + 0.7365 * x_1$$

```

1 # model_Ex: y- Catholic ~ x_1- Examination
2 model_Ex = lm(Catholic~Examination, data)
3 model_Ex
4 summary(model_Ex)
5

```

Листинг 3: Построение модели *model_Ex*.

Построена зависимость между y - *Catholic* - количеством католиков и x_2 - *Examination* - количеством людей в процентах, которые умеют уровень образования выше начальной школы:

$$y = 90.5137 - 2.9940 * x_2$$

1.1.3 Оценка построенных моделей по значению коэффициента детерминации R^2

Для начала посмотрим на результаты, которые мы получили после построения вышеуказанных моделей. Подробнее см. [Приложение](#)

Таблица 1: Характеристики модели зависимости параметра Catholic от параметра Agriculture в наборе данных Swiss.

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	3.8313	13.8966	0.276	0.7840	
Examination	0.7365	0.2508	2.937	0.0052	**

Таблица 2: Характеристики модели зависимости параметра Catholic от параметра Examination в наборе данных Swiss.

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	90.5137	11.6779	7.751	7.96e-10	***
Examination	-2.9940	0.6388	-4.687	2.59e-05	***

1. $R^2 = 0.1422$

Таким образом можно сделать вывод, что модель *model_Arg* (таблица 1) очень плохо описывает динамику изменения значений описываемой переменной y - *Catholic* через регрессор x_1 - *Agriculture*, следовательно, необходимо выбрать другой регрессор, либо одного регрессора недостаточно, чтобы хорошо описать поведение исследуемой переменной.

2. $R^2 = 0.3131$

Можно сделать вывод о том, что модель *model_Ex* (таблица 2) некорректно описывает поведение переменной y - *Catholic* относительно регрессора x_2 - *Examination*. Для получения точных результатов необходимо построить новую модель, либо добавить в модель новые регрессоры.

1.1.4 Оценка, взаимосвязи между объясняемой переменной и объясняющей переменной - регрессором

Для того, чтобы объяснить наличие взаимосвязи между переменными необходимо проанализировать значение характеристики *p-value* построенной модели.

Отметим, что *p-value* - это вероятность того, что посчитанное в модели значение обуславливается случайностью, а не самой моделью в целом.

1. Оценим связь между регрессором x_1 и переменной y . По полученным в пункте 1.1.3 данным можно сделать вывод о том, что какая-то связь между регрессором и описываемой переменной есть, но эта связь недостаточно сильная - только 2 звездочки, значит вероятность того, что посчитанное в модели значение y обусловлено моделью и не сильно отличается от реальных значений переменной - не высока, кроме того, нет никакой связи между свободным коэффициентом и переменной y . Таким образом, есть слабая причинно-следственная связь между регрессором x_1 и описываемой переменной y .

2. Оценим связь между регрессором x_1 и переменной y . Вновь обратимся к полученным в пункте 1.1.3 данным. Теперь можно сделать вывод о том, что существует сильная связь между x_2 и y : по 3 звездочки у свободного коэффициента. и регрессора x_2 , а также низкие показатели p -value. Значит можно сделать вывод: значения посчитанные в модели слабо отличаются от реальных. Таким образом, есть сильная причинно-следственная связь между регрессором x_2 и описываемой переменной y .

1.2 Вывод

В Практическая работа №1 были построены две модели: *model_Agr* - модель зависимости переменной *Catholic* от регрессора *Agriculture* и *model_Ex* - модель зависимости переменной *Catholic* от регрессора *Examination*. По полученным в пункте 1.1.3 данным, мы выяснили, что во второй модели показатель R^2 больше чем в первой в два раза, следовательно, вторая модель лучше описывает динамику изменения переменной *Catholic*, причем стоит отметить, что коэффициент перед регрессором *Examination* отрицательный, т.е. люди, которые получали образование выше начальной школы в меньшей степени относились к католикам. Относились ли они к православной церкви или другим коммунам сказать нельзя, т.к. для этого нужно проводить дополнительные исследования.

Для подробного ознакомления с используемыми материалами см. Приложение.

2 Практическая работа №2

1. Набор данных: **attitude**
2. Объясняемая переменная: **rating**
3. Регрессоры: **raises, critical, advancel**

Таблица 3: Описание набора данных и регрессоров, задействованных в практической работе.

attitude	Данные опроса канцелярских служащих
rating	Общий рейтинг
raises	Повышение сотрудников, основываясь на их производительности.
critical	Критичность
advancel	Продвжение сотрудников по службе

2.1 Решение

2.1.1 Проверка отсутствия линейной зависимости между регрессорами

Для начала проверим зависимость между регрессорами, если некоторые будут зависеть друг от друга, то одного из них нужно будет исключить из рассмотрения. Для того, чтобы проверить зависит ли один из регрессоров от другого необходимо построить модели линейной зависимости первого от второго.

```
1 model_auxiliary_1 = lm(raises ~ critical, data)
2 summary(model_auxiliary_1)
3
```

Листинг 4: Построение линейной зависимости регрессора *raises* от регрессора *critical*.

Таблица 4: Характеристики модели: *model_auxiliary_1* (*raises* ~ *critical*).

Coefficients:					
	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	35.0246	13.8680	2.526	0.0175	*
critical	0.3960	0.1839	2.153	0.0401	*
Residual standard error: 9.801 on 28 degrees of freedom					
Multiple R-squared:	0.142	Adjusted R-squared:	0.1114		
F-statistic:	4.636 on 1 and 28 DF	p-value:	0.04008		

Теперь посмотрим на результаты построенной модели, которые представлены в таблице 4. R^2 оказался очень низким, что говорит нам о том, что построенная модель очень плохая, кроме того *p-value* довольно высок, исходя из выше сказанного, можно заключить, что регрессоры *raises* и *critical* линейно независимы.

2.1.2 Построение линейных моделей зависимости переменной от указанных регрессоров

Построим всевозможные модели и сделаем вывод.

```

1 model_auxiliary_2 = lm(raises ~ advance, attitude)
2 summary(model_auxiliary_2)
3
4 model_auxiliary_3 = lm(critical ~ advance, data)
5 summary(model_auxiliary_3)
6

```

Листинг 5: Построение линейных зависимостей.

Вывод: из таблиц 5 и 6 как итог получаем, что регрессоры *raises* и *advance* зависят друг от друга, поэтому одного из них нужно исключить из будущей модели. В дальнейшем мы проверим, какой из них полезнее для модели.

Построим линейную модель, которая описывает переменную *rating* через регрессоры *raises*, *critical*, *advance*. При этом помним, что одну из переменных *raises* и *advance* желательно исключить из модели, выясним какую.

Таблица 5: Характеристики модели: *model_auxiliary_2* (raises ~ advance).

Coefficients:					
	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	39.7216	6.8967	5.759	3.5e-06	***
advance	0.5802	0.1564	3.711	0.000907	***
Residual standard error: 8.663 on 28 degrees of freedom					
Multiple R-squared:	0.3297	Adjusted R-squared:		0.3058	
F-statistic:	13.77 on 1 and 28 DF	p-value:		0.0009068	

Таблица 6: Характеристики модели: *model_auxiliary_3* (critical ~ advance).

Coefficients:					
	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	63.0674	7.6882	8.203	6.28e-09	***
advance	0.2725	0.1743	1.563	0.129	
Residual standard error: 9.657 on 28 degrees of freedom					
Multiple R-squared:	0.08028	Adjusted R-squared:		0.04744	
F-statistic:	2.444 on 1 and 28 DF	p-value:		0.1292	

```

1  model_all = lm(rating ~ raises + critical + advance, data)
2  summary(model_all)
3
4  no_critical_model = lm(rating ~ raises + advance, data)
5  summary(no_critical_model)
6
7  no_raises_model = lm(rating ~ critical + advance, data)
8  summary(no_raises_model)
9

```

Листинг 6: Построение линейных зависимостей.

Из анализа построенных моделей(см. таблицы 6 - 8) видно, что наилучшая из всех регрессий - *no_critical_model*(таблица 8), которая имеет **Adjusted R-squared: 0.3541** и наименьшее значение p-value: 0.001043. Проанализируем модель *no_critical_model*. Один из её регрессоров *raises* хорошо связан с описываемой переменной, в то время как, второй регрессор модели *advance* - нет, его значение p-value: 0.144200 очень высоко, что говорит о большом количестве неточно-

Таблица 7: Характеристики модели: *model_all* (rating \sim raises + critical + advance).

Coefficients:					
	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	25.50063	15.60204	1.634	0.114218	
raises	0.89612	0.22555	3.973	0.000501	***
critical	-0.06882	0.20233	-0.340	0.736483	
advance	-0.31773	0.22014	-1.443	0.160870	
Residual standard error: 9.947 on 26 degrees of freedom					
Multiple R-squared:	0.4013	Adjusted R-squared:	0.3322		
F-statistic:	5.809 on 3 and 26 DF	p-value:	0.003537		

Таблица 8: Характеристики модели: *no_critical_model* (rating \sim raises + advance).

Coefficients:					
	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	21.9917	11.5114	1.910	0.066753	.
raises	0.8752	0.2134	4.101	0.000339	***
advance	-0.3243	0.2157	-1.504	0.144200	
Residual standard error: 9.783 on 27 degrees of freedom					
Multiple R-squared:	0.3986	Adjusted R-squared:	0.3541		
F-statistic:	8.949 on 2 and 27 DF	p-value:	0.001043		

Таблица 9: Характеристики модели: *no_raises_model* (rating \sim critical + advance).

Coefficients:					
	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	47.2659	18.1737	2.601	0.0149	*
critical	0.1505	0.2422	0.621	0.5396	
advance	0.1425	0.2329	0.612	0.5458	
Residual standard error: 9.783 on 27 degrees of freedom					
Multiple R-squared:	0.3986	Adjusted R-squared:	0.3322		
F-statistic:	8.949 on 2 and 27 DF	p-value:	0.001043		

стей при попытке описания *rating* через регрессор *advance*. В итоге получаем **Adjusted R-squared: 0.3541** и **Multiple R-squared: 0.3986**, что говорит нам о том, что модель не самая удачная.

2.1.3 Введение в модель логарифмов регрессоров

Теперь введем в наилучшие модели из прошлого шага (*model_all*, *no_critical_model*) логарифмы регрессоров, где это возможно. Это возможно для каждого регрессора, т.к. все они имеют числовые значения. Выявим из всех построенных моделей наилучшую.

```
1      model_all_log_1 = lm(rating ~ I(log(raises)) + critical +
2      advance, data)
3      summary(model_all_log_1)
4
5      model_all_log_2 = lm(rating ~ raises + I(log(critical)) +
6      advance, data)
7      summary(model_all_log_2)
8
9      model_all_log_3 = lm(rating ~ raises + I(log(advance)) +
10     critical, data)
11     summary(model_all_log_3)
12
13     model_all_log_4 = lm(rating ~ raises + I(log(advance)) + I(
14     log(critical)), data)
15     summary(model_all_log_4)
16
17     no_critical_model_log_1 = lm(rating ~ raises + I(log(advance
18     )), data)
19     summary(no_critical_model_log_1)
20
21     no_critical_model_log_2 = lm(rating ~ I(log(raises)) + I(log
22     (advance)), data)
23     summary(no_critical_model_log_2)
24
25     model_all_mult_1 = lm(rating ~ I(log(raises)) + I(critical *
26     advance), data)
27     summary(model_all_mult_1)
28
29     model_all_mult_2 = lm(rating ~ I(log(raises)) + I(critical
30     ^2) + advance, data)
31     summary(model_all_mult_2)
32
33     model_all_mult_3 = lm(rating ~ I(log(raises)) + I(critical
34     ^2) + I(advance^2), data)
35     summary(model_all_mult_3)
```

```

27
28     model_all_mult_4 = lm(rating ~ I(log(raises)) + critical + I
29     (advance^2), data)
30     summary(model_all_mult_4)
31     no_critical_model_mult_1 = lm(rating ~ I(raises^2) + advance
32     , data)
33     summary(no_critical_model_mult_1)
34     no_critical_model_mult_2 = lm(rating ~ I(log(raises)) + I(
    advance^2), data)
    summary(no_critical_model_mult_2)

```

Листинг 7: Построение линейных зависимостей.

Поскольку построенных моделей достаточно много, здесь лишь остановимся на некоторых комментариях, которые дают общее представление о логике построения последующих регрессий, и результатах лучших полученных моделей. С результатами остальных построенных моделей можно ознакомиться в разделе [Приложение](#).

При введении логарифмов каждой из переменных (таблицы: [10](#), [11](#)) были видны следующие изменения. После логарифмирования регрессора *raises* Multiple R-squared: 0.4102, Adjusted R-squared: 0.3422 изменились не сильно, Multiple R-squared возрос на 1,5 процента и Adjusted R-squared снизился на 1,2 процента, но значение p-value также понизилось, поэтому мы больше не будем трогать эту переменную, оставив её в покое (за исключением дальнейшего изучения квадрата этой переменной). От логарифмирования регрессоров *advance* и *critical* мы не получили пользы, показатель Multiple R-squared: 0.3986 практически не поменялся, а показатель Adjusted R-squared: 0.3293 уменьшился, и кроме того значение p-value: 0.003736 стало довольно высоким.

Получили 2 наилучшие модели: *model_all_mult_1* и *no_critical_model_log_2*.

2.2 Вывод

Лучшие модели нельзя назвать образцовыми, скорее всего это из-за того, что переменную *rating* нельзя в полной мере описать только через регрессоры *advance* и *critical* либо зависимость сложнее ли-

Таблица 10: Характеристики модели: *model_all_mult_1*.

Coefficients:					
	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-1.603e+02	5.330e+01	-3.008	0.005633	**
I(log(raises))	5.667e+01	1.382e+01	4.100	0.000339	***
I(critical * advance)	-3.259e-03	2.342e-03	-1.391	0.175475	
Residual standard error: 9.715 on 27 degrees of freedom					
Multiple R-squared:	0.4069	Adjusted R-squared:	0.363		
F-statistic:	9.264 on 2 and 27 DF	p-value:	0.0008645		

Таблица 11: Характеристики модели: *model_all_mult_2*.

Coefficients:					
	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-1.571e+02	5.199e+01	-3.022	0.005579	**
I(log(raises))	5.738e+01	1.391e+01	4.125	0.000337	***
I(critical ²)	-8.898e-04	1.417e-03	-0.628	0.535463	
advance	-2.725e-01	2.118e-01	-1.287	0.209550	
Residual standard error: 9.838 on 26 degrees of freedom					
Multiple R-squared:	0.4143	Adjusted R-squared:	0.3468		
F-statistic:	6.131 on 3 and 26 DF	p-value:	0.002693		

нейной. Наилучшим вариантом будет использование других регрессоров.

2.2.1 Построение доверительных интервалов для всех коэффициентов модели

Для построения доверительного интервала необходимо знать значение коэффициента, полученного из построенной модели, σ - стандартная ошибка (СКО коэффициента), полученную из построенной выше модели, и значение t-критерия Стьюдента, которое может быть вычислено с помощью функций языка R.

- Количество наблюдений- 30
- Количество исследуемых коэффициентов- 3
- Количество степеней свободы- $30 - 3 = 27$

```
1 t_value = qt(0.975, df = 27)
2 t_value
```

Листинг 8: Расчитаем t -критерий Стъедента.

Таким образом получим, что $t_{value} = 2.051831$.

Доверительный интервал имеет вид:

$$[\beta - t * \sigma, \beta + t * \sigma]$$

1. Расчитаем доверительный интервал для коэффициента *Intercept*

- значение коэффициента: (-115.572)
- стандартная ошибка: 46.471
- t -критерий Стъедента: 2.051831

Доверительный интервал:

$$[-115.572 - 2.051831 \times 46.471 : -115.572 + 2.051831 \times 46.471]$$

т.е.

$$[-210.9226 : -20.22136]$$

Вывод: С вероятностью $\frac{19}{20}$ значение коэффициента Intercept будет лежать в интервале $[-210.9226 : -20.22136]$. Отвергаем гипотезу $\beta = 0$, т.е. с вероятностью $\frac{1}{20}$ коэффициент Intercept может принимать значение 0, что говорит о том, что это получено хорошее значение коэффициента.

2. Расчитаем доверительный интервал для коэффициента $I(\log(\text{raises}))$

- значение коэффициента: 55.004
- стандартная ошибка: 13.119
- t -критерий Стъедента: 2.051831

Доверительный интервал:

$$[55.004 - 2.051831 \times 13.119 : 55.004 + 2.051831 \times 13.119]$$

т.е.

$$[28.08603 : 81.92197]$$

Вывод: С вероятностью $\frac{19}{20}$ значение регрессора $I(\log(raises))$ будет лежать в интервале $[28.08603 : 81.92197]$. Отвергаем гипотезу $\beta = 0$, т.е. с вероятностью $\frac{1}{20}$ регрессор $I(\log(raises))$ может принимать значение 0, что еще раз говорит о том, что это хороший регрессор.

3. Расчитаем доверительный интервал для коэффициента $I(\log(advance))$

- значение коэффициента: (-12.963)
- стандартная ошибка: 9.264
- t-критерий Стъедента: 2.051831

Доверительный интервал:

$$[55.004 - 2.051831 \times 13.119 : 55.004 + 2.051831 \times 13.119]$$

т.е.

$$[28.08603 : 81.92197]$$

Вывод: С вероятностью $\frac{19}{20}$ значение регрессора $I(\log(advance))$ будет лежать в интервале $[-31.97116 : 6.045162]$. Принимаем гипотезу $\beta = 0$, т.е. с вероятностью $\frac{1}{20}$ регрессор $I(\log(advance))$ может принимать значение 0, что еще раз говорит о том, что это плохой регрессор.

Таблица 12: Описание результатов, полученных выше.

Регрессор	Оценка коэффициента β	Std. Error	Доверительный интервал	$\beta = 0$ гипотеза
Свободный коэффициент	-115.572	46.471	[-210.9226 , -20.22136]	Отвергаем
$I(\log(raises))$	55.004	13.119	[28.08603 , 81.92197]	Отвергаем
$I(\log(advance))$	-12.963	9.264	[-31.97116 , 6.045162]	Принимаем

2.3 Вывод об отвержении или невозможности отвергнуть статистическую гипотезу о том, что коэффициент равен 0

После дополнительного изучения построенной модели *model_1* мы получили значения доверительных интервалов всех коэффициентов перед регрессорами в модели.

1. Посмотрим на доверительный интервал ‘Свободного коэффициента’.

$$[-210.9226, -20.22136]$$

Поскольку $0 \notin [-210.9226, -20.22136]$, то мы отвергает гипотезу о том, что значение этого коэффициента может равняться нулю. Значит этот коэффициент правильно описывает модель.

2. Посмотрим на доверительный интервал ‘ $I(\log(\text{raises}))$ ’.

$$[28.08603, 81.92197]$$

Поскольку $0 \notin [28.08603, 81.92197]$, то мы отвергает гипотезу о том, что значение этого коэффициента может равняться нулю. Значит переменная ‘rating’ зависит от этого регрессора, хотя границы доверительного интервала и значение Std. Error равное 13.119 слишком большие, чтобы говорить о сильной зависимости между ними.

3. Посмотрим на доверительный интервал ‘ $I(\log(\text{advance}))$ ’.

$$[-31.97116, 6.045162]$$

Поскольку $0 \in [-31.97116, 6.045162]$, то мы принимаем гипотезу о том, что значение этого коэффициента может равняться нулю. Значит переменная ‘rating’ не зависит от этого регрессора, поэтому лучше исключить её из модели.

2.4 Построение доверительного интервала для одного прогноза ($p = 95\%$).

Для построения доверительного интервала необходимо знать:

- ошибка σ по всей модели
- прогноз модели
- доверительный интервал

Но мы можем сразу построить доверительный интервал (см. таблицу 23) для прогноза модели при конкретных значениях регрессоров, не находя каждый из пунктов выше по отдельности.

```
1 new.data = data.frame(raises = 34, advance = 51
2 predict(model_1, new.data, interval = "confidence")
3
```

Листинг 9: Построение доверительного интервала для модели с помощью функций языка R.

Таблица 13: Доверительный интервал для модели.

	fit	lwr	upr
1	27.42529	7.856132	46.99445

С вероятностью $p = 95\%$ значение модели на значениях данных регрессоров будет равняться 27.42529. Таким образом, мы построили доверительный интервал для модели *model_1* (rating $I(\log(\text{raises})) + I(\log(\text{advance}))$). Но доверительный интервал получился достаточно большим ≈ 40 , что говорит о невысокой точности нашей модели, поэтому для хорошего доверительного интервала необходимо построить более верную модель.

3 Практическая работа №3

1. Набор данных: **r20i_os26c.sav** - wave 20
2. Набор параметров(обязательные): **sj13.2, age, sh5, s_marst, sj72.5c, status, sj6.2**
3. Набор параметров(дополнительные): **sj11.1, sj23**

3.1 Решение

3.1.1 Начальная обработка данных

- Создадим переменную **ds** в которую поместим информацию о выбранных столбцах.
- Создание дамми-переменной из параметра, отвечающего семейному положению.
- Преобразование переменной *sex*
- Создание дамми-переменной из параметра, отвечающего за тип населенного пункта
- Введение параметра, характеризующего уровень образования респондента
- Преобразование факторных переменных в вещественные

Подробную реализацию обработки данных с помощью языка программирования *R* см. в разделе [Приложение](#).

3.1.2 Построение линейной регрессии заработной платы на все введенные регрессоры

Построим линейную регрессию переменной ‘salary’ от всех выделенных ранее регрессоров и оценим коэффициент вздутия ‘VIF’. Обозначим эту модель как - *model_def*.


```

1 ds_2 = select(ds, salary, age, sex, h_educ, city_status, dur,
2 wed1, wed2, wed3, satisfy, of, gov)
3
4 # модель_1 - 'model_def'
5 model_def = lm(data = ds_2, salary ~ age + sex + h_educ + city_
6 status +
7 dur + wed1 + wed2 + wed3 + satisfy + of + gov)
8 summary(model_def)
vif(model_def)

```

Листинг 10: Построение модели *model_def*.

Таблица 14: Характеристики модели: *model_def*.

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-1.03259	0.06699	-15.413	< 2e-16	***
age	-0.06357	0.01320	-4.814	1.51e-06	***
sex	0.49299	0.02534	19.456	< 2e-16	***
h_educ	0.51861	0.02628	19.734	< 2e-16	***
city_status	0.33425	0.02652	12.603	< 2e-16	***
dur	0.13087	0.01226	10.672	< 2e-16	***
wed1	0.03696	0.03733	0.990	0.322194	
wed2	-0.01408	0.04761	-0.296	0.767452	
wed3	-0.11033	0.04697	-2.349	0.018847	*
satisfy	0.23748	0.03463	6.858	7.72e-12	***
of	0.29224	0.03452	8.465	< 2e-16	***
gov	0.17184	0.04992	3.442	0.000581	***
Residual standard error: 0.8972 on 5772 degrees of freedom					
Multiple R-squared: 0.1966 Adjusted R-squared: 0.195					
F-statistic 128.4 on 11 and 5772 DF p-value: < 2.2e-16					

```

1 vif(model_def)
2 age sex h_educ city_status dur
3 wed1
4 1.252570 1.138695 1.071094 1.026265 1.080268
5 2.425066
6 wed2 wed3 satisfy of gov
2.025865 1.920659 1.035515 1.037693 1.031553

```

Листинг 11: Проверка регрессоров на линейную зависимость.

Как можно видеть по данным регрессоры *wed1* и *wed2* плохо описывают поведение переменной *salary*. Возможно одна зависит от другой или они вовсе не нужны в модели.

3.1.3 Преобразование регрессоров построенной модели

Построим некоторые другие модели, экспериментируя с регрессорами, и выберем из них наилучшую. Поскольку построенных моделей очень много, здесь приведем характеристики лучших моделей, характеристики прочих можно увидеть в разделе [Приложение](#).

Таблица 15: Характеристики модели: *model_5*.

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-1.38986	0.16679	-8.333	3.22e-16	***
log(age)	-0.19561	0.04326	-4.522	7.01e-06	***
sex	0.56428	0.07208	7.829	1.49e-14	***
h_educ	0.63077	0.08806	7.163	1.73e-12	***
city_status	0.46811	0.07522	6.223	7.68e-10	***
log(dur)	0.14344	0.03867	3.710	0.000221	***
wed3	-0.06911	0.21784	-0.317	0.751120	
satisfy	0.28575	0.09670	2.955	0.003214	**
of	0.38896	0.10493	3.707	0.000224	***
gov	0.17498	0.04982	3.512	0.000447	***
Residual standard error: 1.013 on 836 degrees of freedom					
Multiple R-squared: 0.2212 Adjusted R-squared: 0.2128					
F-statistic 26.38 on 9 and 836 DF p-value: < 2.2e-16					

3.2 Вывод

Каждая из моделей(см. таблицы 14 - 17) предоставляет практически идентичные другим результаты. Посмотрев на коэффициенты перед регрессорами в моделях, можно заметить некоторую закономерность. В каждой из построенных моделей коэффициент перед регрессором *age* - количество полных лет - отрицательный. Отсюда

Таблица 16: Характеристики модели: *model_6*.

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
Intercept)	-1.38986	0.16679	-8.333	3.22e-16	***
log(age)	-0.19561	0.04326	-4.522	7.01e-06	***
sex	0.56428	0.07208	7.829	1.49e-14	***
h_educ	0.63077	0.08806	7.163	1.73e-12	***
city_status	0.46811	0.07522	6.223	7.68e-10	***
log(dur)	0.14344	0.03867	3.710	0.000221	***
wed3	-0.06911	0.21784	-0.317	0.751120	
satisfy	0.28575	0.09670	2.955	0.003214	**
of	0.38896	0.10493	3.707	0.000224	***
gov	0.40415	0.12620	3.202	0.001414	**

Residual standard error: 1.013 on 836 degrees of freedom

Multiple R-squared: 0.2212 Adjusted R-squared: 0.2128

F-statistic 26.38 on 9 and 836 DF p-value: < 2.2e-16

Таблица 17: Характеристики модели: *model_14*.

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-1.39116	0.16665	-8.348	2.87e-16	***
log(age)	-0.19572	0.04323	-4.527	6.85e-06	***
sex	0.56719	0.07145	7.938	6.58e-15	***
h_educ	0.62861	0.08775	7.164	1.72e-12	***
city_status	0.46660	0.07503	6.219	7.88e-10	***
log(dur)	0.14340	0.03864	3.711	0.000220	***
$I(satisfy^{0.5})$	0.28644	0.09662	2.965	0.003117	**
of	0.39069	0.10474	3.730	0.000204	***
gov	0.40243	0.12602	3.193	0.001458	**

Residual standard error: 1.013 on 837 degrees of freedom

Multiple R-squared: 0.2211 Adjusted R-squared: 0.2136

F-statistic 29.7 on 8 and 837 DF p-value: < 2.2e-16

можно сделать вывод, что с повышением возраста люди статистически зарабатывают меньше. В каждой из моделей регрессоры *of* - наличие официального трудоустройства, *gov* - принадлежность организации государству, *satisfy* - удовлетворенность работой - всегда довольно хорошо связаны с описываемой переменной. Логически это

действительно так. Регрессоры *h_educ*, *city_status* так же оказались очень важны. Сравнивая с моделями в которых эти регрессоры выражены не так сильно, можно заметить прирост показателя параметра ‘Adjusted R-squared’ на 1-2%. Как итог получим, что **наиболее успешны в плане зарплаты молодые мужчины с высшим образованием из развитых городов**, статистика на противоречит здравому смыслу.

3.3 Оценка регрессий на предложенных подмножествах

- Подмножество 1: **Не вступавшие в брак мужчины, без высшего образования.** Уточним датасеты и построим модели:

```

1 data5_1 = subset(ds_2, sex = 1)
2 data5_2 = subset(data5_1, wed3 = 1)
3 data5_fin = subset(data5_2, h_educ = 0)
4
5 model_5_1 = lm(data = data5_fin, salary ~ log(age) + city_status
6 +
7               log(dur) + I(satisfy^2) + of + gov)
8 summary(model_5_1)
```

Листинг 12: Подмножество 1.

Вывод: по полученным данным(см. таблицу 18) можно предположить, что заработок респондентов ограниченных на «подмножестве 1» сильно зависит от возраста - с увеличением возраста респонденты статистически зарабатывают меньше. Если посмотреть на другие регрессоры, например, на *city_status* & *log_dur*, то можно убедиться что они действительно положительно влияют на описываемую переменную *salary*.

- Подмножество 2: **Городские жители, мужчины состоящие в браке.** Уточним датасеты и построим модели:

```

1 data5_1 = subset(ds_2, sex = 1)
2 data5_2 = subset(data5_1, city_status = 1)
3 data5_fin = subset(data5_2, wed1 = 1)
```

Таблица 18: Характеристики модели: *model_5_1*.

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-0.96175	0.16832	-5.714	1.53e-08	***
log(age)	-0.19842	0.04618	-4.296	1.94e-05	***
city_status	0.48497	0.07960	6.093	1.69e-09	***
log(dur)	0.17883	0.04057	4.408	1.18e-05	***
I(<i>satisfy</i> ²)	0.30497	0.10313	2.957	0.00319	**
of	0.44941	0.11149	4.031	6.06e-05	***
gov	0.38369	0.13390	2.865	0.00427	**

Residual standard error: 1.082 on 839 degrees of freedom

Multiple R-squared: 0.1089 Adjusted R-squared: 0.1025

F-statistic 17.09 on 6 and 839 DF p-value: < 2.2e-16

```

4  model_5_2 = lm(data = data5_fin, salary ~ log(age) + h_educ +
5              log(dur) + I(satisfy^2) + of + gov)
6  summary(model_5_2)
7
8

```

Листинг 13: Подмножество 2.

Таблица 19: Характеристики модели: *data5_fin*.

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-0.61121	0.15321	-3.989	7.20e-05	***
log(age)	-0.18727	0.04557	-4.110	4.35e-05	***
h_educ	0.71432	0.09195	7.769	2.31e-14	***
log(dur)	0.16612	0.03971	4.183	3.18e-05	***
I(<i>satisfy</i> ²)	0.25840	0.10182	2.538	0.011333	*
of	0.40250	0.11037	3.647	0.000282	***
gov	0.23492	0.13147	1.787	0.074321	.

Residual standard error: 1.068 on 839 degrees of freedom

Multiple R-squared: 0.1319 Adjusted R-squared: 0.1257

F-statistic 21.25 on 6 and 839 DF p-value: < 2.2e-16

Вывод: по полученным данным(см. таблицу 19) можно предположить, что заработок респондентов ограниченных на «подмножестве

2» так же сильно зависит от возраста - с увеличением возраста респонденты статистически зарабатывают меньше. Посмотрим на другие регрессоры. Теперь регрессоры gov & $I(satisfy^2)$ уже не оказывают такое сильное влияние на описываемую переменную, что с одной стороны весьма странно, ведь удовлетворенность работой должна весьма хорошо описывать переменную $salary$, возможно, городское население не так сильно заботится своей удовлетворенности от работы, закрывая на многое глаза, в пользу высокой зарплаты. Наличие высшего образования и возраст так же сильно положительно и отрицательно соответственно влияют на зарплату респондентов.

4 Практическая работа №4

1. Набор данных: [Credit Card customers](#)

4.1 Описание данных

Представим таблицу, которая описывает значения переменных из текущего набора данных. Смотри таблицу [20](#).

4.2 Введение

Необходимо по указанному набору данных классифицировать всех респондентов по признаку.

4.3 Решение

Для решения задачи необходимо выполнить следующие пункты:

1. Обработка пропущенных значений
2. Нормализация признаков
3. Разбиение данных на тестовую и обучающую выборки

Таблица 20: Описание переменных текущего набора данных.

Название переменной	Описание
CLIENTNUM	Уникальный идентификатор клиента
Attrition_Flag	Активность клиента
Customer_Age	Возраст клиента в годах
Gender	М=Мужчина, F=Женщина
Education_Level	Уровень образования клиента
Marital_Status	Состоит ли респондент в браке
Income_Category	Категория годового дохода владельца счета
Credit_Limit	Кредитный лимит по Кредитной карте
Total_Trans_Amt	Общая сумма транзакции
Avg_Open_To_Buy	Открыта кредитная линия на покупку
Total_Revolving_Bal	Общий Возобновляемый остаток на Кредитной карте
Total_Amt_Chng_Q4_Q1	Изменение суммы транзакции (4 квартал по сравнению с 1 кварталом)
Total_Trans_Ct	Общее количество транзакций
Total_Ct_Chng_Q4_Q1	Изменение количества транзакций (4 квартал по сравнению с 1 кварталом)
Avg_Utilization_Ratio	Средний Коэффициент Использования Карты

4. Построение классификатора методом опорных векторов

5. Построение классификатора типа Случайный Лес

Решение первых трех пунктов вы можете найти в разделе [Приложение](#) в полном коде решения. Здесь же мы приведем полученные результаты(см. рисунки 1, 3).

Вернемся к задаче классификации. Построим классификатор методом опорных векторов.

```

1 X_train, X_test, y_train, y_test = train_test_split(std_bank_df.iloc
   [:,1:],
2 std_bank_df.iloc[:,0], test_size=0.25, random_state=42)
3 svm_clf = svm.SVC()
4
5 # Поиск наилучших параметров для классификации методом опорных вектор
    ов
6 svm_param = {'C': [0.1, 1, 10, 100],
7               'gamma': ['scale', 'auto'],
8               'kernel': ['rbf', 'poly', 'sigmoid'],
9               'probability':[True],
10              'class_weight':['balanced', 'none']}
```

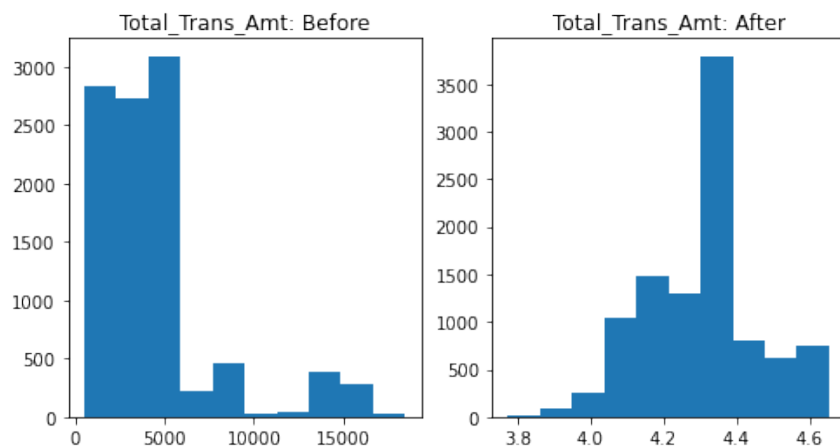


Рис. 1: Результат ноормализации признака Total_Trans_Amt.

```

11 svm_grid = GridSearchCV(svm_clf, svm_param, n_jobs=-1, cv=5, verbose
    =1)
12 svm_grid.fit(X_train, y_train)
13 print("Best param: ", svm_grid.best_params_)
14 svm_pred = svm_grid.predict(X_test)
15 svm_pred_prob = svm_grid.predict_proba(X_test)[:,-1]
16 svm_pred_df = pd.DataFrame({"pred":svm_pred, "prob":svm_pred_prob, "
    actual":y_test})
17 # manually adjust pred based on prob
18 thresh = np.quantile(svm_pred_prob, (1-y_train.mean()))
19 svm_pred_df['pred'] = (svm_pred_df['prob']>thresh).astype(int)
20 svm_pred_df
21
22 X, y = std_bank_df.iloc[:,1:], std_bank_df.iloc[:,0]
23
24

```

Листинг 14: Реализация метода опорных векторов. Поиск наилучших параметров.

Получили, что лучшие параметры для данного классификатора

- class_weight = 'balanced'
- gamma='scale'
- kernel='rbf'
- probability=True

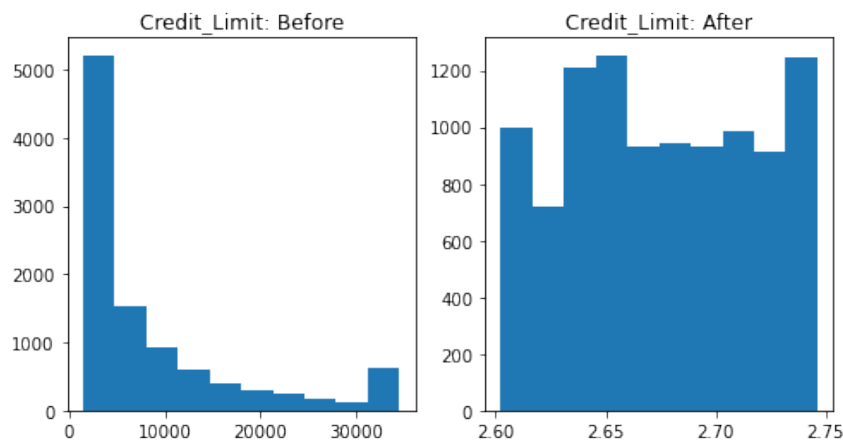


Рис. 2: Результат ноормализации признака Credit_Limit.

```

1 estimator = svm.SVC(C=10,
2                 class_weight='balanced',
3                 gamma='scale',
4                 kernel='rbf',
5                 probability=True)

```

Листинг 15: Реализация метода опорных векторов.

Тогда построив классификатор с данными параметрами получим следующие значения:

- accuracy: 0.9048280593431095
- f1: 0.7060378855046817
- precision: 0.7263938934825012
- recall: 0.6875215146299484

Точность на валидации 90%, что значит, что классификатор неплохо разделил всех респондентов по выделенному признаку. Посмотрим на другие метрики. Из всех положительных результатов наша модель верно описала лишь 68%, а нашел из всех положительных только 72%. Этот классификатор вполне может использоваться для построения линейной регрессии, но не стоит рассчитывать на полноценную точность. Теперь построим классификатор типа Случайный Лес.

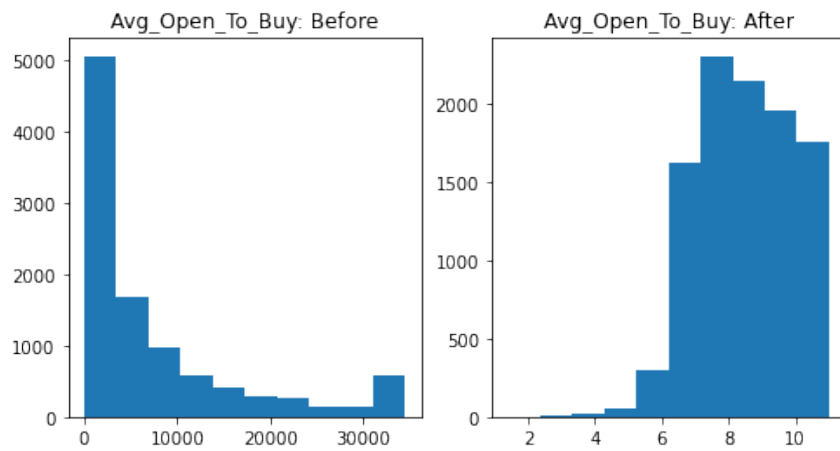


Рис. 3: Результат нормализации признака Avg_Open_To_Buy.

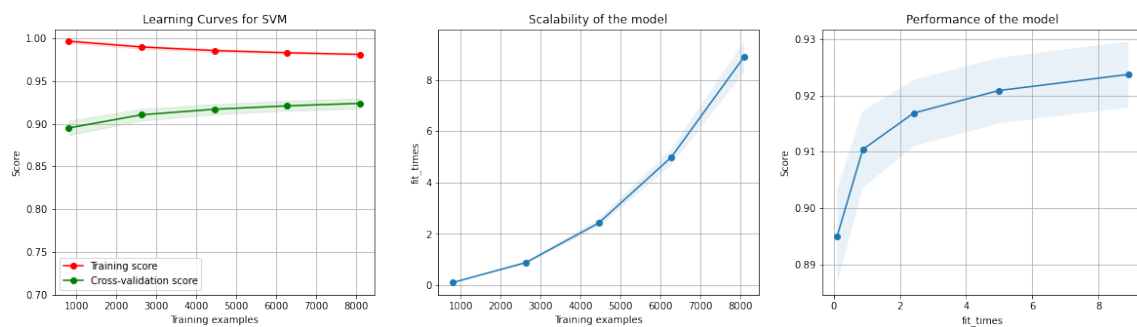


Рис. 4: Результат классификатора SVM.

```

1 rf = RandomForestClassifier()
2 n_estimators = [100, 200, 300]
3 max_features = [0.5, 0.25, 'log2', 'sqrt']
4 max_depth = [100, 200, 'none']
5 min_samples_split = [2, 5, 10]
6 min_samples_leaf = [1, 2, 4]
7 bootstrap = [True, False]
8 class_weight = ['balanced', 'none']
9 rf_param = {'n_estimators': n_estimators,
10             'max_features': max_features,
11             'max_depth': max_depth,
12             'min_samples_split': min_samples_split,
13             'min_samples_leaf': min_samples_leaf,
14             'bootstrap': bootstrap,
15             'class_weight': class_weight}
16 rf_grid = GridSearchCV(rf, rf_param, n_jobs=-1, cv=5, verbose=1)

```

```
17 rf_grid.fit(X_train, y_train)
```

Листинг 16: Реализация классификатора Случайный Лес. Поиск наилучших параметров.

Таким образом были выделены параметры, дающие наилучшую точность:

- bootstrap=False
- class_weight='balanced'
- max_depth=200
- max_features=0.25
- min_samples_leaf=1
- min_samples_split=2
- n_estimators=300

По полученным параметрам построим классификатор типа Случайный Лес.

```
1 estimator = RandomForestClassifier(bootstrap=False,
2                                     class_weight='balanced',
3                                     max_depth=200,
4                                     max_features=0.25,
5                                     min_samples_leaf=1,
6                                     min_samples_split=2,
7                                     n_estimators=300)
8 estimator.fit(X_train, y_train)
```

Листинг 17: Реализация классификатора Случайный Лес.

Получим следующие результаты:

- accuracy: 0.9407590180165432
- f1: 0.8085149959922957
- precision: 0.888746562219956

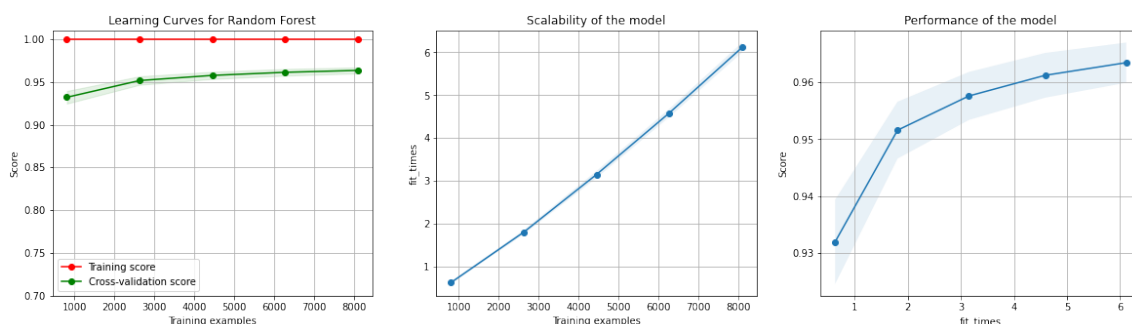


Рис. 5: Результат классификатора Random Forest.

- recall: 0.7494549627079747

Оказалось, что точность второго классификатора лучше (см. рисунки 4,5). Из всех названных классификатором положительными объектов действительно оказались такими 88%, а всего классификатор распознал 75% положительных объектов. Конечно, в сравнении с первым классификатором, второй оказался куда лучше, но второй также и более затратный по времени исполнения. Все же алгоритм использования многочисленного ансамбля решающих деревьев, восполняя недостаток точности большим количеством, в этой задаче классификации показала себя с хорошей стороны, в то время как, разбиение объектов гиперплоскостями не было столь же эффективным. Полный код решения и все представленные и дополнительные материалы см. в разделе [Приложение](#).

5 Практическая работа №5

1. Набор данных: [Medical Cost Personal Datasets](#)

Отметим, что набор данных создан на основе демографической статистики Бюро переписи населения США, согласно книге, из которой он взят, поэтому эти данные можно приближенно считать реальными.

5.1 Решение

5.1.1 Введение в данные

- В выбранном датасете 1338 объектов и 7 признаков. Посмотрим на начальные данные см. таблицы [21](#), [22](#), [23](#).

Таблица 21: Представление начальных данных.

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520
5	31	female	25.740	0	no	southeast	3756.62160
6	46	female	33.440	1	no	southeast	8240.58960

Таблица 22: Представление начальных данных.

#	Column	Non-Null	Count	Dtype	smoker	region	charges
0	age	1338	non-null	int64	yes	southwest	16884.92400
1	sex	1338	non-null	object	no	southeast	1725.55230
2	bmi	1338	non-null	float64	no	southeast	4449.46200
3	children	1338	non-null	int64	no	northwest	21984.47061
4	smoker	1338	non-null	object	no	northwest	3866.85520
5	region	1338	non-null	object	no	southeast	3756.62160
6	charges	1338	non-null	float64	no	southeast	8240.58960

dtypes:

float64(2), int64(2), object(3)

Теперь представим описание признаков из текущего набора данных.

- **age** - возраст основного бенефициара
- **sex** - пол бенефициара: женщина или мужчина
- **bmi** - индекс массы тела
- **children** - количество застрахованных людей

Таблица 23: Разбиение данных по категориальному признаку **region**.

southeast	364
southwest	325
northwest	325
northeast	324
<hr/>	
Name: region, dtype: int64	

- **smoker** - является ли бенефициар курильщиком
- **region** - район, в котором проживает бенефициар
- **charges** - индивидуальные расходы, оплачиваемые медицинским страхованием

5.1.2 Представление признаков

• Категориальные признаки представленные в датасете- **region**, **smoker**, **sex**. Признак 'region' имеет наибольшее количество уникальных значений.

- В датасете имеются бинарные признаки: **sex**, **smoker**
- В датасете также имеются числовые признаки **age**, **children**, **bmi**, **charges**.

Обработаем категориальные признаки.

```

1 region = pd.get_dummies(data['region'])
2 data = pd.concat((data, region), axis=1)
3 data = data.loc[:, data.columns.isin(['age', 'sex', 'bmi', 'children',
4   'smoker', 'charges', 'northeast', 'northwest', 'southeast', 'southwest'])]
```

Листинг 18: Обработка категориальных признаков.

5.1.3 Обработка пропущенных значений

- Проверим данные на наличие пропусков см. таблицу [24](#).
- Всего в датасете представлены данные о 1338 клиентах, в каждой строке, отвечающей одному из признаков по 1338 значений, значит

Таблица 24: Представление данных.

	Column	Non-Null	Count	Dtype
0	age	1338	non-null	int64
1	sex	1338	non-null	object
2	bmi	1338	non-null	float64
3	children	1338	non-null	int64
4	smoker	1338	non-null	object
5	charges	1338	non-null	float64
6	northeast	1338	non-null	uint8
7	northwest	1338	non-null	uint8
8	southeast	1338	non-null	uint8
9	southwest	1338	non-null	uint8
dtypes: float64 int64(2), object(2), uint8(4)				

можно сделать вывод о том, что в датасете отсутствуют пропуски. Значит нам не придется их отдельно обрабатывать.

5.1.4 Определение и обработка аномальных значений и выбросов

- Проверим датасет на наличие в нем аномальных значений.

Таблица 25: Проверка данных на наличие аномальных значений.

	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

Как можно видеть из представленной таблицы [25](#), среднее значение расходов по медицинской страховке - 13270.4 пунктов, но макси-

мальное значение этого же столбца - 63770.4 пункта, что превышает среднее значение почти в 5 раз. Можно утверждать, что это и есть искомое аномальное значение. Кроме того, стоит обратить внимание на столбец **bmi** - индекс массы тела, который у желательного должен быть в рамках от 18.5 до 24.9, хотя из таблицы видно, что даже у четверти клиентов нет таких показателей. Среднее значение этого показателя 30.7, что превышает норму более чем на 5 пунктов. В целом можно сказать, что большинство клиентов имеют ожирение. Давайте посмотрим на это более детально.

- Сравним рост индекса массы тела с ростом возраста и с ростом индивидуальных расходов по медицинской страховке.

```

1 plt.figure(figsize=(12,10), dpi= 80)
2 sns.heatmap(data.corr(), xticklabels=data.corr().columns,
3 yticklabels=data.corr().columns, cmap='RdYlGn', center=0, annot=
4 True)
5
6 plt.xticks(fontsize=12)
7 plt.yticks(fontsize=12)
8 plt.show()

```

Листинг 19: Сравнение роста индекса массы тела с ростом возраста.

- В итоге, см. рис 8 и 7 и таблицу 26, никакой простой зависимости роста индекса тела от возраста или от медицинских расходов выявить не удалось, это значит, что нет повышения или понижения возраста или медицинских расходов связанных индексом массы тела. Т.е. другими словами, практически каждый клиент имеет индекс массы тела выше нормы, но медицинские расходы не связаны с лечением ожирения, значит это является нормальным в районах, где они живут.

5.1.5 Стандартизация значений признаков

Произведем стандартизацию значений признаков

```

1 data['sex'] = np.where(data['sex'] == 'female' , 0, data['sex'])
2 data['sex'] = np.where(data['sex'] == 'male' , 1, data['sex'])
3
4 data['smoker'] = np.where(data['smoker'] == 'no' , 0, data['smoker']
5 )

```

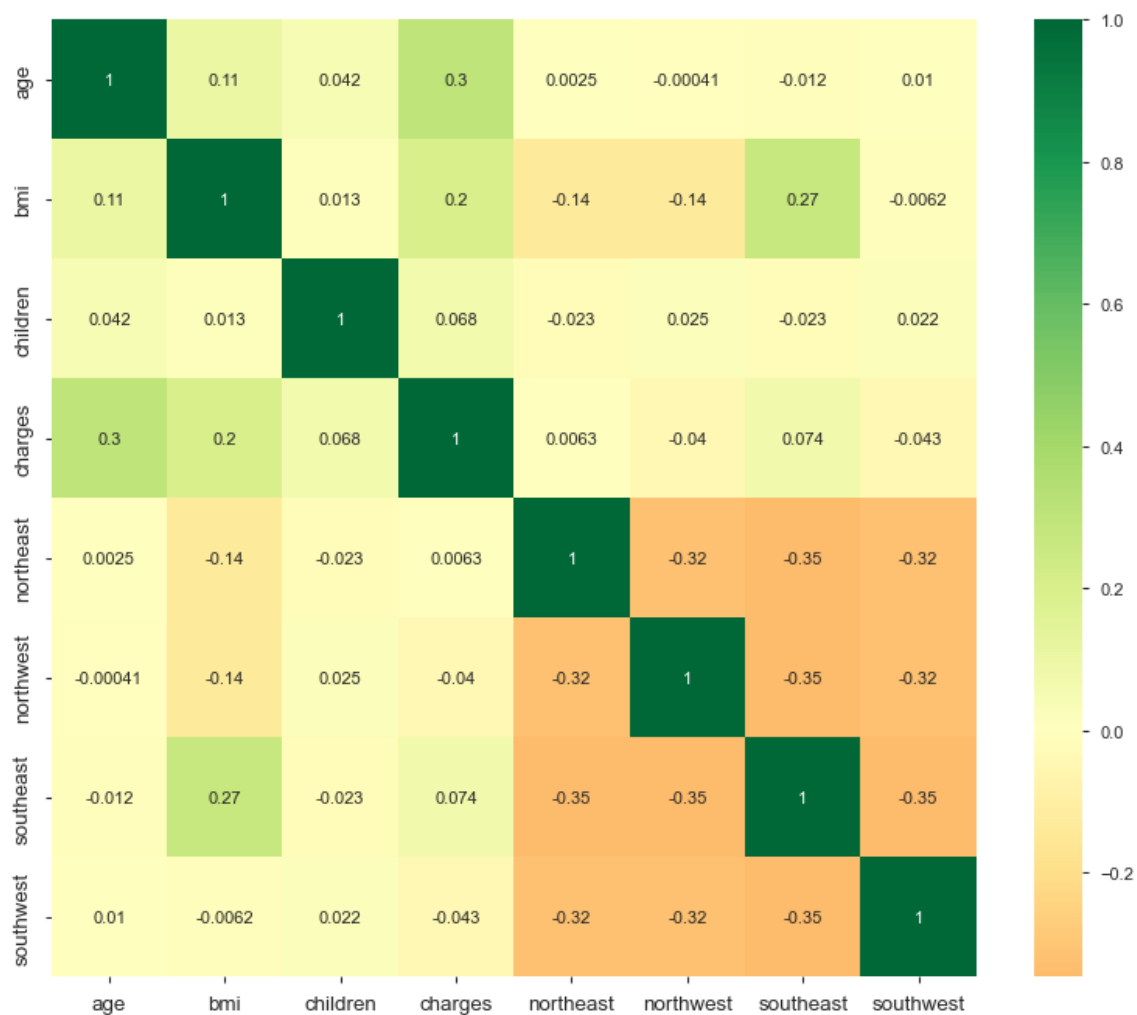



Рис. 6: Корреляция признаков датасета.

```

5 data['smoker'] = np.where(data['smoker'] == 'yes' , 1, data['smoker']
6   )
7
8 from sklearn import preprocessing
9 import seaborn as sns # Для графиков
10
11 # Инициализируем стандартизатор
12 s_scaler = preprocessing.StandardScaler()
13
14 # Копируем исходный датасет
15 data_s = s_scaler.fit_transform(data)

```

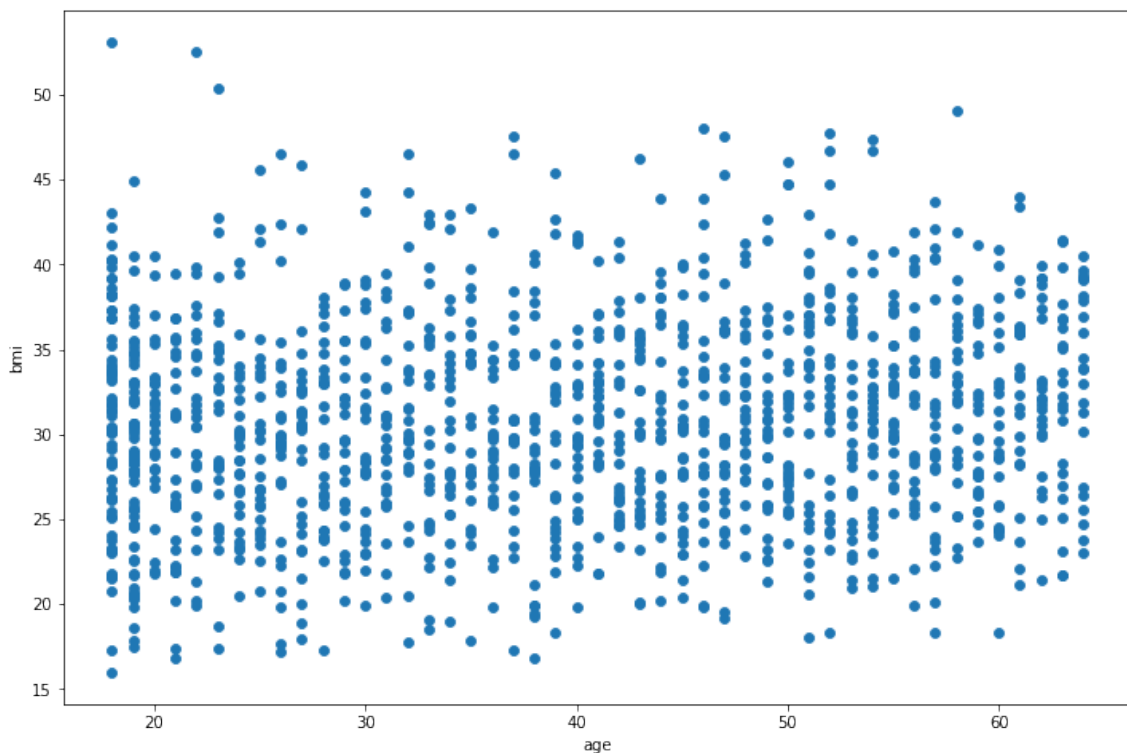


Рис. 7: График зависимости индекса массы тела от медицинских расходов.

```

16
17 # Копируем названия столбцов, которые теряются при использовании
    fit_transform()
18 col_names = list(data.columns)
19
20 # Преобразуем промежуточный датасет в полноценный датафрейм для визу
    ализации
21 data_s = pd.DataFrame(data_s, columns = col_names)
22
23

```

Листинг 20: Стандартизация значений признаков.

Визуализируем стандартизацию признаков, см. рис. [8 - 8](#)

5.1.6 Определение целевого признака

charges - целевой признак. Исключим его из данных.

Таблица 26: Результат работы функции `data.corr()`.

	age	bmi	children	charges	northeast	northwest	southeast	southwest
age	1.000000	0.109272	0.042469	0.299008	0.002475	-0.000407	-0.011642	0.010016
bmi	0.109272	1.000000	0.012759	0.198341	-0.138156	-0.135996	0.270025	-0.006205
children	0.042469	0.012759	1.000000	0.067998	-0.022808	0.024806	-0.023066	0.021914
charges	0.299008	0.198341	0.067998	1.000000	0.006349	-0.039905	0.073982	-0.043210
northeast	0.002475	-0.138156	-0.022808	0.006349	1.000000	-0.320177	-0.345561	-0.320177
northwest	-0.000407	-0.135996	0.024806	-0.039905	-0.320177	1.000000	-0.346265	-0.320829
southeast	-0.011642	0.270025	-0.023066	0.073982	-0.345561	-0.346265	1.000000	-0.346265
southwest	0.010016	-0.006205	0.021914	-0.043210	-0.320177	-0.320829	-0.346265	1.000000

```

1 target = data.charges
2 train = data.drop(['charges'], axis=1)
3

```

Листинг 21: Исключение признака `charges` из данных.

5.1.7 Разбиение данных на выборки

Поскольку параметр `test_size` имеет значение 0.3, а все объекты разделяются тренировочную и обучающую выборки, то количество объектов в тренировочной выборке будет равно 0.7 от общего числа объектов. Параметр `random_state` отвечает за начальное значение случайного числа, используемого для перетасовки т.е. чтобы сделать эксперимент воспроизводимым.

5.1.8 Определение зависимости между описывающими признаками

Проверим линейную зависимость между введенными признаками с помощью инструментов библиотеки **pandas**.

```

1 data.corr()
2

```

Листинг 22: Проверка корреляции признаков между собой.

Визуализация текущих действий представлено на рис. 8 и в таблице 26. Таким образом, мы понимаем, что есть некоторая зависимо-

сти между признаками **charges** и **smoker**. Будем иметь это ввиду, если в будущем мы будем описывать ими целевой признак.

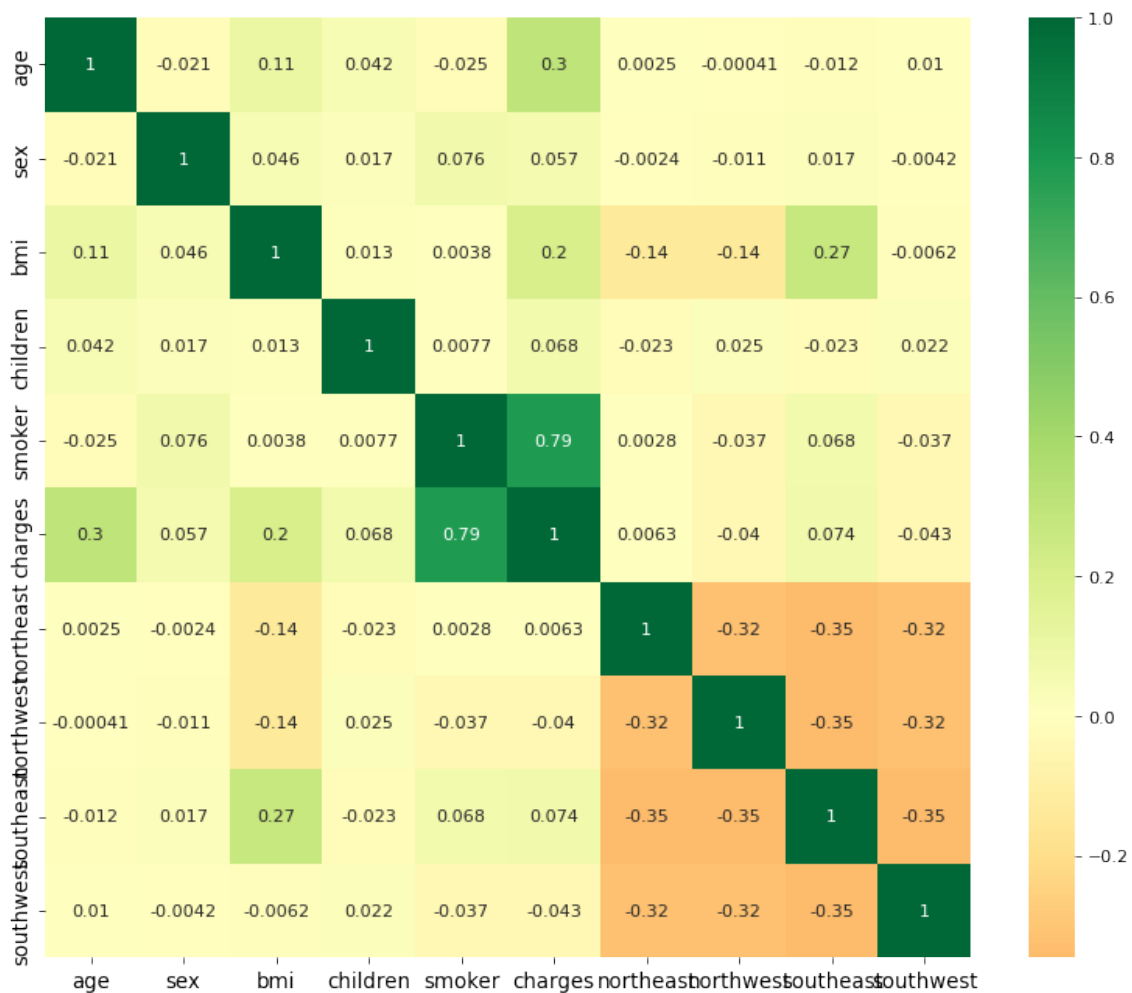


Рис. 8: Корреляция признаков датасета.

Теперь разделим данные на обучающую и тестовую выборки.

```

1 from sklearn.model_selection import train_test_split
2 X_train, X_test, y_train, y_test = train_test_split(train, target,
3     test_size = 0.25, random_state = 42)
4 N_train, _ = X_train.shape
5 N_test, _ = X_test.shape
6 print (N_train, N_test)

```

Листинг 23: Разделение данных на выборки.

5.1.9 Построение линейной зависимости

Для построения линейной зависимости признака **charges** от регрессоров представленных в таблице 26, исключая столбец с выбранным целевым признаком, воспользуемся методом главных компонент.

```

1 pca = PCA()
2 pca.fit(X_train)
3 X_pca = pca.transform(X_train)
4 for i, component in enumerate(pca.components_):
5     print("{} component: {}% of initial variance".format(i + 1,
6         round(100 * pca.explained_variance_ratio_[i], 2)))
7     print(" + ".join("%.3f x %s" % (value, name)
8         for value, name in zip(component, train.columns)
9     ))
10    print('\n')

```

Листинг 24: Реализация метода главных компонент.

Мы получили, что для описания 90% дисперсии данных достаточно 6-х компонент, а наибольший вклад вносит признак **southeast**. Это значит, что для людей, проживающих на юго-востоке США, медицинские страховки самые дорогие. Что на первый взгляд очень странно. Проведя дополнительные исследования в интернете, обнаруживается, что согласно источнику <https://www.talktomira.com/post/the-10-most-expensive-health-insurance-markets-in-the-u-s> 8 из 10 штатов США с самыми дорогими медицинскими страховками расположены на либо на юго-востоке страны, либо на северо-востоке. Значит, мы действительно получили достоверные результаты, и наша модель правильно описывает динамику стоимости медицинской страховки. Визуализацию зависимости описания дисперсии от количество описываемых признаков см. на рис. 9

Все приложенные таблицы и рисунки, а также код всей программы см. в разделе [Приложение](#).

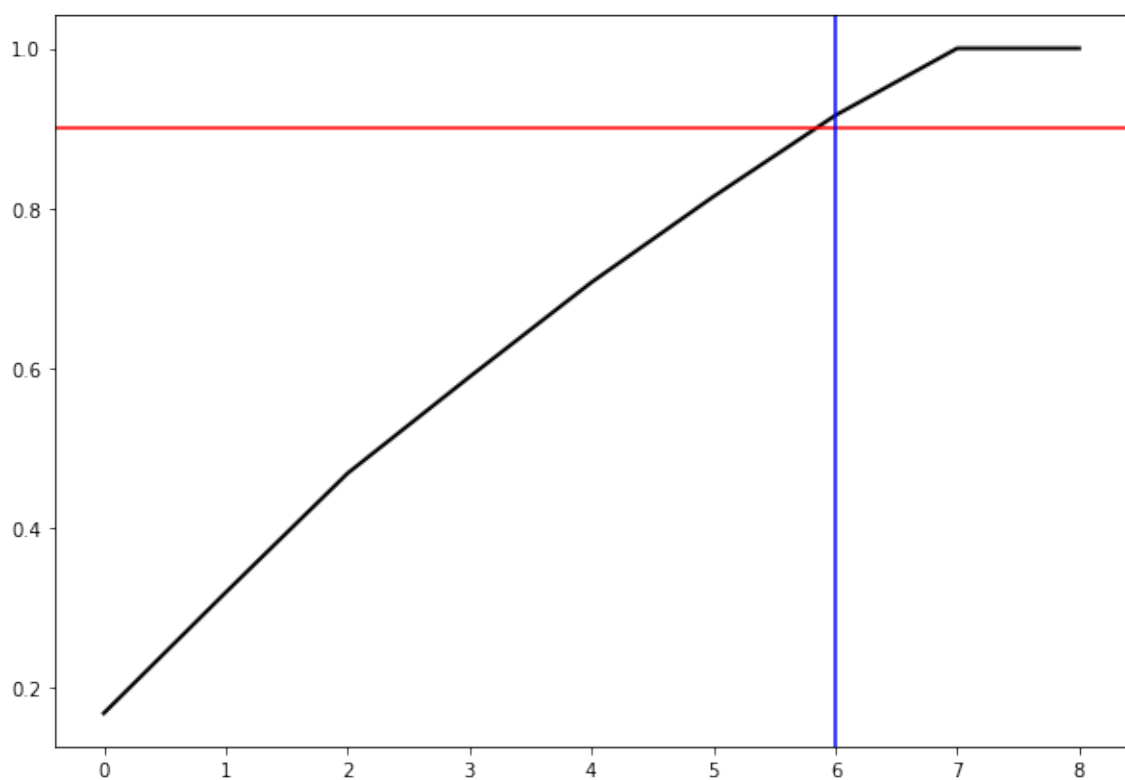


Рис. 9: Визуализация зависимости качества описываемой дисперсии от количество описывающих признаков.

6 Практическая работа №6

В практической работе №5 мы провели начальную обработку [данных](#) и с помощью метода главных компонент продемонстрировали сильную линейную зависимость между целевым признаком **charges** и признаком **southeast**. Поскольку мы получили сильную зависимость между стоимостью медицинской страховки и регионом, исследуем эту тему более детально, попутно выделив другие компоненты, оказывающие наибольшее влияние на стоимость.

6.1 Решение

В практической работе №5 уже были выполнены все необходимые для решения задачи шаги, поэтому сделав приведем лишь некоторые визуальные результаты фрагментов кода (полный код решения задачи см. в разделе [Приложение](#)) и их анализ.

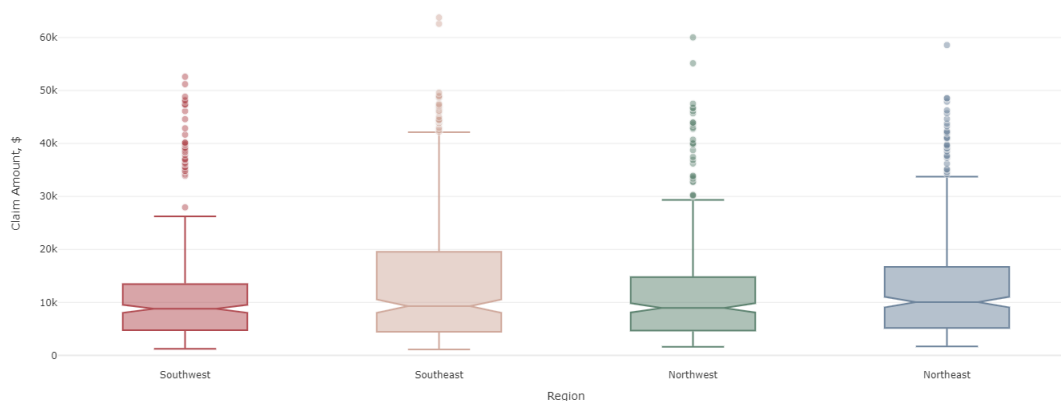


Рис. 10: Визуализация стоимости медицинской страховки к каждому региону.

Таким образом из рис. 10 мы еще раз убеждаемся, что в юго-восточной части США самая дорогая медицинская страховка, но в тоже время в северо-восточной части стоимость страховки не мно-

гим ниже и если посмотреть, то в северо-восточном регионе самая высокая средняя цена за медицинскую страховку.

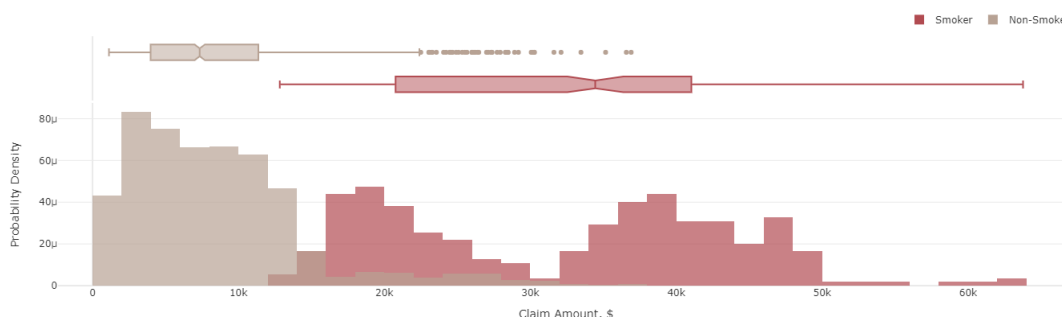


Рис. 11: Визуализация распределения стоимости медицинской страховки среди курящих и некурящих.

Здесь же (см. рис 11) можно заметить весьма важную деталь. Стоимость медицинской страховки курильщиков значительно выше и вариативнее, чем у не курящих людей. Далее окажется, что этот фактор является самым значимым среди всех остальных.

Выясним действительно ли фактор курения настолько сильно влияет на стоимость медицинской страховки.

Представим график зависимости курящих и нет мужчин от женщин от стоимости медицинской страховки.

Удивительно, из рис. 12 видна следующая тенденция. В каждом из регионов некурящие женщины платят за медицинскую страховку больше, чем мужчины, возможно, в стоимости каждой из страховок учитываются возможные будущие расходы на роды женщины. Однако, если мужчина и женщина курят, то стоимость медицинской страховки для мужчины выше, чем у женщины. Кроме того, стоит обратить внимание на стоимость медицинской страховки у курящих и нет. Первые платят больше в **4-5 раз!**

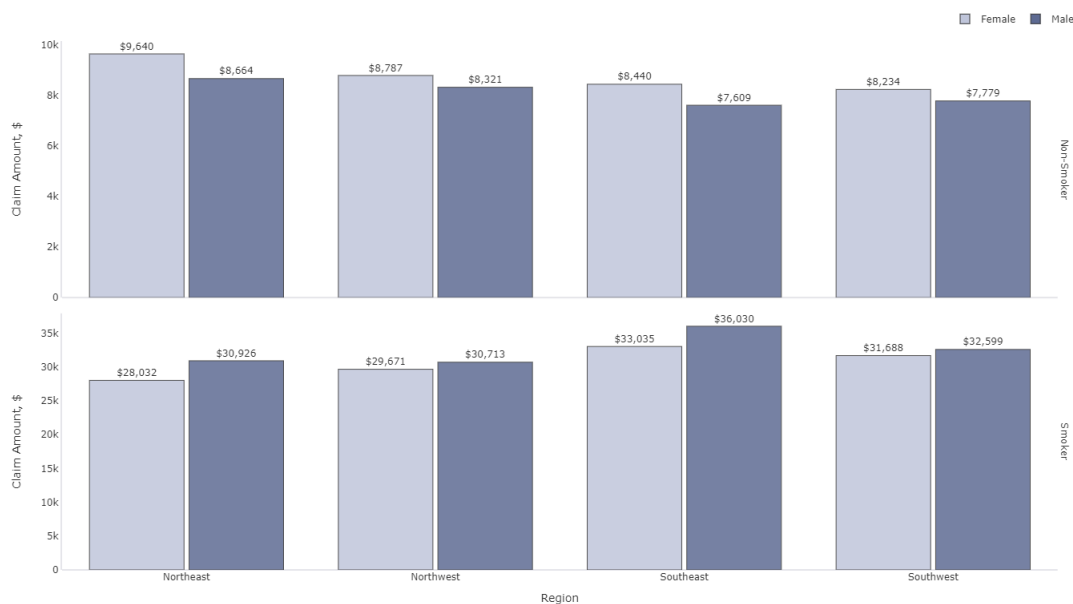


Рис. 12: Визуализация распределения стоимости медицинской страховки.

Построим линейные регрессии зависимости параметра **charges** от введенных признаков и определим признак, который оказывает наибольшее влияние.

- Как выяснилось в практической работе №5 есть существенная зависимость описываемой переменной **charges** от регрессора **smoke**, поэтому обязательно включим последний признак в модель. Но других существенных связей между параметрами нет. Поэтому для описания переменной **charges** мы будем использовать все признаки.

- Перейдем к построению линейных регрессий.

Для начала построим зависимость от всех параметров.

```

1 y_train=data.pop('charges')
2 X_train = data.copy()
3
4 import statsmodels.api as sm
5 X_train_sm = sm.add_constant(X_train)
6
7 all= sm.OLS(y_train, X_train_sm).fit()
8
9 all.summary()
```

Таблица 27: Линейная зависимость параметра **charges** от всех параметров.

	coef	std err	t	P> t	[0.025	0.975]
const	-1.013e+04	791.569	-12.792	0.000	-1.17e+04	-8572.628
age	256.8564	11.899	21.587	0.000	233.514	280.199
bmi	339.1935	28.599	11.860	0.000	283.088	395.298
children	475.5005	137.804	3.451	0.001	205.163	745.838
female	131.3144	332.945	0.394	0.693	-521.842	784.470
smoker_yes	2.385e+04	413.153	57.723	0.000	2.3e+04	2.47e+04
northeast	-1944.3632	334.580	-5.811	0.000	-2600.725	-1288.001
northwest	-2297.3271	335.065	-6.856	0.000	-2954.641	-1640.014
southeast	-2979.3852	386.076	-7.717	0.000	-3736.771	-2222.000
southwest	-2904.4142	352.330	-8.243	0.000	-3595.597	-2213.231

Таблица 28: Параметры модели all

R-squared:	0.749
Adj. R-squared:	0.748
F-statistic:	496.0

10

Листинг 25: Линейная зависимость параметра **charges** от всех параметров.

Приведем таблицы 27, 28 с результатами, построенной модели *all*.

Таким образом, модель *all* описывает до 74,8% различий в стоимость медицинской страховки. Стоит обратить внимание на регрессор **smoker_yes**, которые имеет большой коэффициент, что в очередной раз подтверждает его важность в нашей модели. Однако отметим, что регрессеры, отвечающие за регион имеют отрицательные значения коэффициентов в построенной модели, что, на самом деле, противоречит тому, что было получено ранее через анализ зависимости описываемой переменнo и регрессорами, отвечающими за регион.

Распределим полученные результаты по регионам см. рис. 13.

```
1 for i in range(0,4):
```

```

2     actuals[i].loc[:, 'index']=regions[i]
3 actual = pd.concat([actuals[i] for i in range(4)], axis = 0)
4 pred = pd.concat([preds[i] for i in range(4)], axis = 0)
5 df = pd.concat([actual, pred], axis=1).reset_index(drop=True)
6 col = ["#B14B51", '#D0A99C', '#5D8370', '#6C839B']
7 fig = px.scatter(df, x="actuals", y="preds", color="index",
8                 trendline="ols", height=700,
9                 title="Actual vs Predicted Insurance Costs by
10                    Region,<br>Linear Regression with Principal Component Analysis",
11                 color_discrete_sequence=col, opacity=0.7, facet_col
12                 ='index', facet_col_wrap=2)
13
14 fig.for_each_annotation(lambda a: a.update(text=a.text.split("=")
15 [-1]))
16 fig.add_annotation(text="Adj. R-Squared = {:.1f}%<br>RMSE = {:.0f}"
17 .format(adj_r2_scores[0]*100,rmse[0]),
18                 x=51e3,y=15e3, row=2,col=1, showarrow=False)
19 fig.add_annotation(text="Adj. R-Squared = {:.1f}%<br>RMSE = {:.0f}"
20 .format(adj_r2_scores[1]*100,rmse[1]),
21                 x=51e3,y=15e3, row=2,col=2, showarrow=False)
22 fig.add_annotation(text="Adj. R-Squared = {:.1f}%<br>RMSE = {:.0f}"
23 .format(adj_r2_scores[2]*100,rmse[2]),
24                 x=51e3,y=15e3, row=1,col=1, showarrow=False)
25 fig.add_annotation(text="Adj. R-Squared = {:.1f}%<br>RMSE = {:.0f}"
26 .format(adj_r2_scores[3]*100,rmse[3]),
27                 x=51e3,y=15e3, row=1,col=2, showarrow=False)
28
29 fig.show()

```

Листинг 26: Распределение результатов модели **all** по регионам.

Как видно из рис. 13 наша линейная регрессия хорошо описывает поведение стоимости медицинской страховки в юго-восточном регионе, но достаточно плохо в остальных с большой долей ошибок.

Далее для краткости приведем таблицу других построенных моделей (полный код см. в разделе [Приложение](#)) и сделаем выводы.

Всего было построено 23 линейных моделей. В таблице приведены ключевые для нашего анализа, все построенные регрессии находятся в разделе [Приложение](#). В итоге, мы получили, что наилучшая модель это **lm_16**, где были использованы все признаки, но признак **bmi** был заменен на свой логарифм, а признак **age** на свой квадрат. После того, как из модели были убраны регрессоры **bmi** или **smoker_yes** показатель R-squared падает ниже 1%. Т.е. эти ре-

Таблица 29: Таблица, представляющая построенные линейные модели.

Название модели	Описание	R-squared	Adj. R-squared
bmi_sq	bmi^2	0.749	0.748
age_sq	age^2	0.754	0.752
...
bmi_age_sq	$\text{bmi}^2, \text{age}^2$	0.752	0.750
age_lg	$\log(\text{age}, 2)$	0.664	0.662
bmi_lg	$\log(\text{bmi}, 2)$	0.751	0.750
bmi_age_lg	$\log(\{\text{age}, \text{bmi}\}, 2)$	0.665	0.663
lm_15	$\log(\text{age}, 2), \text{bmi}^2$	0.660	0.659
lm_16	$\log(\text{bmi}, 2), \text{age}^2$	0.754	0.753
bmi_smoke	$\text{bmi} + \text{smoke_yes}$	0.658	0.657
age_smoke	$\text{age} + \text{smoke_yes}$	0.721	0.721
smk_sq	smoke_yes^2	0.751	0.749
lm_12	$\text{age} + \text{bmi}$	0.117	0.116

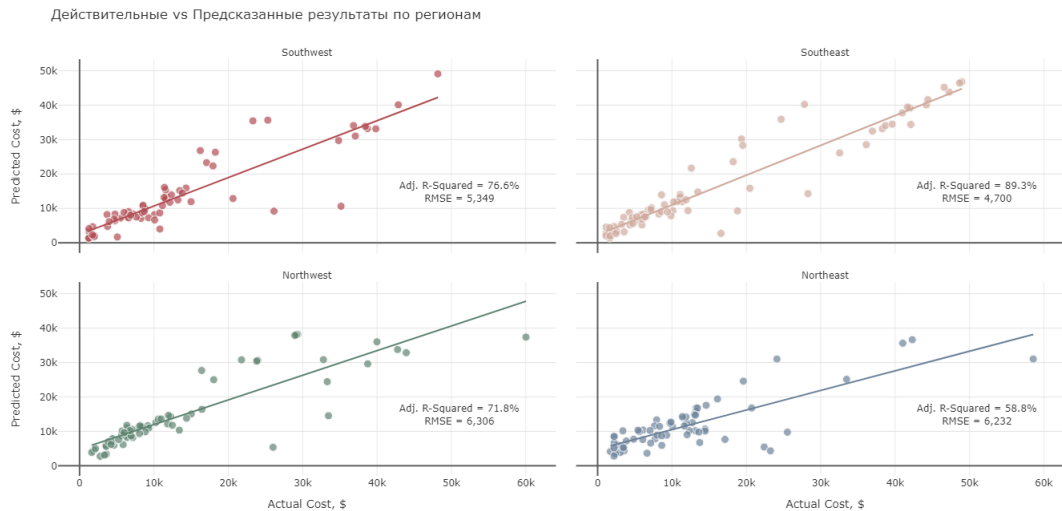


Рис. 13: Распределение результатов модели **all** по регионам.

грессоры существенно влияют на результаты модели, т.е. на саму описываемую переменную.

- Теперь построим модель с теми же регрессорами, но с помощью градиентного бустинга.

```

1 X_train = pd.DataFrame(data=s.fit_transform(X_train), columns=
  X_pf.columns)
2 X_test = pd.DataFrame(data=s.transform(X_test), columns=X_pf.columns
  )
3
4
5 grid = {'learning_rate': [0.0001, 0.001, 0.01, 0.1, 0.25, 0.5],
6         'n_estimators': [int(x) for x in np.linspace(start = 200,
7         stop = 1000, num = 5)],
8         'subsample': [0.5, 0.8, 1],
9         'min_samples_split': [2, 5, 10],
10        'min_samples_leaf': [1, 2, 4],
11        'max_depth': [int(x) for x in np.linspace(2, 10, num = 5)],
12        'max_features': [None, 'sqrt']}
13 xgb=GradientBoostingRegressor(random_state=21)
14 xgb_cv=RandomizedSearchCV(estimator=xgb, param_distributions=grid,
15        scoring='neg_mean_squared_error',
16        n_iter=100, cv=3, random_state=21, n_jobs
17        =-1)
18 xgb_cv.fit(X_train, y_train)
19 y_pred=xgb_cv.predict(X_test)

```

```

17 preds.append(pd.Series(y_pred, name='preds').reset_index(drop=True))
18 rmse=np.sqrt(mean_squared_error(y_test, y_pred)).round(2)
19 r2=r2_score(y_test, y_pred)
20 adj_r2 = 1 - (1-r2)*(len(y_test)-1)/(len(y_test)-X_test.shape[1]-1)
21 rmsees.append(rmse)
22 r2_scores.append(r2)
23 adj_r2_scores.append(adj_r2)
24

```

Листинг 27: Построение модели на основе градиентного бустинга.

Посмотрим на результаты.

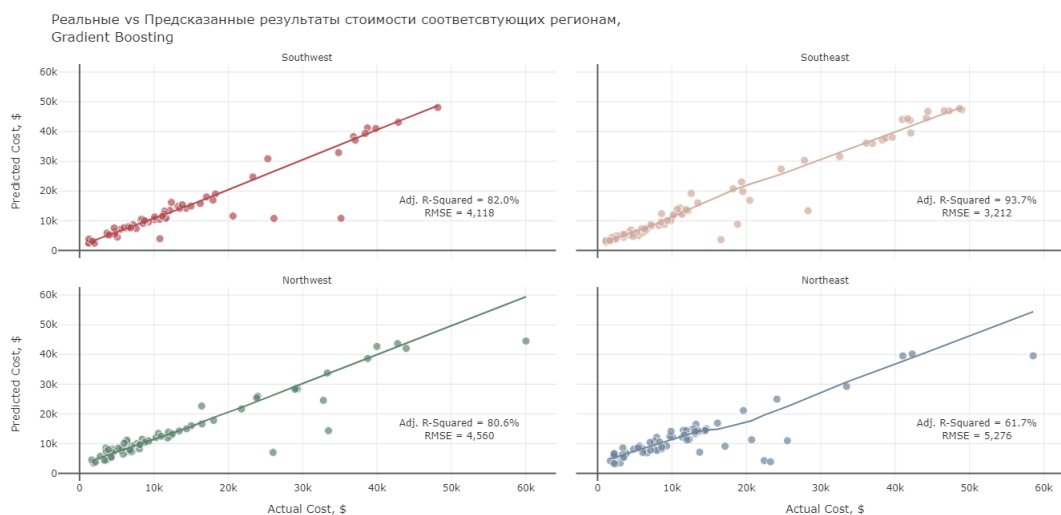


Рис. 14: Визуализация результатов построенной модели *xgb_cv*.

Таким образом нам удалось построить хорошие модели в регионах *Southwest* и *Southeast*, хотя в двух других получить похожие оценки получить не удалось. Однако, для всех регионов среднее значение R-squared больше 80%, поэтому, в целом, можно считать, что построенная с помощью градиентного спуска модель удачная.

Теперь мы приведем усредненные значения значимости отдельных параметров и их композиций при оценке стоимости медицинской страховки (см. рис 15, которые были использованы в модели *xgb_cv*).

Полный код решения см. в разделе [Приложение](#).

Вывод. В представленной практической работе №6 мы выяснили, какие из параметров оказывают наиболее значимое воздействие на рост стоимости медицинской страховки. Как оказалось, наибольшая средняя цена медицинской страховки представлена в северо-восточном регионе, а наибольшая стоимость страховки в юго-восточном. Курящие люди, имеющие ожирение платят за страховку значительно больше остальных, и разница стоимости страховки для курящих и нет весьма значительная. Вторым и третьим по значимости оказались признаки возраста и индекс массы тела, которые так же достаточно сильно влияют на стоимость медицинской страховки. Чем старше или чем больше у человека лишнего веса, тем дороже ему обходится медицинская страховка. Однако признак, определяющий количество детей наоборот не вносит существенной разницы в стоимость, как и гендер клиента.

Значимость факторов при оценке стоимости медицинской страховки

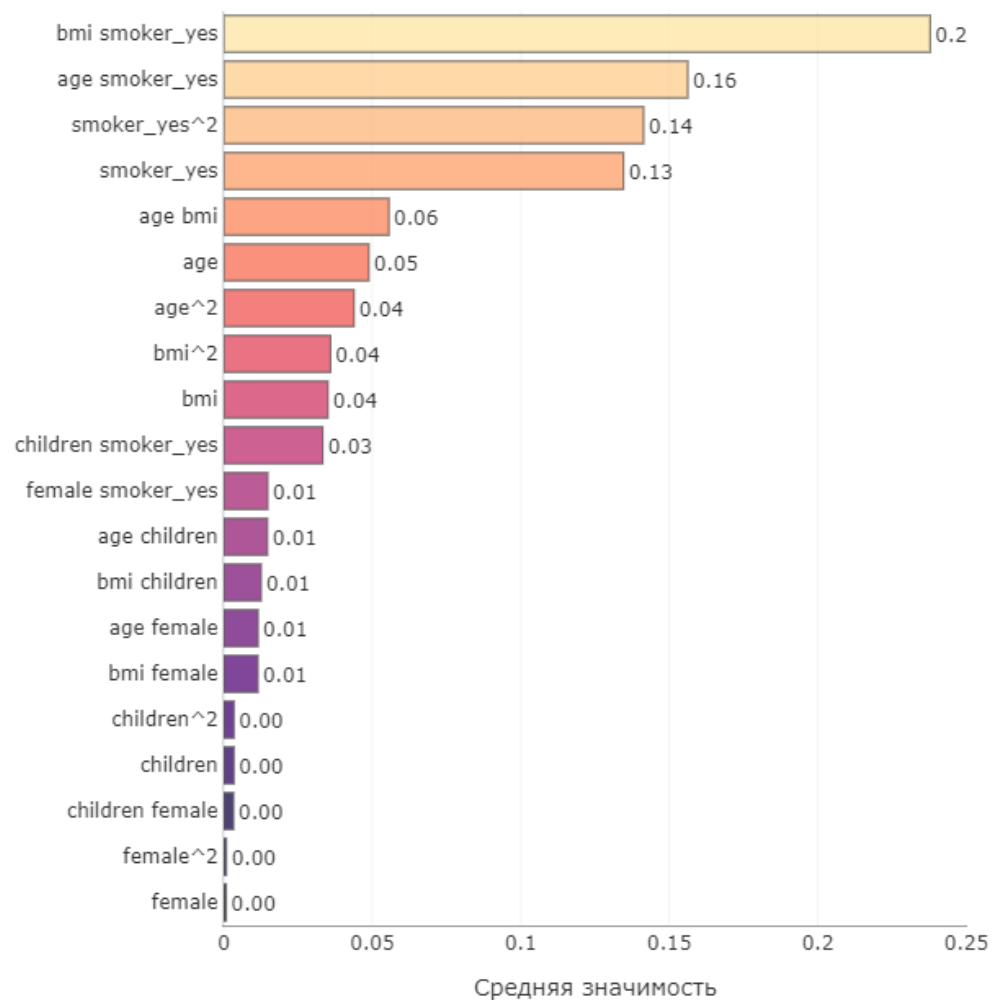


Рис. 15: Визуализация распределения стоимости медицинской страховки.

7 Заключение

В Практической работе №1 были построены линейные регрессии, на основе которых мы выяснили, что к католической церкви в большей мере относили себя те, кто не получил образование выше начальной школы.

В Практической работе №2 были построены доверительные интервалы к регрессорам и линейной модели, тем самым была оценена точность каждого из них и линейной регрессии в целом.

В Практической работе №3 были проанализированы данные обследования РМЭЗ НИУ ВШЭ. Как оказалось наиболее высокие зарплаты были у мужчин средних лет с высшим образованием, родившихся в развитых городах.

В Практической работе №4 с помощью классификатора Random Forest и метода вспомогательных векторов была выполнена классификация респондентов по признаку брачного статуса.

В Практической работе №5 были введены и описаны данные, построенные на основе демографической статистики США. Данные были стандартизированы, был определен целевой признак - стоимость медицинской страховки и была построена линейная модель на основе метода главных компонент, выделяющая из регионов, в которых проходили исследования, область с самой дорогой медицинской страховкой.

В Практической работе №6 на основе данных из Практической работы №5 с помощью линейных регрессий и анализа зависимостей между признаками была получена динамика стоимости медицинской страховки в регионах исследования, а с помощью метода градиентного бустинга выделены самые значимые признаки.

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, Марно. Путеводитель по современной эконометрике/ Марно Вербик - Москва: «Научная книга», 2008. - 616 с.
- [4] Доугерти, Кристофер. Введение в эконометрику/ Кристофер 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.

8 Приложение

```
1 output model_Agr
2 Coefficients:
3           Estimate Std. Error t value Pr(>|t|)
4 (Intercept)    3.8313    13.8966   0.276   0.7840
5 Agriculture     0.7365     0.2508   2.937   0.0052 **
6 ---
7 Residual standard error: 38.63 on 45 degrees of freedom
8 Multiple R-squared:  0.1609, Adjusted R-squared:  0.1422
9 F-statistic: 8.627 on 1 and 45 DF, p-value: 0.005204
10
11 #####
12 output model_Ex
13 Coefficients:
14           Estimate Std. Error t value Pr(>|t|)
15 (Intercept)   90.5137    11.6779   7.751 7.96e-10 ***
16 Examination   -2.9940     0.6388  -4.687 2.59e-05 ***
17 ---
18
19 Residual standard error: 34.56 on 45 degrees of freedom
20 Multiple R-squared:  0.328, Adjusted R-squared:  0.3131
21 F-statistic: 21.97 on 1 and 45 DF, p-value: 2.588e-05
22
```

Листинг 1: Практическая работа 1: Результаты построенных моделей

```
1 library("lmtest")
2 library("GGally")
3 data = swiss # insertion base data
4 summary(data)
5
6 #-----
7 # среднее значение столбцов Catholic, Agriculture,
8 Examination
9 mean(data$Catholic) # 41.14383
10 mean(data$Agriculture) # 50.65957
11 mean(data$Examination) # 16.48936
12
13 # дисперсия(сумма квадратов отклонений всех значений от средне
14 го) столбцов
15 #Catholic, Agriculture, Examination
16 var(data$Catholic) # 1739.295
17 var(data$Agriculture) # 515.7994
18 var(data$Examination) # 63.64662
19
20 # SKO(среднеквадратическое отклонение столбцов Catholic,
21 Agriculture, Examination
```

```

19 sd(data$Catholic)      # 41.70485
20 sd(data$Agriculture)  # 22.71122
21 sd(data$Examination)  # 7.977883
22 #-----
23
24
25 # model_Agr: y- Catholic ~ x1- Agriculture
26 model_Arg = lm(Catholic~Agriculture, data)
27 model_Arg
28 summary(model_Arg)
29
30 model_Ex = lm(Catholic~Examination, data)
31 model_Ex
32 summary(model_Ex)
33

```

Листинг 2: Практическая работа 1: Полный код программы

```

1 model_auxiliary_2 = lm(raises ~ advance, attitude)
2 summary(model_auxiliary_2)
3
4 model_auxiliary_3 = lm(critical ~ advance, data)
5 summary(model_auxiliary_3)
6

```

Листинг 3: Построение линейных зависимостей

```

1 model_all = lm(rating ~ raises + critical + advance, data)
2 summary(model_all)
3
4 no_critical_model = lm(rating ~ raises + advance, data)
5 summary(no_critical_model)
6
7 no_raises_model = lm(rating ~ critical + advance, data)
8 summary(no_raises_model)
9

```

Листинг 4: Построение линейных зависимостей

Таблица 1: Характеристики модели зависимости параметра Catholic от параметра Agriculture в наборе данных Swiss

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	3.8313	13.8966	0.276	0.7840	
Examination	0.7365	0.2508	2.937	0.0052	**

Таблица 2: Характеристики модели зависимости параметра Catholic от параметра Examination в наборе данных Swiss

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	90.5137	11.6779	7.751	7.96e-10	***
Examination	-2.9940	0.6388	-4.687	2.59e-05	***

Таблица 3: Характеристики модели: (raises ~ critical)

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	35.0246	13.8680	2.526	0.0175	*
critical	0.3960	0.1839	2.153	0.0401	*
Residual standard error:	9.801 on 28 degrees of freedom				
Multiple R-squared:	0.142	Adjusted R-squared:	0.1114		
F-statistic:	4.636 on 1 and 28 DF	p-value:	0.04008		

Таблица 4: Характеристики модели: (raises ~ advance)

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	39.7216	6.8967	5.759	3.5e-06	***
advance	0.5802	0.1564	3.711	0.000907	***
Residual standard error:	8.663 on 28 degrees of freedom				
Multiple R-squared:	0.3297	Adjusted R-squared:	0.3058		
F-statistic:	13.77 on 1 and 28 DF	p-value:	0.0009068		

Таблица 5: Характеристики модели: (critical \sim advance)

Coefficients:					
	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	63.0674	7.6882	8.203	6.28e-09	***
advance	0.2725	0.1743	1.563	0.129	
Residual standard error: 9.657 on 28 degrees of freedom					
Multiple R-squared:	0.08028	Adjusted R-squared:	0.04744		
F-statistic:	2.444 on 1 and 28 DF	p-value:	0.1292		

Таблица 6: Характеристики модели: (rating \sim raises + critical + advance)

Coefficients:					
	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	25.50063	15.60204	1.634	0.114218	
raises	0.89612	0.22555	3.973	0.000501	***
critical	-0.06882	0.20233	-0.340	0.736483	
advance	-0.31773	0.22014	-1.443	0.160870	
Residual standard error: 9.947 on 26 degrees of freedom					
Multiple R-squared:	0.4013	Adjusted R-squared:	0.3322		
F-statistic:	5.809 on 3 and 26 DF	p-value:	0.003537		

Таблица 7: Характеристики модели: (rating \sim raises + advance)

Coefficients:					
	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	21.9917	11.5114	1.910	0.066753	.
raises	0.8752	0.2134	4.101	0.000339	***
advance	-0.3243	0.2157	-1.504	0.144200	
Residual standard error: 9.783 on 27 degrees of freedom					
Multiple R-squared:	0.3986	Adjusted R-squared:	0.3322		
F-statistic:	8.949 on 2 and 27 DF	p-value:	0.001043		

Таблица 8: Характеристики модели: (rating \sim critical + advance)

Coefficients:					
	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	47.2659	18.1737	2.601	0.0149	*
critical	0.1505	0.2422	0.621	0.5396	
advance	0.1425	0.2329	0.612	0.5458	
Residual standard error: 9.783 on 27 degrees of freedom					
Multiple R-squared:	0.3986	Adjusted R-squared:	0.3322		
F-statistic:	8.949 on 2 and 27 DF	p-value:	0.001043		

Таблица 9: Характеристики модели: *model_all_log_1*

Coefficients:					
	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-152.81517	51.46419	-2.969	0.006339	**
I(log(raises))	56.83671	14.02714	4.052	0.000408	***
critical	-0.09279	0.20231	-0.459	0.650289	
advance	-0.27560	0.21253	-1.297	0.206108	
Residual standard error: 9.873 on 26 degrees of freedom					
Multiple R-squared:	0.4102	Adjusted R-squared:	0.3422		
F-statistic:	6.028 on 3 and 26 DF	p-value:	0.002937		

Таблица 10: Характеристики модели: *model_all_log_2*

Coefficients:					
	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	31.5748	56.5610	0.558	0.581453	
raises	0.8864	0.2268	3.909	0.000593	***
I(log(critical))	-2.4274	14.0154	-0.173	0.863842	
advance	-0.3210	0.2205	-1.456	0.157342	
Residual standard error: 9.963 on 26 degrees of freedom					
Multiple R-squared:	0.3993	Adjusted R-squared:	0.33		
F-statistic:	5.762 on 3 and 26 DF	p-value:	0.003684		

Таблица 11: Характеристики модели: *model_all_log_3*

Coefficients:					
	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	63.70537	31.44509	2.026	0.053148	.
raises	0.88450	0.22237	3.978	0.000496	***
I(log(advance))	-13.95696	9.80667	-1.423	0.166565	
critical	-0.05525	0.20363	-0.271	0.788265	
Residual standard error: 9.957 on 26 degrees of freedom					
Multiple R-squared:	0.4001	Adjusted R-squared:	0.3309		
F-statistic:	5.78 on 3 and 26 DF	p-value:	0.003628		

Таблица 12: Характеристики модели: *model_all_log_4*

Coefficients:					
	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	67.5385	60.4114	1.118	0.273801	.
raises	0.8757	0.2235	3.918	0.000579	***
I(log(advance))	-14.1787	9.8125	-1.445	0.160411	
I(log(critical))	-1.5251	14.0925	-0.108	0.914653	
Residual standard error:	9.969 on 26 degrees of freedom				
Multiple R-squared:	0.3986	Adjusted R-squared:	0.3293		
F-statistic:	5.745 on 3 and 26 DF		p-value:	0.003736	

Таблица 13: Характеристики модели: *no_critical_model*

Coefficients:					
	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	61.9131	30.2116	2.049	0.050262	.
raises	0.8692	0.2114	4.112	0.000329	***
I(log(advance))	-14.3182	9.5478	-1.500	0.145311	
Residual standard error:	9.785 on 27 degrees of freedom				
Multiple R-squared:	0.3984	Adjusted R-squared:	0.3538		
F-statistic:	8.939 on 2 and 27 DF		p-value:	0.001049	

Таблица 14: Характеристики модели: *no_critical_model_log_2*

Coefficients:					
	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-115.572	46.471	-2.487	0.019361	*
I(log(raises))	55.004	13.119	4.193	0.000265	***
I(log(advance))	-12.963	9.264	-1.399	0.173132	
Residual standard error:	9.711 on 27 degrees of freedom				
Multiple R-squared:	0.4074	Adjusted R-squared:	0.3635		
F-statistic:	9.281 on 2 and 27 DF		p-value:	0.0008557	

Таблица 15: Характеристики модели: *model_all_mult_1*

Coefficients:					
	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-1.603e+02	5.330e+01	-3.008	0.005633	**
I(log(raises))	5.667e+01	1.382e+01	4.100	0.000339	***
I(critical * advance)	-3.259e-03	2.342e-03	-1.391	0.175475	
Residual standard error:	9.715 on 27 degrees of freedom				
Multiple R-squared:	0.4069	Adjusted R-squared:	0.363		
F-statistic:	9.264 on 2 and 27 DF		p-value:	0.0008645	

Таблица 16: Характеристики модели: *model_all_mult_2*

Coefficients:					
	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-1.571e+02	5.199e+01	-3.022	0.005579	**
I(log(raises))	5.738e+01	1.391e+01	4.125	0.000337	***
I(<i>critical</i> ²)	-8.898e-04	1.417e-03	-0.628	0.535463	
advance	-2.725e-01	2.118e-01	-1.287	0.209550	
Residual standard error: 9.838 on 26 degrees of freedom					
Multiple R-squared:	0.4143	Adjusted R-squared:	0.3468		
F-statistic:	6.131 on 3 and 26 DF	p-value:	0.002693		

Таблица 17: Характеристики модели: *model_all_mult_2*

Coefficients:					
	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-1.571e+02	5.199e+01	-3.022	0.005579	**
I(log(raises))	5.738e+01	1.391e+01	4.125	0.000337	***
I(<i>critical</i> ²)	-8.898e-04	1.417e-03	-0.628	0.535463	
advance	-2.725e-01	2.118e-01	-1.287	0.209550	
Residual standard error: 9.838 on 26 degrees of freedom					
Multiple R-squared:	0.4143	Adjusted R-squared:	0.3468		
F-statistic:	6.131 on 3 and 26 DF	p-value:	0.002693		

Таблица 18: Характеристики модели: *model_all_mult_3*

Coefficients:					
	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-1.633e+02	5.403e+01	-3.023	0.005569	**
I(log(raises))	5.751e+01	1.396e+01	4.120	0.000342	***
I(<i>critical</i> ²)	-9.771e-04	1.412e-03	-0.692	0.495141	
I(<i>advance</i> ²)	-2.819e-03	2.191e-03	-1.286	0.209689	
Residual standard error: 9.838 on 26 degrees of freedom					
Multiple R-squared:	0.4143	Adjusted R-squared:	0.3467		
F-statistic:	6.131 on 3 and 26 DF	p-value:	0.002694		

Таблица 19: Характеристики модели: *model_all_mult_4*

Coefficients:					
	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-1.585e+02	5.348e+01	-2.964	0.006426	**
I(log(raises))	5.690e+01	1.408e+01	4.041	0.000421	***
critical	-1.049e-01	2.018e-01	-0.520	0.607609	
I(<i>advance</i> ²)	-2.831e-03	2.200e-03	-1.287	0.209569	
Residual standard error: 9.877 on 26 degrees of freedom					
Multiple R-squared:	0.4097	Adjusted R-squared:	0.3416		
F-statistic:	6.014 on 3 and 26 DF	p-value:	0.002971		

Таблица 20: Характеристики модели: *no_critical_model_mult_1*

Coefficients:					
	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	51.307766	8.019086	6.398	7.46e-07	***
I(<i>raises</i> ²)	0.006657	0.001686	3.949	0.000507	***
advance	-0.353557	0.224881	-1.572	0.127551	
Residual standard error: 9.923 on 27 degrees of freedom					
Multiple R-squared:	0.3813	Adjusted R-squared:	0.3355		
F-statistic:	8.321 on 2 and 27 DF	p-value:	0.00153		

Таблица 21: Характеристики модели: *no_critical_model_mult_2*

Coefficients:					
	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-1.565e+02	5.262e+01	-2.975	0.006110	**
I(log(raises))	5.457e+01	1.317e+01	4.144	0.000302	***
I(<i>advance</i> ²)	-2.885e-03	2.168e-03	-1.331	0.194465	
Residual standard error: 9.743 on 27 degrees of freedom					
Multiple R-squared:	0.4035	Adjusted R-squared:	0.3594		
F-statistic:	9.133 on 2 and 27 DF	p-value:	0.0009341		

```

1      library("lmtest")
2      library("GGally")
3
4
5      data = na.omit(attitude)  # insertion base data
6
7      help(attitude)
8
9      model\_auxiliary\_1 = lm(raises ~ critical, data)
10     summary(model\_auxiliary\_1)
11
12     model\_auxiliary\_2 = lm(raises ~ advance, attitude)
13     summary(model\_auxiliary\_2)
14
15     model\_auxiliary\_3 = lm(critical ~ advance, data)
16     summary(model\_auxiliary\_3)
17
18
19     model\_all = lm(rating ~ raises + critical + advance, data)
20     summary(model\_all)
21
22
23     no\_critical\_model = lm(rating ~ raises + advance, data)
24     summary(no\_critical\_model)
25
26     no\_raises\_model = lm(rating ~ critical + advance, data)
27     summary(no\_raises\_model)
28
29
30     model\_all\_log\_1 = lm(rating ~ I(log(raises)) + critical +
31     advance, data)
32     summary(model\_all\_log\_1)
33
34     model\_all\_log\_2 = lm(rating ~ raises + I(log(critical)) +
35     advance, data)
36     summary(model\_all\_log\_2)
37
38     model\_all\_log\_3 = lm(rating ~ raises + I(log(advance)) +
39     critical, data)
40     summary(model\_all\_log\_3)
41
42     model\_all\_log\_4 = lm(rating ~ raises + I(log(advance)) + I(
43     log(critical)), data)
44     summary(model\_all\_log\_4)
45
46     no\_critical\_model\_log\_1 = lm(rating ~ raises + I(log(advance
47     )), data)
48     summary(no\_critical\_model\_log\_1)

```

```

45     no_critical_model_log_2 = lm(rating ~ I(log(raises)) + I(log
(advance))), data)
46     summary(no_critical_model_log_2)
47
48     model_all_mult_1 = lm(rating ~ I(log(raises)) + I(critical *
advance), data)
49     summary(model_all_mult_1)
50
51     model_all_mult_2 = lm(rating ~ I(log(raises)) + I(critical
^2) + advance, data)
52     summary(model_all_mult_2)
53
54     model_all_mult_3 = lm(rating ~ I(log(raises)) + I(critical
^2) + I(advance^2), data)
55     summary(model_all_mult_3)
56
57     model_all_mult_4 = lm(rating ~ I(log(raises)) + critical + I
(advance^2), data)
58     summary(model_all_mult_4)
59
60     no_critical_model_mult_1 = lm(rating ~ I(raises^2) + advance
, data)
61     summary(no_critical_model_mult_1)
62
63     no_critical_model_mult_2 = lm(rating ~ I(log(raises)) + I(
advance^2), data)
64     summary(no_critical_model_mult_2)
65
66     data = na.omit(attitude) # insertion database exclude NA
answers
67     # rating- объясняемая переменная
68     # {raises, critical, advance}- регрессоры
69
70     help(attitude)
71
72     model_1 = lm(rating ~ I(log(raises)) + I(log(advance)), data
)
73     summary(model_1)
74
75     t_value = qt(0.975, df = 27)
76     t_value
77
78
79     new.data = data.frame(raises = 34, advance = 51)
80
81     predict(model_1, new.data, interval = "confidence")
82

```

Листинг 5: Практическая работа 2: Полный код программы

```

1      vif(model_def)
2      age          sex      h\_educ city\_status      dur
3      wed1
1.252570      1.138695      1.071094      1.026265      1.080268
2.425066
4      wed2          wed3      satisfy      of      gov
5      2.025865      1.920659      1.035515      1.037693      1.031553
6

```

Листинг 6: Проверка регрессоров на линейную зависимость

Таблица 22: Описание результатов, полученных выше.

Регрессор	Оценка коэффициента β	Std. Error	Доверительный интервал	$\beta = 0$ гипотеза
Свободный коэффициент	-115.572	46.471	[-210.9226 , -20.22136]	Отвергаем
I(log(raises))	55.004	13.119	[28.08603 , 81.92197]	Отвергаем
I(log(advance))	-12.963	9.264	[-31.97116 , 6.045162]	Принимаем

Таблица 23: Доверительный интервал для модели

fit	lwr	upr	Доверительный интервал
1	27.42529	7.856132	46.99445

Таблица 24: Характеристики модели: *model_def*

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-1.03259	0.06699	-15.413	< 2e-16	***
age	-0.06357	0.01320	-4.814	1.51e-06	***
sex	0.49299	0.02534	19.456	< 2e-16	***
h_educ	0.51861	0.02628	19.734	< 2e-16	***
city_status	0.33425	0.02652	12.603	< 2e-16	***
dur	0.13087	0.01226	10.672	< 2e-16	***
wed1	0.03696	0.03733	0.990	0.322194	
wed2	-0.01408	0.04761	-0.296	0.767452	
wed3	-0.11033	0.04697	-2.349	0.018847	*
satisfy	0.23748	0.03463	6.858	7.72e-12	***
of	0.29224	0.03452	8.465	< 2e-16	***
gov	0.17184	0.04992	3.442	0.000581	***
Residual standard error:	0.8972 on 5772 degrees of freedom				
Multiple R-squared:	0.1966	Adjusted R-squared:	0.195		
F-statistic	128.4 on 11 and 5772 DF		p-value:	< 2.2e-16	

Таблица 25: Характеристики модели: *model_2*

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-1.00537	0.06109	-16.456	< 2e-16	***
age	-0.06206	0.01312	-4.732	2.28e-06	***
sex	0.49382	0.02533	19.499	< 2e-16	***
h_educ	0.52052	0.02621	19.860	< 2e-16	***
city_status	0.33311	0.02650	12.572	< 2e-16	***
dur	0.13046	0.01226	10.645	< 2e-16	***
wed2	-0.04508	0.03587	-1.257	0.208843	
wed3	-0.13923	0.03680	-3.783	0.000156	***
satisfy	0.23734	0.03463	6.854	7.94e-12	***
of	0.29334	0.03450	8.501	< 2e-16	***
gov	0.17498	0.04982	3.512	0.000447	***
<hr/>					
Residual standard error:	0.8972 on 5773 degrees of freedom				
Multiple R-squared:	0.1964	Adjusted R-squared:	0.195		
F-statistic	141.1 on 10 and 5773 DF		p-value:	< 2.2e-16	

Таблица 26: Характеристики модели: *model_3*

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-1.01803	0.06026	-16.894	< 2e-16	***
age	-0.06542	0.01284	-5.094	3.61e-07	***
sex	0.50181	0.02452	20.469	< 2e-16	***
h_educ	0.52143	0.02620	19.902	< 2e-16	***
city_status	0.33185	0.02648	12.533	< 2e-16	***
dur	0.12995	0.01225	10.609	< 2e-16	***
wed3	-0.13527	0.03667	-3.689	0.000227	***
satisfy	0.23784	0.03463	6.869	7.16e-12	***
of	0.29268	0.03450	8.483	< 2e-16	***
gov	0.17740	0.04978	3.563	0.000369	***
<hr/>					
Residual standard error:	0.8972 on 5774 degrees of freedom				
Multiple R-squared:	0.1962	Adjusted R-squared:	0.195		
F-statistic	156.6 on 9 and 5774 DF		p-value:	< 2.2e-16	

Таблица 27: Характеристики модели: *model_5*

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-1.38986	0.16679	-8.333	3.22e-16	***
log(age)	-0.19561	0.04326	-4.522	7.01e-06	***
sex	0.56428	0.07208	7.829	1.49e-14	***
h_educ	0.63077	0.08806	7.163	1.73e-12	***
city_status	0.46811	0.07522	6.223	7.68e-10	***
log(dur)	0.14344	0.03867	3.710	0.000221	***
wed3	-0.06911	0.21784	-0.317	0.751120	
satisfy	0.28575	0.09670	2.955	0.003214	**
of	0.38896	0.10493	3.707	0.000224	***
gov	0.17498	0.04982	3.512	0.000447	***

Residual standard error: 1.013 on 836 degrees of freedom

Multiple R-squared: 0.2212 Adjusted R-squared: 0.2128

F-statistic 26.38 on 9 and 836 DF p-value: < 2.2e-16

Таблица 28: Характеристики модели: *model_6*

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
Intercept)	-1.38986	0.16679	-8.333	3.22e-16	***
log(age)	-0.19561	0.04326	-4.522	7.01e-06	***
sex	0.56428	0.07208	7.829	1.49e-14	***
h_educ	0.63077	0.08806	7.163	1.73e-12	***
city_status	0.46811	0.07522	6.223	7.68e-10	***
log(dur)	0.14344	0.03867	3.710	0.000221	***
wed3	-0.06911	0.21784	-0.317	0.751120	
satisfy	0.28575	0.09670	2.955	0.003214	**
of	0.38896	0.10493	3.707	0.000224	***
gov	0.40415	0.12620	3.202	0.001414	**

Residual standard error: 1.013 on 836 degrees of freedom

Multiple R-squared: 0.2212 Adjusted R-squared: 0.2128

F-statistic 26.38 on 9 and 836 DF p-value: < 2.2e-16

Таблица 29: Характеристики модели: *model_4*

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-1.38986	0.16679	-8.333	3.22e-16	***
log(age)	-0.19561	0.04326	-4.522	7.01e-06	***
sex	0.56428	0.07208	7.829	1.49e-14	***
h_educ	0.63077	0.08806	7.163	1.73e-12	***
city_status	0.46811	0.07522	6.223	7.68e-10	***
log(dur)	0.14344	0.03867	3.710	0.000221	***
wed3	-0.06911	0.21784	-0.317	0.751120	***
satisfy	0.28575	0.09670	2.955	0.003214	**
of	0.38896	0.10493	3.707	0.000224	***
gov	0.40415	0.12620	3.202	0.001414	**
<hr/>					
Residual standard error:	1.013 on 836 degrees of freedom				
Multiple R-squared:	0.2212	Adjusted R-squared:	0.2128		
F-statistic	26.38 on 9 and 836 DF		p-value:	< 2.2e-16	

Таблица 30: Характеристики модели: *model_7*

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-0.806083	0.099525	-8.099	9.68e-16	***
I(<i>age</i> ²)	-0.189591	0.024424	-7.763	1.34e-14	***
sex	0.577242	0.045879	12.582	< 2e-16	***
h_educ	0.607754	0.054626	11.126	< 2e-16	***
city_status	0.395695	0.049110	8.057	1.35e-15	***
log(dur)	0.112069	0.024147	4.641	3.70e-06	***
wed3	-0.004729	0.065708	-0.072	0.9426	***
satisfy	0.262643	0.061808	4.249	2.25e-05	***
of	0.324860	0.066965	4.851	1.33e-06	***
gov	0.167293	0.074947	2.232	0.0257	*
<hr/>					
Residual standard error:	0.9857 on 1935 degrees of freedom				
Multiple R-squared:	0.2169	Adjusted R-squared:	0.2132		
F-statistic	59.54 on 9 and 1935 DF		p-value:	< 2.2e-16	

Таблица 31: Характеристики модели: *model_8*

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-0.885618	0.061460	-14.410	< 2e-16	***
I(<i>age</i> ²)	-0.147415	0.011051	-13.339	< 2e-16	***
sex	0.564877	0.023921	23.614	< 2e-16	***
h_educ	0.499908	0.026007	19.222	< 2e-16	***
city_status	0.345314	0.026427	13.066	< 2e-16	***
I(<i>dur</i> ²)	0.009851	0.003183	3.095	0.00198	**
wed3	0.012800	0.034397	0.372	0.70982	***
satisfy	0.242118	0.034544	7.009	2.68e-12	***
of	0.316647	0.034409	9.202	< 2e-16	***
gov	0.120238	0.049649	2.422	0.01548	*
Residual standard error:	0.8944 on 5774 degrees of freedom				
Multiple R-squared:	0.2012	Adjusted R-squared:	0.2		
F-statistic	161.6 on 9 and 5774 DF		p-value:	< 2.2e-16	

Таблица 32: Характеристики модели: *model_9*

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-0.885618	0.061460	-14.410	< 2e-16	***
I(<i>age</i> ²)	-0.147415	0.011051	-13.339	< 2e-16	***
sex	0.564877	0.023921	23.614	< 2e-16	***
h_educ	0.499908	0.026007	19.222	< 2e-16	***
city_status	0.345314	0.026427	13.066	< 2e-16	***
I(<i>dur</i> ²)	0.009851	0.003183	3.095	0.00198	**
wed3	0.012800	0.034397	0.372	0.70982	***
I(<i>satisfy</i> ²)	0.242118	0.034544	7.009	2.68e-12	***
of	0.316647	0.034409	9.202	< 2e-16	***
gov	0.120238	0.049649	2.422	0.01548	*
Residual standard error:	0.8944 on 5774 degrees of freedom				
Multiple R-squared:	0.2012	Adjusted R-squared:	0.2		
F-statistic	161.6 on 9 and 5774 DF		p-value:	< 2.2e-16	

Таблица 33: Характеристики модели: *model_10*

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-0.806083	0.099525	-8.099	9.68e-16	***
I(age^2)	-0.189591	0.024424	-7.763	1.34e-14	***
sex	0.577242	0.045879	12.582	< 2e-16	***
h_educ	0.607754	0.054626	11.126	< 2e-16	***
city_status	0.395695	0.049110	8.057	1.35e-15	***
log(dur)	0.112069	0.024147	4.641	3.70e-06	***
wed3	-0.004729	0.065708	-0.072	0.9426	***
I($satisfy^2$)	0.262643	0.061808	4.249	2.25e-05	***
of	0.324860	0.066965	4.851	1.33e-06	***
gov	0.167293	0.074947	2.232	0.0257	*
Residual standard error: 0.9857 on 1935 degrees of freedom					
Multiple R-squared: 0.2169 Adjusted R-squared: 0.2132					
F-statistic 59.54 on 9 and 1935 DF p-value: < 2.2e-16					

Таблица 34: Характеристики модели: *model_11*

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	0.73571	0.22483	3.272	0.001080	**
I($age^{0.1}$)	-1.92185	0.21873	-8.786	< 2e-16	***
sex	0.42329	0.03345	12.654	< 2e-16	***
h_educ	0.51306	0.03670	13.978	< 2e-16	***
city_status	0.36291	0.03513	10.331	< 2e-16	***
dur	0.12613	0.01635	7.712	1.72e-14	***
I($satisfy^2$)	0.21390	0.04585	4.666	3.23e-06	***
of	0.27076	0.04856	5.575	2.71e-08	***
gov	0.26573	0.07316	3.632	0.000286	***
Residual standard error: 0.8369 on 2711 degrees of freedom					
Multiple R-squared: 0.2109 Adjusted R-squared: 0.2085					
F-statistic 90.54 on 8 and 2711 DF p-value: < 2.2e-16					

Таблица 35: Характеристики модели: *model_12*

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-0.66145	0.09655	-6.851	9.04e-12	***
I($age^{0.5}$)	-0.51647	0.05321	-9.706	< 2e-16	***
sex	0.42565	0.03335	12.762	< 2e-16	***
h_educ	0.51776	0.03660	14.147	< 2e-16	***
city_status	0.36490	0.03503	10.418	< 2e-16	***
dur	0.12248	0.01633	7.501	8.51e-14	***
I($satisfy^2$)	0.21585	0.04571	4.722	2.45e-06	***
of	0.27557	0.04843	5.690	1.40e-08	***
gov	0.25996	0.07293	3.564	0.000371	***
Residual standard error: 0.8344 on 2711 degrees of freedom					
Multiple R-squared: 0.2156 Adjusted R-squared: 0.2133					
F-statistic 93.16 on 8 and 2711 DF p-value: < 2.2e-16					

Таблица 36: Характеристики модели: *model_13*

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-1.39116	0.16665	-8.348	2.87e-16	***
log(age)	-0.19572	0.04323	-4.527	6.85e-06	***
sex	0.56719	0.07145	7.938	6.58e-15	***
h_educ	0.62861	0.08775	7.164	1.72e-12	***
city_status	0.46660	0.07503	6.219	7.88e-10	***
log(dur)	0.14340	0.03864	3.711	0.000220	***
I($satisfy^2$)	0.28644	0.09662	2.965	0.003117	**
of	0.39069	0.10474	3.730	0.000204	***
gov	0.40243	0.12602	3.193	0.001458	**
Residual standard error: 1.013 on 837 degrees of freedom					
Multiple R-squared: 0.2211 Adjusted R-squared: 0.2136					
F-statistic 29.7 on 8 and 837 DF p-value: < 2.2e-16					

Таблица 37: Характеристики модели: *model_14*

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-0.98408	0.09878	-9.962	< 2e-16	***
age	-0.05213	0.02369	-2.200	0.0279	*
sex	0.56649	0.04662	12.150	< 2e-16	***
h_educ	0.62089	0.05536	11.215	< 2e-16	***
city_status	0.38111	0.04988	7.641	3.36e-14	***
log(dur)	0.12104	0.02449	4.943	8.35e-07	***
I(<i>satisfy</i> ²)	0.25514	0.06274	4.067	4.96e-05	***
of	0.29991	0.06786	4.419	1.05e-05	***
gov	0.19042	0.07618	2.500	0.0125	*
<hr/>					
Residual standard error:	1.001 on 1936 degrees of freedom				
Multiple R-squared:	0.1925	Adjusted R-squared:	0.1891		
F-statistic	57.68 on 8 and 1936 DF		p-value:	< 2.2e-16	

Таблица 38: Характеристики модели: *model_15*

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-1.236089	0.088460	-13.973	< 2e-16	***
log(age)	-0.192734	0.020448	-9.426	< 2e-16	***
sex	0.473786	0.033086	14.320	< 2e-16	***
h_educ	0.489537	0.036965	13.243	< 2e-16	***
city_status	0.354453	0.035506	9.983	< 2e-16	***
I(<i>dur</i> ²)	0.009629	0.003676	2.620	0.008855	**
I(<i>satisfy</i> ²)	0.227148	0.046341	4.902	1.01e-06	***
of	0.276562	0.049080	5.635	1.93e-08	***
gov	0.270378	0.074116	3.648	0.000269	***
<hr/>					
Residual standard error:	0.8459 on 2711 degrees of freedom				
Multiple R-squared:	0.1939	Adjusted R-squared:	0.1915		
F-statistic	81.5 on 8 and 2711 DF		p-value:	< 2.2e-16	

Таблица 39: Характеристики модели: *model_16*

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-2.29310	0.26668	-8.599	< 2e-16	***
age	-0.05173	0.02368	-2.185	0.0290	*
sex	0.56489	0.04660	12.122	< 2e-16	***
h_educ	0.62154	0.05533	11.232	< 2e-16	***
city_status	0.38165	0.04985	7.656	3.00e-14	***
$I(dur^{0.1})$	1.30480	0.25384	5.140	3.02e-07	***
satisfy	0.25617	0.06271	4.085	4.59e-05	***
of	0.29954	0.06783	4.416	1.06e-05	***
gov	0.19191	0.07614	2.520	0.0118	*
Residual standard error: 1 on 1936 degrees of freedom					
Multiple R-squared: 0.1933 Adjusted R-squared: 0.19					
F-statistic 57.98 on 8 and 1936 DF p-value: < 2.2e-16					

Таблица 40: Характеристики модели: *model_17*

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-1.10310	0.11188	-9.860	< 2e-16	***
$I(age^2)$	-0.18832	0.02350	-8.013	1.91e-15	***
sex	0.57424	0.04576	12.550	< 2e-16	***
h_educ	0.60937	0.05446	11.190	< 2e-16	***
city_status	0.39628	0.04901	8.086	1.08e-15	***
$I(dur^{0.5})$	0.26856	0.05170	5.195	2.26e-07	***
satisfy	0.26638	0.06172	4.316	1.67e-05	***
of	0.32401	0.06681	4.849	1.34e-06	***
gov	0.16972	0.07480	2.269	0.0234	*
Residual standard error: 0.9841 on 1936 degrees of freedom					
Multiple R-squared: 0.219 Adjusted R-squared: 0.2158					
F-statistic 67.87 on 8 and 1936 DF p-value: < 2.2e-16					

```

1
2 library("lmtest")
3 library("rlms")
4 library("dplyr")
5 library("GGally")
6 library("car")
7 library("sandwich")
8
9 #####
10 ds_0 <- rlms_read("r20i_os26c.sav")
11
12 ds = select(ds_0, pj13.2, p_age, ph5, p_marst, p_diplom, status,
13 pj6.2, pj1.1.1, pj11.1, pj23)
14
15 # исключение строк со значением NA
16 ds = na.omit(ds)
17
18 #####
19 # Семейное положение
20 ds["wed"] = ds$p_marst
21 ds["wed"] = lapply(ds["wed"], as.character)
22 ds$wed1 = 0
23
24 ds$wed1[which(ds$wed == '2')] <- 1
25 ds$wed1[which(ds$wed == '6')] <- 1
26 ds$wed1 = as.numeric(ds$wed1)
27
28 ds["wed2"] = ds$p_marst
29 ds$wed2 = 0
30
31 ds$wed2[which(ds$wed == '4')] <- 1
32 ds$wed2[which(ds$wed == '5')] <- 1
33 ds$wed2 = as.numeric(ds$wed2)
34
35 ds["wed3"] = ds$p_marst
36 ds$wed3 = 0
37
38 ds$wed3[which(ds$wed == '1')] <- 1
39 # ds$wed3[which(ds$wed == '3')] <- 1
40 ds$wed3 = as.numeric(ds$wed3)
41
42 #####
43 # Пол
44 ds["sex"] = ds$ph5
45 ds["sex"] = lapply(ds["sex"], as.character)
46
47 ds$sex[which(ds$sex == '1')] <- 1
48 ds$sex[which(ds$sex == '2')] <- 0

```

```

49 ds$sex = as.numeric(ds$sex)
50
51 #####
52 # Населенный пункт
53 ds["status1"] = ds$status
54 ds["status1"] = lapply(ds["status1"], as.character)
55 ds["city_status"] = 0
56 ds$city_status[which(ds$status1 == '1')] <- 1
57 ds$city_status[which(ds$status1 == '2')] <- 1
58 ds$city_status = as.numeric(ds$city_status)
59
60 #####
61 # Наличие высшего образования
62 ds["higher_educ"] = ds$p_diplom
63 ds["higher_educ"] = lapply(ds["higher_educ"], as.character)
64 ds["h_educ"] = ds$p_diplom
65 ds["h_educ"] = 0
66 ds$h_educ[which(ds$higher_educ == '6')] <- 1
67
68 # Образование
69 # ds["higher_educ"] = ds$s_educ
70 # ds["higher_educ"] = lapply(ds["higher_educ"], as.character)
71 # ds["h_educ"] = ds$s_educ
72 # ds["h_educ"] = 0
73 # ds$h_educ[which(ds$higher_educ=='21')] <- 1
74 # ds$h_educ[which(ds$higher_educ=='22')] <- 1
75 # ds$h_educ[which(ds$higher_educ=='23')] <- 1
76
77 #####
78 # Зарплата: преобразование в вещественную и нормализация
79 sal1 = as.character(ds$pj13.2)
80 sal2 = lapply(sal1, as.integer)
81 sal = as.numeric(unlist(sal2))
82
83 mean(sal)
84
85 ds["salary"] = (sal - mean(sal)) / sqrt(var(sal))
86 ds["salary"]
87
88 #####
89 # Продолжительность рабочей недели: преобразование в вещественну
ю и нормализация
90 dur1 = as.character(ds$pj6.2)
91 dur2 = lapply(dur1, as.integer)
92 dur3 = as.numeric(unlist(dur2))
93
94 mean(dur3)
95
96 ds["dur"] = (dur3 - mean(dur3)) / sqrt(var(dur3))

```



```

97   ds["dur"]
98
99   #####
100  # Возраст: преобразование в вещественную и нормализация
101  age1 = as.character(ds$p_age)
102  age2 = lapply(age1, as.integer)
103  age3 = as.numeric(unlist(age2))
104
105  mean(age3)
106
107  ds["age"] = (age3 - mean(age3)) / sqrt(var(age3))
108  ds["age"]
109
110  #####
111  # Удовлетворенность работой в целом
112  ds["sat"] = ds$pj1.1.1
113  ds["sat"] = lapply(ds["sat"], as.character)
114  ds["satisfy"] = 0
115  ds$satisfy[which(ds$sat == '1')] <- 1
116  ds$satisfy[which(ds$sat == '2')] <- 1
117  ds$satisfy[which(ds$sat == '3')] <- 1
118  ds$satisfy = as.numeric(ds$satisfy)
119
120  #####
121  # Официальное трудоустройство
122  ds["of_w"] = ds$pj1.1.1
123  ds["of_w"] = lapply(ds["of_w"], as.character)
124  ds["of"] = 0
125  ds$of[which(ds$of_w == '1')] <- 1
126  ds$of = as.numeric(ds$of)
127
128  #####
129  # Является государство владельцем или совладельцем Вашего предпр-
иятия, организации?
130  ds["gov_1"] = ds$pj11.1
131  ds["gov_1"] = lapply(ds["gov_1"], as.character)
132  ds["gov"] = 0
133  ds$gov[which(ds$gov_1 == '1')] <- 1
134  ds$gov = as.numeric(ds$gov)
135
136  #####
137  ds = na.omit(ds)
138
139  #####
140  # Построение линейной регрессии зависимости переменной 'salary'
от
141  # всех введенных регрессоров
142
143  ds_2 = select(ds, salary, age, sex, h_educ, city_status, dur,

```

```

144     wed1, wed2, wed3, satisfy, of, gov)
145
146     # модель_1 - 'model_def'
147     model_def = lm(data = ds_2, salary ~ age + sex + h_educ + city_
148     status +
149     dur + wed1 + wed2 + wed3 + satisfy + of + gov)
150     summary(model_def)
151     vif(model_def)
152
153     # модель_2 - 'model_2'
154     model_2 = lm(data = ds_2, salary ~ age + sex + h_educ + city_
155     status +
156     dur + wed2 + wed3 + satisfy + of + gov)
157     summary(model_2)
158     vif(model_2)
159
160     # модель_3 - 'model_3'
161     model_3 = lm(data = ds_2, salary ~ age + sex + h_educ + city_
162     status +
163     dur + wed3 + satisfy + of + gov)
164     summary(model_3)
165     vif(model_3)
166
167     # модель_4 - 'model_4'
168     model_4 = lm(data = ds_2, salary ~ log(age) + sex + h_educ +
169     city_status +
170     dur + wed3 + satisfy + of + gov)
171     summary(model_4)
172     vif(model_4)
173
174     # модель_5 - 'model_5'
175     model_5 = lm(data = ds_2, salary ~ log(age) + sex + h_educ +
176     city_status +
177     log(dur) + wed3 + satisfy + of + gov)
178     summary(model_5)
179     vif(model_5)
180
181     # модель_6 - 'model_6'
182     model_6 = lm(data = ds_2, salary ~ log(age) + sex + h_educ +
183     city_status +
184     log(dur) + wed3 + satisfy + of + gov)
185     summary(model_6)
186     vif(model_6)
187
188     # модель_7 - 'model_7'
189     model_7 = lm(data = ds_2, salary ~ I(age^2) + sex + h_educ +
190     city_status +
191     log(dur) + wed3 + satisfy + of + gov)
192     summary(model_7)
193     vif(model_7)

```

```

185     vif(model_7)
186
187     # модель_8 - 'model_8'
188     model_8 = lm(data = ds_2, salary ~ I(age^2) + sex + h_educ +
189                 city_status +
190                 I(dur^2) + wed3 + satisfy + of + gov)
191     summary(model_8)
192     vif(model_8)
193
194     # модель_9 - 'model_9'
195     model_9 = lm(data = ds_2, salary ~ I(age^2) + sex + h_educ +
196                 city_status +
197                 I(dur^2) + wed3 + I(satisfy^2) + of + gov)
198     summary(model_9)
199     vif(model_9)
200
201     # модель_10 - 'model_10'
202     model_10 = lm(data = ds_2, salary ~ I(age^2) + sex + h_educ +
203                 city_status +
204                 log(dur) + wed3 + I(satisfy^2) + of + gov)
205     summary(model_10)
206     vif(model_10)
207
208     # модель_11 - 'model_11'
209     model_11 = lm(data = ds_2, salary ~ I(age^2) + sex + h_educ +
210                 city_status +
211                 log(dur) + wed3 + I(satisfy^2) + of + gov)
212     summary(model_11)
213     vif(model_11)
214
215     # модель_12 - 'model_12'
216     model_12 = lm(data = ds_2, salary ~ I(age^0.1) + sex + h_educ +
217                 city_status +
218                 dur + I(satisfy^2) + of + gov)
219     summary(model_12)
220     vif(model_12)
221
222     # модель_13 - 'model_13'
223     model_13 = lm(data = ds_2, salary ~ I(age^0.5) + sex + h_educ +
224                 city_status +
225                 dur + I(satisfy^2) + of + gov)
226     summary(model_13)

```

```

227     vif(model_14)
228
229     # модель_15 - 'model_15'
230     model_15 = lm(data = ds_2, salary ~ log(age) + sex + h_educ +
231                   city_status +
232                   log(dur) + I(satisfy^0.5) + of + gov)
233     summary(model_15)
234     vif(model_15)
235
236     # модель_16 - 'model_16'
237     model_16 = lm(data = ds_2, salary ~ log(age) + sex + h_educ +
238                   city_status +
239                   log(dur) + I(satisfy^1.5) + of + gov)
240     summary(model_16)
241     vif(model_16)
242
243     # модель_17 - 'model_17'
244     model_17 = lm(data = ds_2, salary ~ age + sex + h_educ + city_
245                   status +
246                   log(dur) + I(satisfy^2) + of + gov)
247     summary(model_17)
248     vif(model_17)
249
250     # модель_18 - 'model_18'
251     model_18 = lm(data = ds_2, salary ~ log(age) + sex + h_educ +
252                   city_status +
253                   I(dur^2) + I(satisfy^2) + of + gov)
254     summary(model_18)
255     vif(model_18)
256
257     # модель_19 - 'model_19'
258     model_19 = lm(data = ds_2, salary ~ age + sex + h_educ + city_
259                   status +
260                   I(dur^0.1) + satisfy + of + gov)
261     summary(model_19)
262     vif(model_19)
263
264     # модель_20 - 'model_20'
265     model_20 = lm(data = ds_2, salary ~ I(age^2) + sex + h_educ +
266                   city_status +
267                   I(dur^0.5) + satisfy + of + gov)
268     summary(model_20)
269     vif(model_20)
270
271     # модель_21 - 'model_21'
272     model_21 = lm(data = ds_2, salary ~ I(age^2) + sex + h_educ +
273                   city_status +
274                   I(dur^2) + satisfy + of + gov)
275     summary(model_21)

```

```

269     vif(model_21)
270
271     # модель_22 - 'model_22'
272     model_22 = lm(data = ds_2, salary ~ age + sex + h_educ + city_
273                  status +
274                  log(dur) + satisfy + of + gov)
275     summary(model_22)
276     vif(model_22)
277
278     #####
279
280     # Не вступавшие в брак мужчины, без высшего образования;
281
282     data5_1 = subset(ds_2, sex = 1)
283     data5_2 = subset(data5_1, wed3 = 1)
284     data5_fin = subset(data5_2, h_educ = 0)
285
286     model_5_1 = lm(data = data5_fin, salary ~ log(age) + city_status
287                  +
288                  log(dur) + I(satisfy^2) + of + gov)
289     summary(model_5_1)
290
291     #####
292
293     # Городские жители, мужчины состоящие в браке
294
295     data5_1 = subset(ds_2, sex = 1)
296     data5_2 = subset(data5_1, city_status = 1)
297     data5_fin = subset(data5_2, wed1 = 1)
298
299     model_5_2 = lm(data = data5_fin, salary ~ log(age) + h_educ +
300                  log(dur) + I(satisfy^2) + of + gov)
301     summary(model_5_2)
302
303

```

Листинг 7: Практическая работа №3: Полный код программы

```

1 import pandas as pd
2 import numpy as np
3 import seaborn as sns
4 import matplotlib.pyplot as plt
5 from sklearn.decomposition import PCA
6 from sklearn.linear_model import LogisticRegression
7 from sklearn.model_selection import train_test_split, GridSearchCV,
   RandomizedSearchCV, learning_curve, ShuffleSplit, cross_val_score
8 from sklearn.metrics import confusion_matrix

```

Таблица 41: Характеристики модели: *model_18*

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-0.98408	0.09878	-9.962	< 2e-16	***
age	-0.05213	0.02369	-2.200	0.0279	*
sex	0.56649	0.04662	12.150	< 2e-16	***
h_educ	0.62089	0.05536	11.215	< 2e-16	***
city_status	0.38111	0.04988	7.641	3.36e-14	***
log(dur)	0.12104	0.02449	4.943	8.35e-07	***
satisfy	0.25514	0.06274	4.067	4.96e-05	***
of	0.29991	0.06786	4.419	1.05e-05	***
gov	0.19042	0.07618	2.500	0.0125	*
<hr/>					
Residual standard error:	1.001 on 1936 degrees of freedom				
Multiple R-squared:	0.1925	Adjusted R-squared:	0.1891		
F-statistic	57.68 on 8 and 1936 DF		p-value:	< 2.2e-16	

Таблица 42: Характеристики модели: *data5_fin*

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-0.61121	0.15321	-3.989	7.20e-05	***
log(age)	-0.18727	0.04557	-4.110	4.35e-05	***
h_educ	0.71432	0.09195	7.769	2.31e-14	***
log(dur)	0.16612	0.03971	4.183	3.18e-05	***
I(<i>satisfy</i> ²)	0.25840	0.10182	2.538	0.011333	*
of	0.40250	0.11037	3.647	0.000282	***
gov	0.23492	0.13147	1.787	0.074321	.
<hr/>					
Residual standard error:	1.068 on 839 degrees of freedom				
Multiple R-squared:	0.1319	Adjusted R-squared:	0.1257		
F-statistic	21.25 on 6 and 839 DF		p-value:	< 2.2e-16	

```

9 from sklearn.metrics import f1_score, roc_auc_score, roc_curve
10 from sklearn.preprocessing import StandardScaler, LabelEncoder
11 from sklearn.pipeline import make_pipeline
12 # from pandas_profiling import ProfileReport
13 from sklearn.impute import KNNImputer
14
15 from sklearn.ensemble import RandomForestClassifier
16
17 import warnings
18 warnings.filterwarnings("ignore")
19
20 from sklearn import svm
21
22 from scipy.special import boxcox1p
23 from scipy.stats import boxcox_normmax
24 pd.set_option('display.max_columns', 50)
25 pd.set_option('display.max_rows', 100)
26
27 bank_df = pd.read_csv("C:\\Users\\averu\\Documents\\git_local\\
    programming-practice\\IDA-practice-4\\BankChurners.csv")
28 bank_df
29
30 bank_df['Marital_Status'] = np.where(bank_df['Marital_Status'] == '
    Married' , False, bank_df['Marital_Status'])
31 bank_df['Marital_Status'] = np.where(bank_df['Marital_Status'] == '
    Single' , True, bank_df['Marital_Status'])
32 bank_df['Marital_Status'] = np.where(bank_df['Marital_Status'] == '
    Divorced' , True, bank_df['Marital_Status'])
33 bank_df['Marital_Status'] = np.where(bank_df['Marital_Status'] == '
    Unknown' , True, bank_df['Marital_Status'])
34
35 # Выбрасываем некоторые признаки
36 drop_cols = list(bank_df.iloc[:, [0, -1, -2, -3, 6]].columns)
37 bank_df.drop(columns=drop_cols, inplace=True)
38
39 # Трансформация категориального признака к целочисленному типу 'int'
40 bank_df['Attrition_Flag'] = (bank_df['Attrition_Flag'] == 'Attrited
    Customer').astype(int)
41
42 bank_df.describe()
43
44 # Contacts_Count_12_mon
45 by_contact_df = bank_df.groupby('Contacts_Count_12_mon')['
    Attrition_Flag'].mean()
46 by_contact_df.plot(kind='bar', ylabel='Attrited Ratio')
47
48 # Months_Inactive_12_mon
49 by_inactive_df = bank_df.groupby('Months_Inactive_12_mon')['
    Attrition_Flag'].mean()

```

```

50 by_inactive_df.plot(kind='bar', ylabel='Attrited Ratio')
51
52 # Education level
53 by_edu_df = bank_df.groupby('Education_Level')['Attrition_Flag'].
    mean()
54 by_edu_df.plot(kind='bar', ylabel='Attrited Ratio')
55
56 # Income_Category
57 by_income_df = bank_df.groupby('Income_Category')['Attrition_Flag'].
    mean()
58 by_income_df.plot(kind='bar', ylabel='Attrited Ratio')
59
60 # Gender
61 by_gender_df = bank_df.groupby('Gender')['Attrition_Flag'].mean()
62 by_gender_df.plot(kind='bar', ylabel='Attrited Ratio')
63
64 # Card_Category
65 by_card_df = bank_df.groupby('Card_Category')['Attrition_Flag'].mean
    ()
66 by_card_df.plot(kind='bar', ylabel='Attrited Ratio')
67
68 # Credit Limit
69 print("Avg attrition of customers with minimum credit limit: ",
70       bank_df[bank_df['Credit_Limit']==bank_df['Credit_Limit'].min()
71             ]['Attrition_Flag'].mean())
72 print("Avg attrition of customers with more credit limit: ",
73       bank_df[bank_df['Credit_Limit']!=bank_df['Credit_Limit'].min()
74             ]['Attrition_Flag'].mean())
75
76 # Total_Revolving_Bal
77 print("Avg attrition of customers with 0 revolving balance: ",
78       bank_df[bank_df['Total_Revolving_Bal']==0]['Attrition_Flag'].
    mean())
79 print("Avg attrition of customers with more revolving balance: ",
80       bank_df[bank_df['Total_Revolving_Bal']!=0]['Attrition_Flag'].
    mean())
81
82 tmp_bank_df = bank_df.copy()
83
84 le_ls = []
85 for col in ['Education_Level', 'Income_Category']:
86     le = LabelEncoder()
87     tmp_bank_df[col] = le.fit_transform(tmp_bank_df[col])
88     keys = le.classes_
89     values = le.transform(le.classes_)
90     dictionary = dict(zip(keys, values))
91     le_ls.append(le)
92     print(dictionary)

```



```

92     tmp_bank_df.loc[tmp_bank_df[col]==dictionary['Unknown'], col] =
    np.nan
93
94 tmp_bank_df = pd.get_dummies(tmp_bank_df)
95
96 for col in ['Income_Category', 'Education_Level']:
97     imputer = KNNImputer(n_neighbors = 5)
98     fill_tmp_bank_df = pd.DataFrame(imputer.fit_transform(
    tmp_bank_df.iloc[:,1:]),
99                                     index=tmp_bank_df.index, columns
    =tmp_bank_df.columns[1:])
100     tmp_bank_df[col] = fill_tmp_bank_df[col]
101
102 i = 0
103 for col in ['Education_Level', 'Income_Category']:
104     tmp_bank_df[col] = le_ls[i].inverse_transform(round(tmp_bank_df[
    col], 0).astype(int))
105     i += 1
106
107 bank_df = tmp_bank_df
108 bank_df
109
110 # One-hot encoding
111 bank_df = pd.get_dummies(bank_df)
112
113
114 bank_df['Avg_Trans_Amt'] = bank_df['Total_Trans_Amt']/bank_df['
    Total_Trans_Ct']
115
116 bank_df.loc[bank_df['Credit_Limit']==bank_df['Credit_Limit'].min(),
    'Min_Credit_Limit'] = 1
117 bank_df['Min_Credit_Limit'].fillna(0, inplace=True)
118
119 bank_df.loc[bank_df['Total_Revolving_Bal']==0, '0
    _Total_Revolving_Bal'] = 1
120 bank_df['0_Total_Revolving_Bal'].fillna(0, inplace=True)
121
122 bank_df.loc[bank_df['Total_Amt_Chng_Q4_Q1']<bank_df['
    Total_Amt_Chng_Q4_Q1'].median(), 'Amt_Q4_Q1_Dec'] = 1
123 bank_df['Amt_Q4_Q1_Dec'].fillna(0, inplace=True)
124
125 bank_df.loc[bank_df['Total_Ct_Chng_Q4_Q1']<bank_df['
    Total_Ct_Chng_Q4_Q1'].median(), 'Ct_Q4_Q1_Dec'] = 1
126 bank_df['Ct_Q4_Q1_Dec'].fillna(0, inplace=True)
127
128
129
130 # Трансформация введенных новых признаков и признаков из таблицы
131 skewed_col = ['Total_Trans_Amt', 'Credit_Limit', 'Avg_Open_To_Buy']

```

```

132 trans_bank_df = bank_df.copy()
133 for col in skewed_col:
134     plt.figure(figsize=(8, 4))
135     plt.subplot(1, 2, 1)
136     plt.hist(bank_df[col])
137     plt.title(f"{col}: Before")
138     trans_bank_df[col] = boxcox1p(bank_df[col], boxcox_normmax(
139         bank_df[col] + 1))
140     plt.subplot(1, 2, 2)
141     plt.hist(trans_bank_df[col])
142     plt.title(f"{col}: After")
143
144 # Стандартизация признаков
145 std = StandardScaler()
146 std_df = pd.DataFrame(std.fit_transform(trans_bank_df.iloc[:,1:]),
147                        index = trans_bank_df.index,
148                        columns = trans_bank_df.columns[1:])
149 std_bank_df = pd.concat([trans_bank_df.iloc[:,0], std_df], axis=1)
150
151 # Reference: https://scikit-learn.org/stable/auto\_examples/
152 # model\_selection/plot\_learning\_curve.html
153 def plot_learning_curve(
154     estimator,
155     title,
156     X,
157     y,
158     axes=None,
159     ylim=None,
160     cv=None,
161     n_jobs=None,
162     train_sizes=np.linspace(0.1, 1.0, 5),
163 ):
164     """
165     Generate 3 plots: the test and training learning curve, the
166     training
167     samples vs fit times curve, the fit times vs score curve.
168
169     Parameters
170     -----
171     estimator : estimator instance
172         An estimator instance implementing 'fit' and 'predict'
173         methods which
174         will be cloned for each validation.
175
176     title : str
177         Title for the chart.
178
179     X : array-like of shape (n_samples, n_features)
180         Training vector, where 'n_samples' is the number of

```

```

177     samples and
178         ‘‘n_features’’ is the number of features.
179
180     y : array-like of shape (n_samples) or (n_samples, n_features)
181         Target relative to ‘‘X’’ for classification or regression;
182         None for unsupervised learning.
183
184     axes : array-like of shape (3,), default=None
185         Axes to use for plotting the curves.
186
187     ylim : tuple of shape (2,), default=None
188         Defines minimum and maximum y-values plotted, e.g. (ymin,
189         ymax).
190
191     cv : int, cross-validation generator or an iterable, default=
192         None
193         Determines the cross-validation splitting strategy.
194         Possible inputs for cv are:
195
196         - None, to use the default 5-fold cross-validation,
197         - integer, to specify the number of folds.
198         - :term:‘CV splitter’,
199         - An iterable yielding (train, test) splits as arrays of
200         indices.
201
202         For integer/None inputs, if ‘‘y’’ is binary or multiclass,
203         :class:‘StratifiedKFold’ used. If the estimator is not a
204         classifier
205         or if ‘‘y’’ is neither binary nor multiclass, :class:‘KFold’
206         is used.
207
208         Refer :ref:‘User Guide <cross_validation>’ for the various
209         cross-validators that can be used here.
210
211     n_jobs : int or None, default=None
212         Number of jobs to run in parallel.
213         ‘‘None’’ means 1 unless in a :obj:‘joblib.parallel_backend’
214         context.
215         ‘‘-1’’ means using all processors. See :term:‘Glossary <
216         n_jobs>’
217         for more details.
218
219     train_sizes : array-like of shape (n_ticks,)
220         Relative or absolute numbers of training examples that will
221         be used to
222         generate the learning curve. If the ‘‘dtype’’ is float, it
223         is regarded
224         as a fraction of the maximum size of the training set (that
225         is

```

```

215         determined by the selected validation method), i.e. it has
to be within
216         (0, 1]. Otherwise it is interpreted as absolute sizes of the
training
217         sets. Note that for classification the number of samples
usually have
218         to be big enough to contain at least one sample from each
class.
219         (default: np.linspace(0.1, 1.0, 5))
220     """
221     if axes is None:
222         _, axes = plt.subplots(1, 3, figsize=(20, 5))
223
224     axes[0].set_title(title)
225     if ylim is not None:
226         axes[0].set_ylim(*ylim)
227     axes[0].set_xlabel("Training examples")
228     axes[0].set_ylabel("Score")
229
230     train_sizes, train_scores, test_scores, fit_times, _ =
learning_curve(
231         estimator,
232         X,
233         y,
234         cv=cv,
235         n_jobs=n_jobs,
236         train_sizes=train_sizes,
237         return_times=True,
238     )
239     train_scores_mean = np.mean(train_scores, axis=1)
240     train_scores_std = np.std(train_scores, axis=1)
241     test_scores_mean = np.mean(test_scores, axis=1)
242     test_scores_std = np.std(test_scores, axis=1)
243     fit_times_mean = np.mean(fit_times, axis=1)
244     fit_times_std = np.std(fit_times, axis=1)
245
246     # Plot learning curve
247     axes[0].grid()
248     axes[0].fill_between(
249         train_sizes,
250         train_scores_mean - train_scores_std,
251         train_scores_mean + train_scores_std,
252         alpha=0.1,
253         color="r",
254     )
255     axes[0].fill_between(
256         train_sizes,
257         test_scores_mean - test_scores_std,
258         test_scores_mean + test_scores_std,

```

```

259         alpha=0.1,
260         color="g",
261     )
262     axes[0].plot(
263         train_sizes, train_scores_mean, "o-", color="r", label="
Training score"
264     )
265     axes[0].plot(
266         train_sizes, test_scores_mean, "o-", color="g", label="Cross
-validation score"
267     )
268     axes[0].legend(loc="best")
269
270     # Plot n_samples vs fit_times
271     axes[1].grid()
272     axes[1].plot(train_sizes, fit_times_mean, "o-")
273     axes[1].fill_between(
274         train_sizes,
275         fit_times_mean - fit_times_std,
276         fit_times_mean + fit_times_std,
277         alpha=0.1,
278     )
279     axes[1].set_xlabel("Training examples")
280     axes[1].set_ylabel("fit_times")
281     axes[1].set_title("Scalability of the model")
282
283     # Plot fit_time vs score
284     axes[2].grid()
285     axes[2].plot(fit_times_mean, test_scores_mean, "o-")
286     axes[2].fill_between(
287         fit_times_mean,
288         test_scores_mean - test_scores_std,
289         test_scores_mean + test_scores_std,
290         alpha=0.1,
291     )
292     axes[2].set_xlabel("fit_times")
293     axes[2].set_ylabel("Score")
294     axes[2].set_title("Performance of the model")
295
296     return plt
297
298 # SVM
299 X_train, X_test, y_train, y_test = train_test_split(std_bank_df.iloc
[:,1:], std_bank_df.iloc[:,0], test_size=0.25, random_state=42)
300 svm_clf = svm.SVC()
301
302 # Поиск наилучших параметров для классификации методом опорных вектор
ов
303 svm_param = {'C': [0.1, 1, 10, 100],

```

```

304         'gamma': ['scale', 'auto'],
305         'kernel': ['rbf', 'poly', 'sigmoid'],
306         'probability': [True],
307         'class_weight': ['balanced', 'none']}
308 svm_grid = GridSearchCV(svm_clf, svm_param, n_jobs=-1, cv=5, verbose
    =1)
309 svm_grid.fit(X_train, y_train)
310 print("Best param: ", svm_grid.best_params_)
311 svm_pred = svm_grid.predict(X_test)
312 svm_pred_prob = svm_grid.predict_proba(X_test)[:,-1]
313 svm_pred_df = pd.DataFrame({"pred":svm_pred, "prob":svm_pred_prob, "
    actual":y_test})
314 # manually adjust pred based on prob
315 thresh = np.quantile(svm_pred_prob, (1-y_train.mean()))
316 svm_pred_df['pred'] = (svm_pred_df['prob']>thresh).astype(int)
317 svm_pred_df
318
319 X, y = std_bank_df.iloc[:,1:], std_bank_df.iloc[:,0]
320
321
322 estimator = svm.SVC(C=10,
323                     class_weight='balanced',
324                     gamma='scale',
325                     kernel='rbf',
326                     probability=True)
327 estimator.fit(X_train, y_train)
328
329 plt.show()
330
331 print("accuracy:"+str(np.average(cross_val_score(estimator, X_test,
    y_test, scoring= 'accuracy'))))
332 print("f1:"+str(np.average(cross_val_score(estimator, X_test,
    y_test, scoring= 'f1'))))
333 print("precision:"+str(np.average(cross_val_score(estimator, X_test,
    y_test, scoring= 'precision'))))
334 print("recall:"+str(np.average(cross_val_score(estimator, X_test,
    y_test, scoring= 'recall'))))
335
336
337 # Random Forest
338
339 rf = RandomForestClassifier()
340 n_estimators = [100, 200, 300]
341 max_features = [0.5, 0.25, 'log2', 'sqrt']
342 max_depth = [100, 200, 'none']
343 min_samples_split = [2, 5, 10]
344 min_samples_leaf = [1, 2, 4]
345 bootstrap = [True, False]
346 class_weight = ['balanced', 'none']

```

```

347 rf_param = {'n_estimators': n_estimators,
348             'max_features': max_features,
349             'max_depth': max_depth,
350             'min_samples_split': min_samples_split,
351             'min_samples_leaf': min_samples_leaf,
352             'bootstrap': bootstrap,
353             'class_weight': class_weight}
354 rf_grid = GridSearchCV(rf, rf_param, n_jobs=-1, cv=5, verbose=1)
355 rf_grid.fit(X_train, y_train)
356
357
358
359 estimator = RandomForestClassifier(bootstrap=False,
360                                   class_weight='balanced',
361                                   max_depth=200,
362                                   max_features=0.25,
363                                   min_samples_leaf=1,
364                                   min_samples_split=2,
365                                   n_estimators=300)
366 estimator.fit(X_train, y_train)
367
368 plt.show()
369
370 print("accuracy:" + str(np.average(cross_val_score(estimator, X_test,
371                                                    y_test, scoring= 'accuracy'))))
371 print("f1:" + str(np.average(cross_val_score(estimator, X_test,
372                                                    y_test, scoring= 'f1'))))
372 print("precision:" + str(np.average(cross_val_score(estimator, X_test,
373                                                    y_test, scoring= 'precision'))))
373 print("recall:" + str(np.average(cross_val_score(estimator, X_test,
374                                                    y_test, scoring= 'recall'))))

```

Листинг 8: Полный код решения Практической работы №4

```

1 import numpy as np # библиотека для эффективной работы с данными
2 import pandas as pd # библиотека для работы с наборами данных
3 import matplotlib.pyplot as plt # библиотека для визуализации
4 import seaborn as sns # еще одна библиотека для построения графиков
5
6 data = pd.read_csv('C:\\Users\\averu\\Documents\\git_local\\
7                   programming-practice\\IDA-practice-5\\insurance.csv')
8
9 data.shape
10 data.head(7)
11 data.info()
12 data['sex'].value_counts()
13 data['smoker'].value_counts()
14 data['region'].value_counts()

```

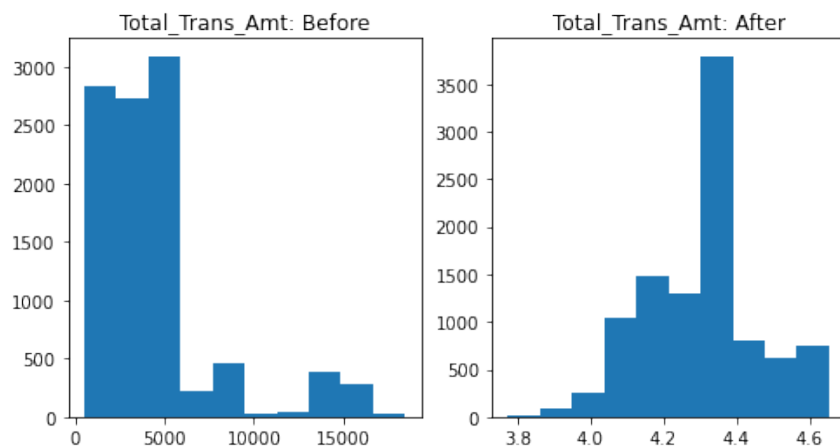


Рис. 16: Результат нормализации признака Total_Trans_Amt.

```

15
16
17 region = pd.get_dummies(data['region'])
18 data = pd.concat((data, region), axis=1)
19 data = data.loc[:, data.columns.isin(['age', 'sex', 'bmi', 'children
    ', 'smoker', 'charges', 'northeast', 'northwest', 'southeast', '
    southwest'])]
20 data.head(7)
21
22 data.info()
23
24 data.describe()
25
26
27 plt.figure(figsize=(12,8))
28 plt.scatter(data.age, data.charges, linewidth=0.8)
29
30 plt.xlabel('age')
31 plt.ylabel('charges')
32
33 data.corr()
34
35 plt.figure(figsize=(12,10), dpi= 80)
36 sns.heatmap(data.corr(), xticklabels=data.corr().columns,
    yticklabels=data.corr().columns, cmap='RdYlGn', center=0, annot=
    True)
37
38 plt.xticks(fontsize=12)
39 plt.yticks(fontsize=12)
40 plt.show()

```

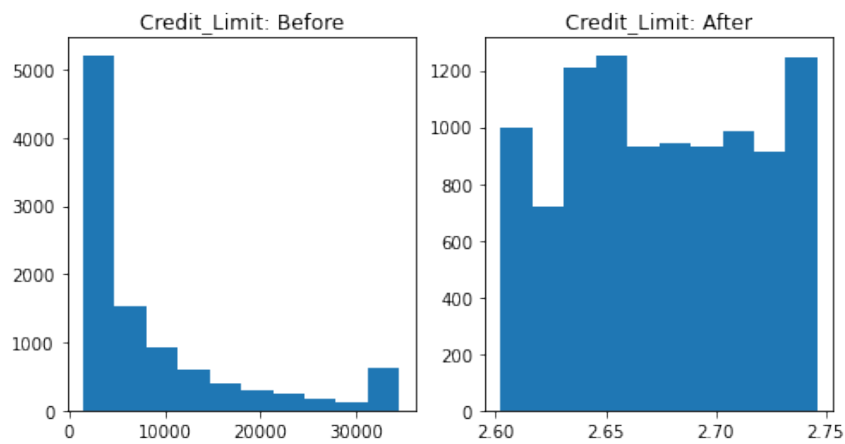



Рис. 17: Результат нормализации признака Credit_Limit.

```

41
42
43 plt.figure(figsize=(12,8))
44 plt.scatter(data.charges, data.children, linewidth=0.8)
45
46 plt.xlabel('charges')
47 plt.ylabel('children')
48
49 plt.figure(figsize=(12,8))
50 plt.scatter(data.age, data.children, linewidth=0.8)
51
52 plt.xlabel('age')
53 plt.ylabel('children')
54
55
56 plt.figure(figsize=(12,8))
57 plt.scatter(data.age, data.bmi, linewidth=0.8)
58
59 plt.xlabel('age')
60 plt.ylabel('bmi')
61
62 plt.figure(figsize=(12,8))
63 plt.scatter(data.charges, data.bmi, linewidth=0.8)
64
65 plt.xlabel('charges')
66 plt.ylabel('bmi')
67
68
69 data['sex'] = np.where(data['sex'] == 'female' , 0, data['sex'])
70 data['sex'] = np.where(data['sex'] == 'male' , 1, data['sex'])

```

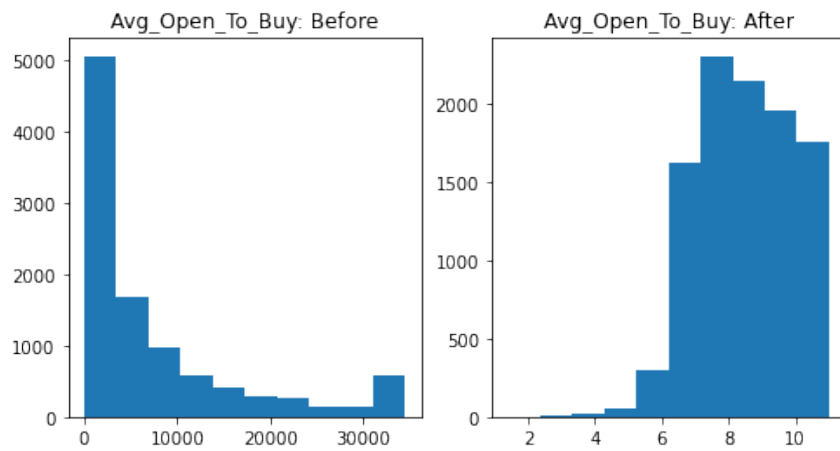


Рис. 18: Результат нормализации признака Avg_Open_To_Buy.

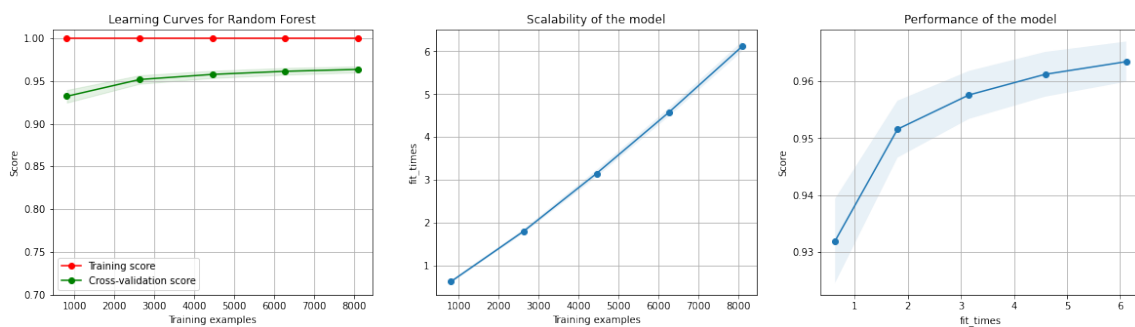


Рис. 19: Результат классификатора Random Forest.

```

71
72 data['smoker'] = np.where(data['smoker'] == 'no' , 0, data['smoker'
73 data['smoker'] = np.where(data['smoker'] == 'yes' , 1, data['smoker'
74
75 data['age'] = data['age']/data['age'].median()
76 data['bmi'] = data['bmi']/data['bmi'].median()
77 data['charges'] = data['charges']/data['charges'].median()
78 data['children'] = data['children']/data['children'].median()
79
80 plt.figure(figsize=(12,8))
81 plt.scatter(data.age, data.bmi, linewidth=0.8)
82
83 plt.xlabel('age')
84 plt.ylabel('bmi')

```

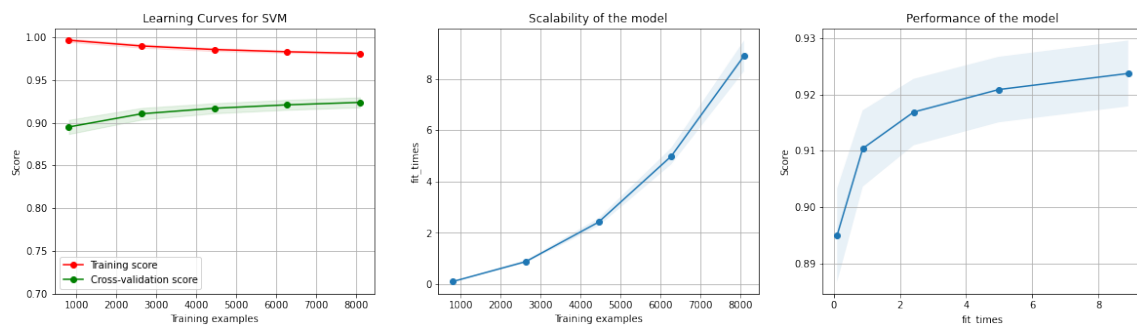


Рис. 20: Результат классификатора SVM.

```

85
86 plt.figure(figsize=(12,8))
87 plt.scatter(data.age, data.charges, linewidth=0.8)
88
89 plt.xlabel('age')
90 plt.ylabel('charges')
91
92 data.head(7)
93 data.describe()
94
95 target = data.charges
96 train = data.drop(['charges'], axis=1)
97
98 train.head(7)
99 data.corr()
100
101 plt.figure(figsize=(12,10), dpi= 80)
102 sns.heatmap(data.corr(), xticklabels=data.corr().columns,
103             yticklabels=data.corr().columns, cmap='RdYlGn', center=0, annot=
104             True)
105 plt.xticks(fontsize=12)
106 plt.yticks(fontsize=12)
107 plt.show()
108
109 sns.heatmap(data.corr(),annot=True)
110
111 from sklearn.model_selection import train_test_split
112 X_train, X_test, y_train, y_test = train_test_split(train, target,
113             test_size = 0.25, random_state = 42)
114
115 N_train, _ = X_train.shape
116 N_test, _ = X_test.shape
117 print (N_train, N_test)

```

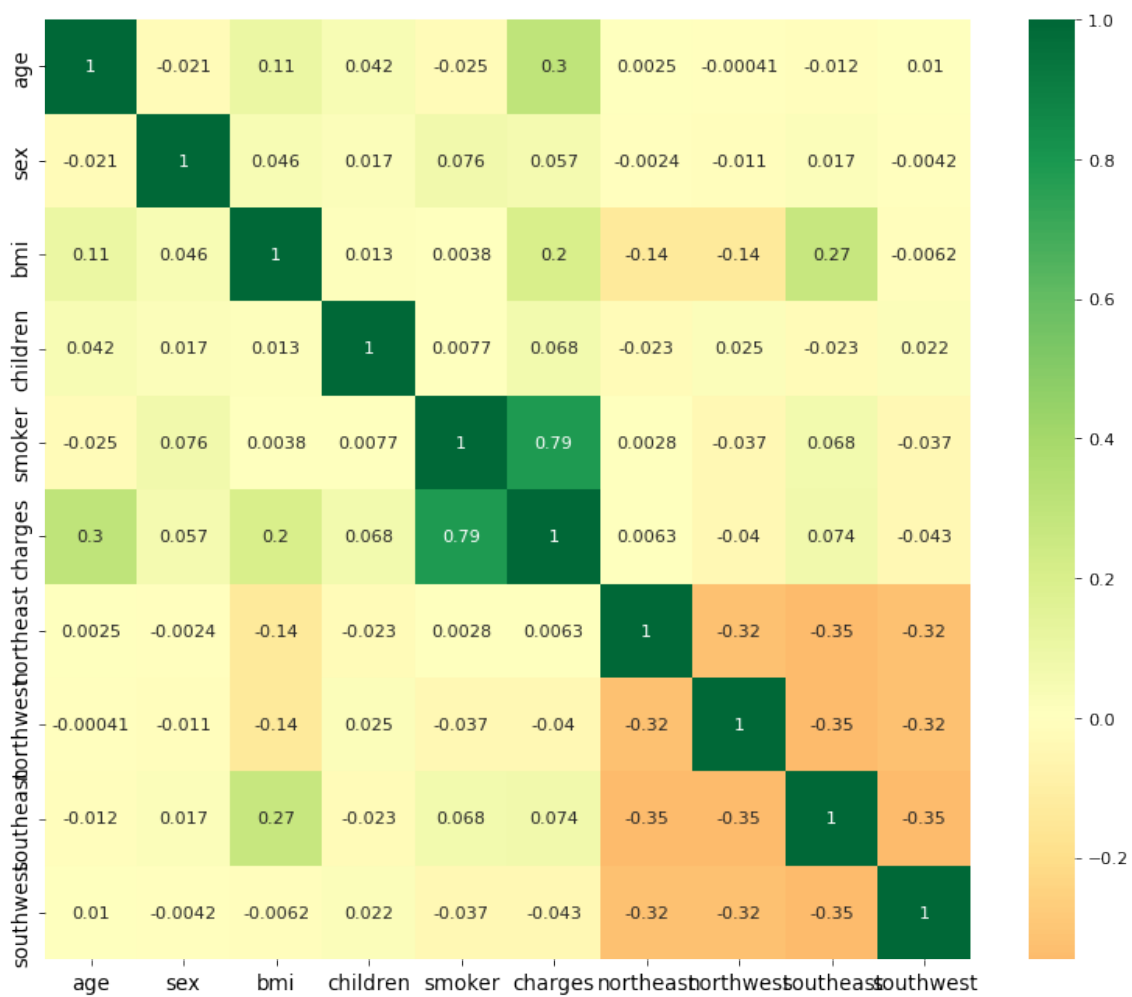


Рис. 21: Корреляция признаков датасета.

```

116
117 from sklearn.decomposition import PCA
118 import matplotlib.pyplot as plt
119 pca = PCA()
120 pca.fit(X_train)
121 X_pca = pca.transform(X_train)
122 for i, component in enumerate(pca.components_):
123     print("{} component: {}% of initial variance".format(i + 1,
124         round(100 * pca.explained_variance_ratio_[i], 2)))
125     print(" + ".join("%.3f x %s" % (value, name)
126         for value, name in zip(component, train.columns)
127     ))
128     print('\n')

```

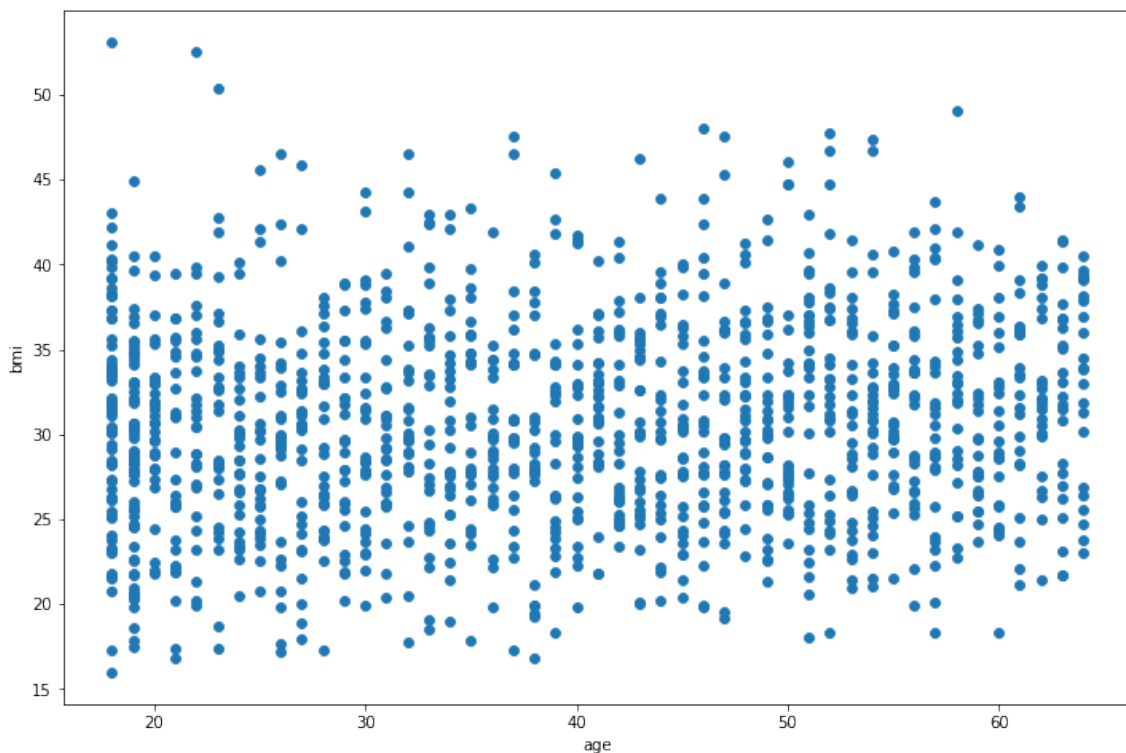


Рис. 22: График зависимости индекса массы тела от медицинских расходов.

```

128 plt.figure(figsize=(10,7))
129 plt.plot(np.cumsum(pca.explained_variance_ratio_), color='k', lw=2)
130 plt.axhline(0.9, c='r')
131 plt.axvline(4, c='b')
132

```

Листинг 9: Полный код решения Практической работы №5

```

1 init_notebook_mode(connected=True)
2 px.defaults.template = "plotly_white"
3 plot_df=ins.copy()
4 fig = px.box(plot_df, x="region", y="charges", color="region",
5             notched=True, points="outliers", height=600,
6             title="Distribution of Insurance Costs by Region",
7             color_discrete_sequence=['#B14B51', '#D0A99C', '#5D8370',
8                                     '#6C839B'])
9 fig.update_traces(marker=dict(size=9, opacity=0.5, line=dict(width
10                        =1,color="#F7F7F7")), showlegend=False)
11 fig.update_layout(font_color="#303030", xaxis_title='Region',
12                   yaxis_title='Claim Amount, $',

```

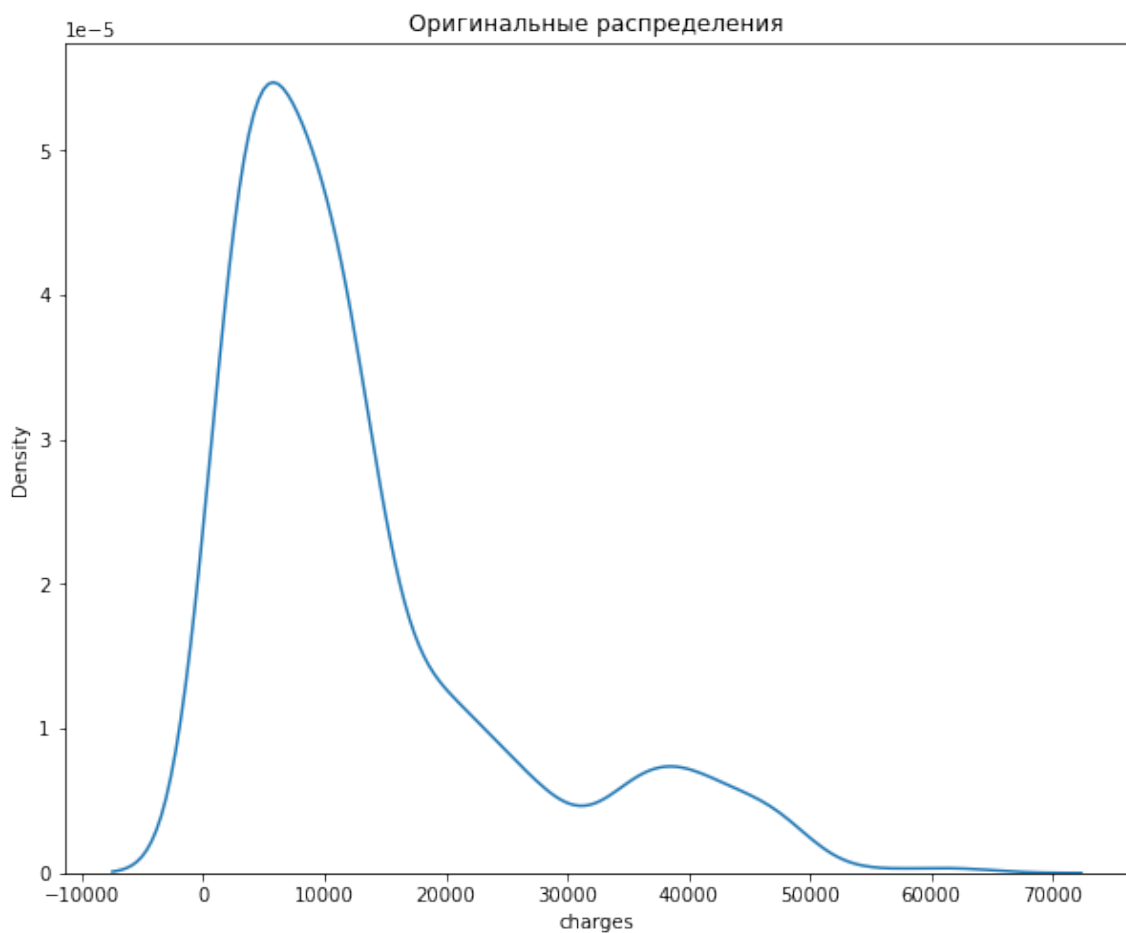


Рис. 23: Признак **charges** до стандартизации.

```

10         yaxis=dict(showgrid=True, gridwidth=1, gridcolor='
11         #EAEAEA', zerolinecolor='#EAEAEA'))
12 fig.show()

```

Листинг 10: Распределение стоимости медицинской страховки в зависимости от региона

```

1 import numpy as np
2 import pandas as pd
3 import seaborn as sns
4 import matplotlib.pyplot as plt
5 import plotly.express as px
6 import plotly.graph_objects as go
7 from plotly.offline import plot, iplot, init_notebook_mode

```



Рис. 24: Признак **age** до стандартизации.

```

8 from sklearn.preprocessing import StandardScaler, PolynomialFeatures
9 from sklearn.decomposition import PCA
10 from sklearn.linear_model import LinearRegression
11 from sklearn.ensemble import GradientBoostingRegressor
12 from sklearn.model_selection import train_test_split,
    RandomizedSearchCV
13 from sklearn.metrics import r2_score, mean_squared_error
14 import warnings
15 warnings.filterwarnings("ignore")
16 import math
17
18 # Вывозка датасета
19 ins = pd.read_csv('C:\\Users\\averu\\Documents\\git_local\\
    programming-practice\\IDA\\IDA-practice-5\\insurance.csv')
20 print("There are {:,} observations and {} columns in the data set.".
    format(ins.shape[0], ins.shape[1]))

```

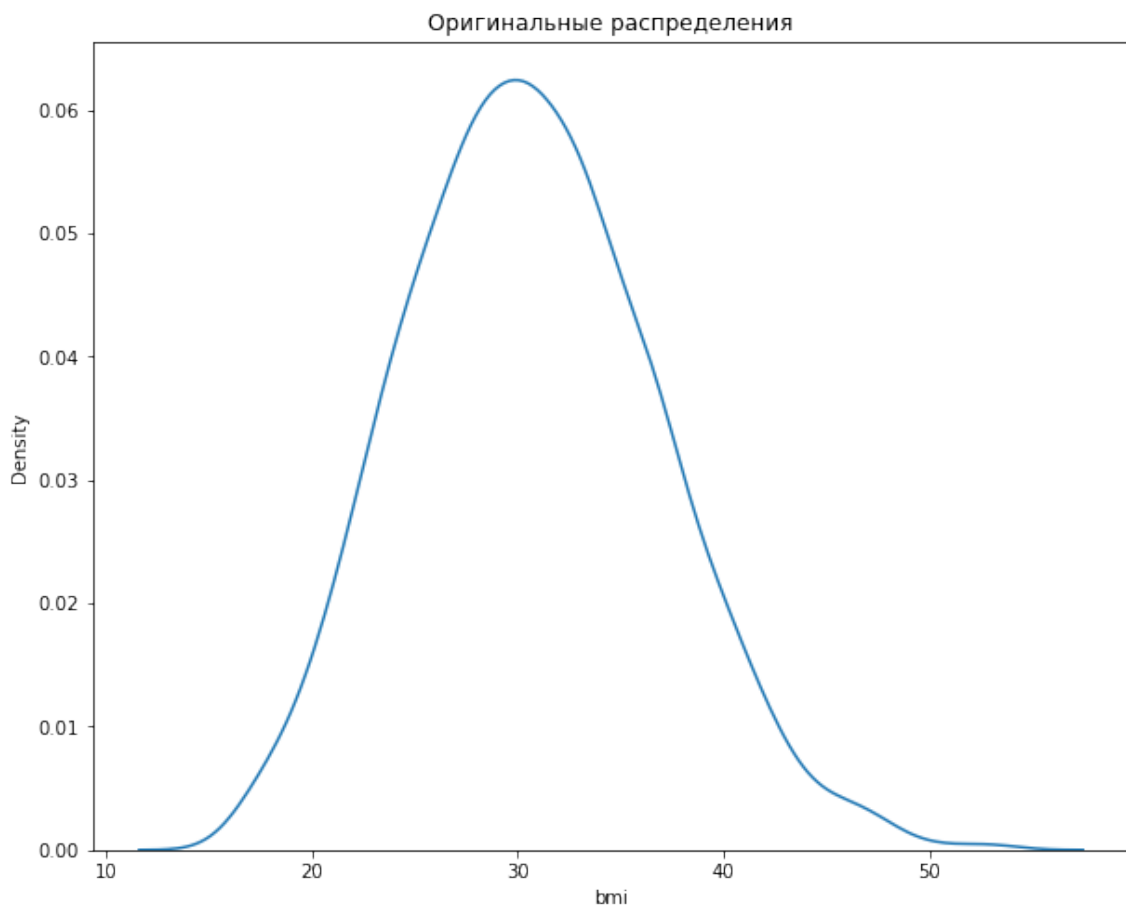


Рис. 25: Признак **bmi** до стандартизации.

```

21 print("There are {} missing values in the data.".format(ins.isna().
    sum().sum()))
22
23
24 ins['sex'] = ins['sex'].str.capitalize()
25 ins['smoker'] = ins['smoker'].apply(lambda x: 'Smoker' if x=='yes'
    else 'Non-Smoker')
26 ins['region'] = ins['region'].str.capitalize()
27
28
29 init_notebook_mode(connected=True)
30 px.defaults.template = "plotly_white"
31 plot_df=ins.copy()
32 fig = px.box(plot_df, x="region", y="charges", color="region",
33             notched=True, points="outliers", height=600,
34             title="",

```

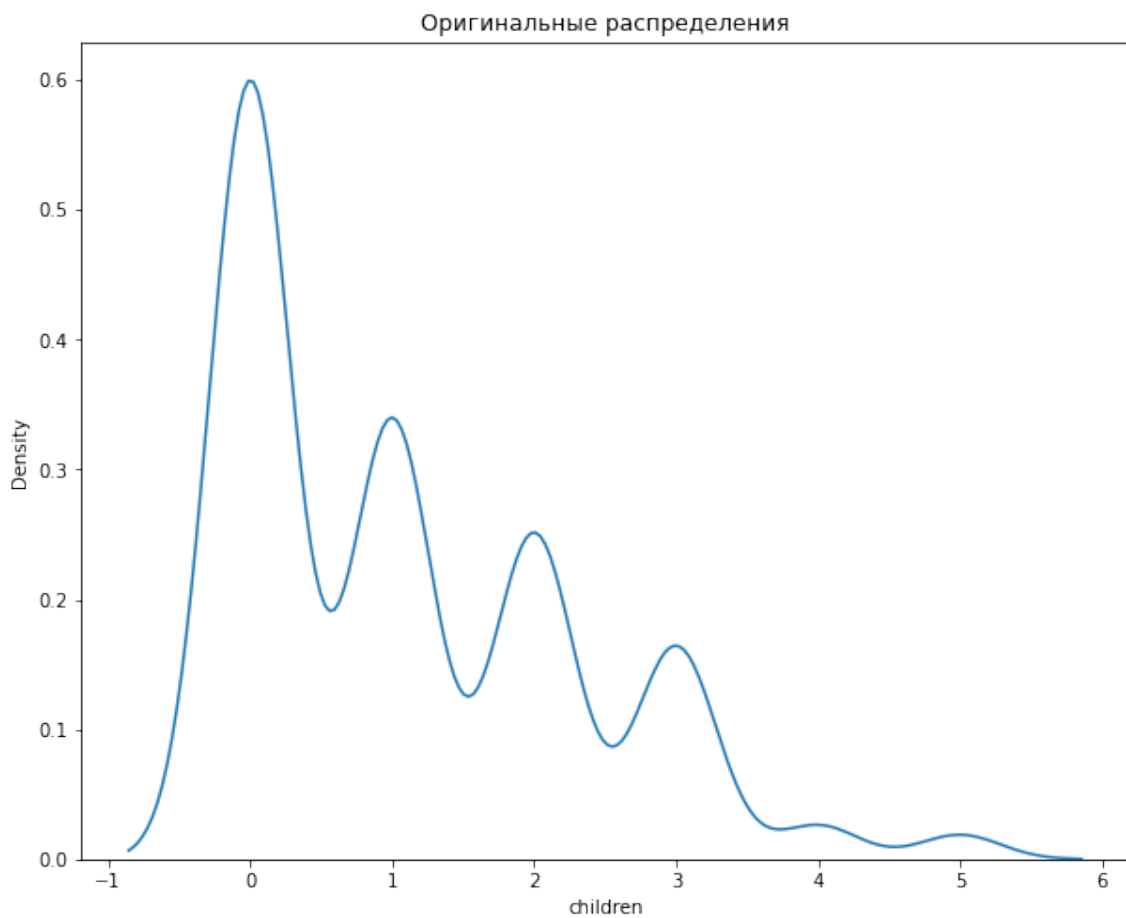



Рис. 26: Признак **children** до стандартизации.

```

35     color_discrete_sequence=['#B14B51', '#D0A99C', '#5D8370',
36     '#6C839B'])
36 fig.update_traces(marker=dict(size=9, opacity=0.5, line=dict(width
37     =1,color="#F7F7F7")), showlegend=False)
37 fig.update_layout(font_color="#303030", xaxis_title='Region',
38     yaxis_title='Claim Amount, $',
39     yaxis=dict(showgrid=True, gridwidth=1, gridcolor='
40     #EAEAEA', zerolinecolor='#EAEAEA'))
41 fig.show()
42
43 ins['female'] = ins['sex'].apply(lambda x: 1 if x=='Female' else 0)
44 ins['smoker_yes'] = ins['smoker'].apply(lambda x: 1 if x=='Smoker'
45     else 0)
46 ins.drop(['sex', 'smoker'], axis=1, inplace=True)

```

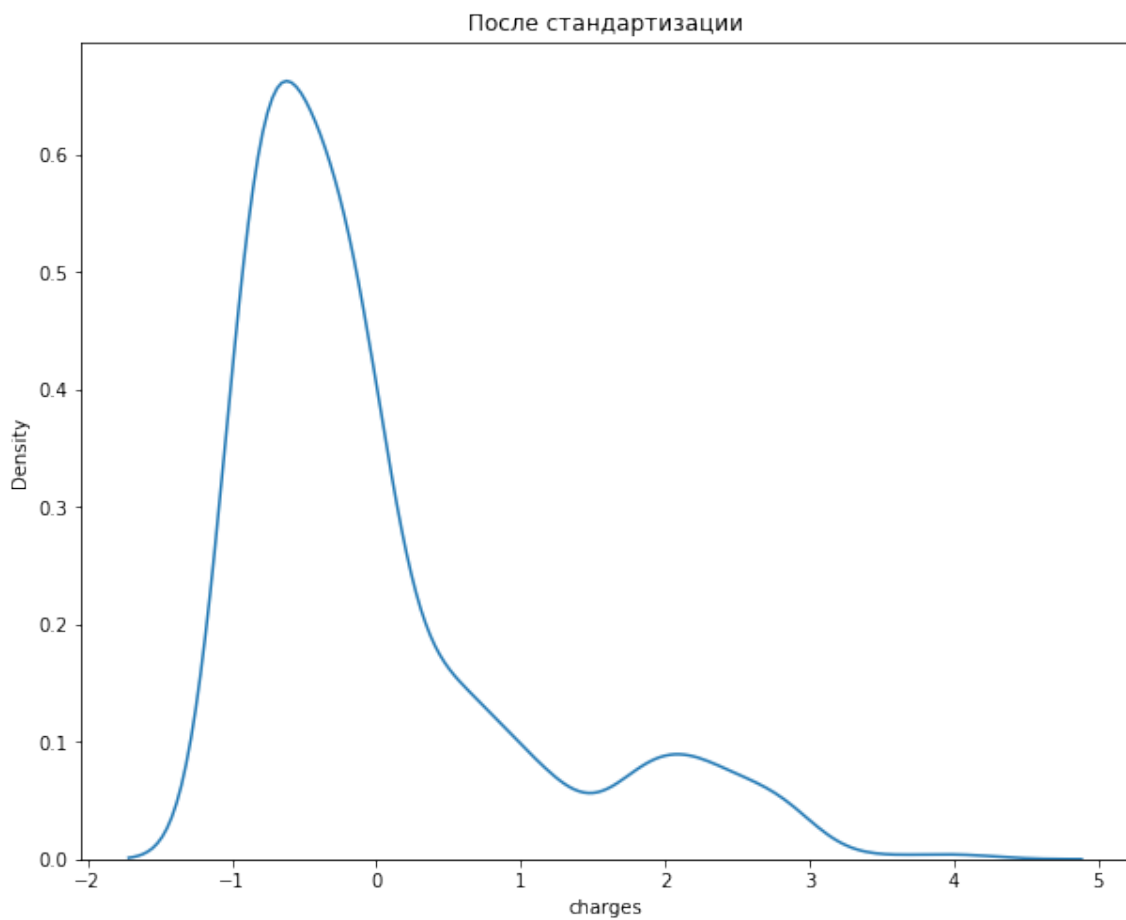


Рис. 27: Признак **charges** после стандартизации.

```

45
46 sns.set_context("notebook")
47 fig, ax = plt.subplots(figsize=(9,6))
48 corr=ins.corr()
49 mask=np.triu(np.ones_like(corr, dtype=bool))[1:, :-1]
50 corr=corr.iloc[1:,:-1].copy()
51 ax=sns.heatmap(corr, mask=mask, vmin=-.1, vmax=.9, center=0, annot=
    True, fmt='.2f',
52                cmap='ocean', linewidths=4, annot_kws={"fontsize"
    :12})
53 ax.set_title('', fontsize=18)
54 ax.set_xticklabels(ax.get_xticklabels(), rotation=45,
    horizontalalignment='right', fontsize=12)
55 ax.set_yticklabels(ax.get_yticklabels(), fontsize=12)
56 fig.show()

```

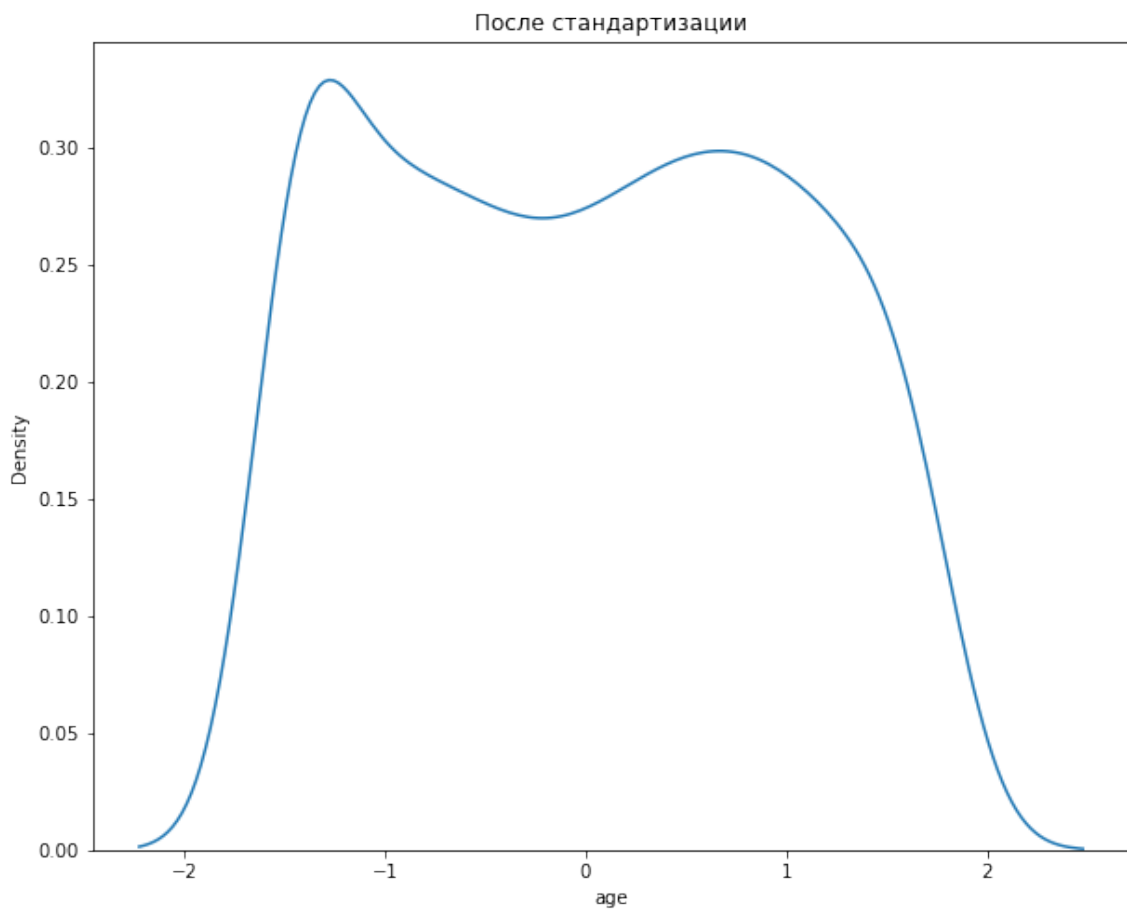


Рис. 28: Признак **age** после стандартизации.

```

57
58
59 data = pd.read_csv('C:\\Users\\averu\\Documents\\git_local\\
    programming-practice\\IDA\\IDA-practice-5\\insurance.csv')
60
61 data['sex'] = data['sex'].str.capitalize()
62 data['smoker'] = data['smoker'].apply(lambda x: 'Smoker' if x=='yes'
    else 'Non-Smoker')
63
64 data['female'] = data['sex'].apply(lambda x: 1 if x=='Female' else
    0)
65 data['smoker_yes'] = data['smoker'].apply(lambda x: 1 if x=='Smoker'
    else 0)
66
67
68 region = pd.get_dummies(data['region'])

```

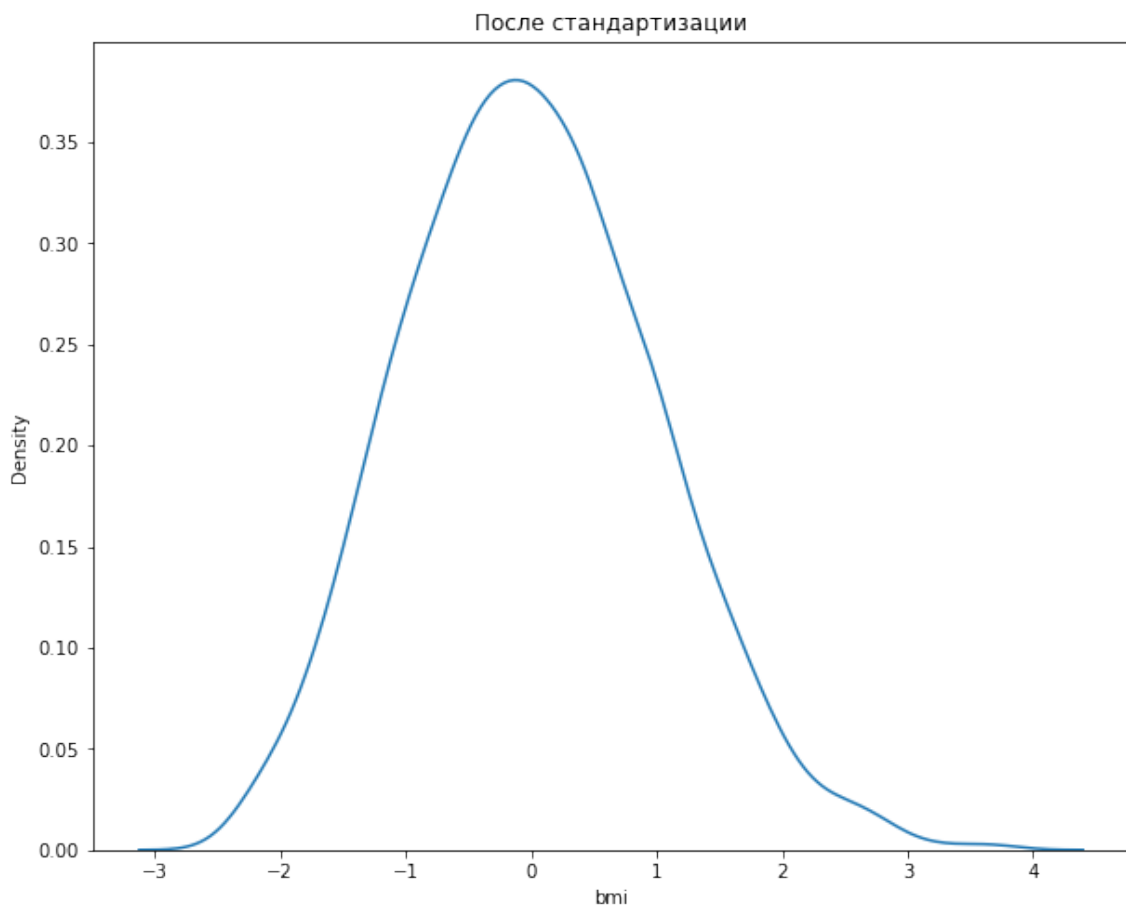


Рис. 29: Признак **bmi** после стандартизации.

```

69 data = pd.concat((data, region), axis=1)
70 data = data.loc[:, data.columns.isin(['age', 'female', 'bmi', '
    children', 'smoker_yes', 'charges', 'northeast', 'northwest', '
    southeast', 'southwest'])]
71
72 data.head(7)
73
74 y_train=data.pop('charges')
75 X_train = data.copy()
76
77 import statsmodels.api as sm
78 X_train_sm = sm.add_constant(X_train)
79
80 lm= sm.OLS(y_train, X_train_sm).fit()
81
82 lm.summary()
```

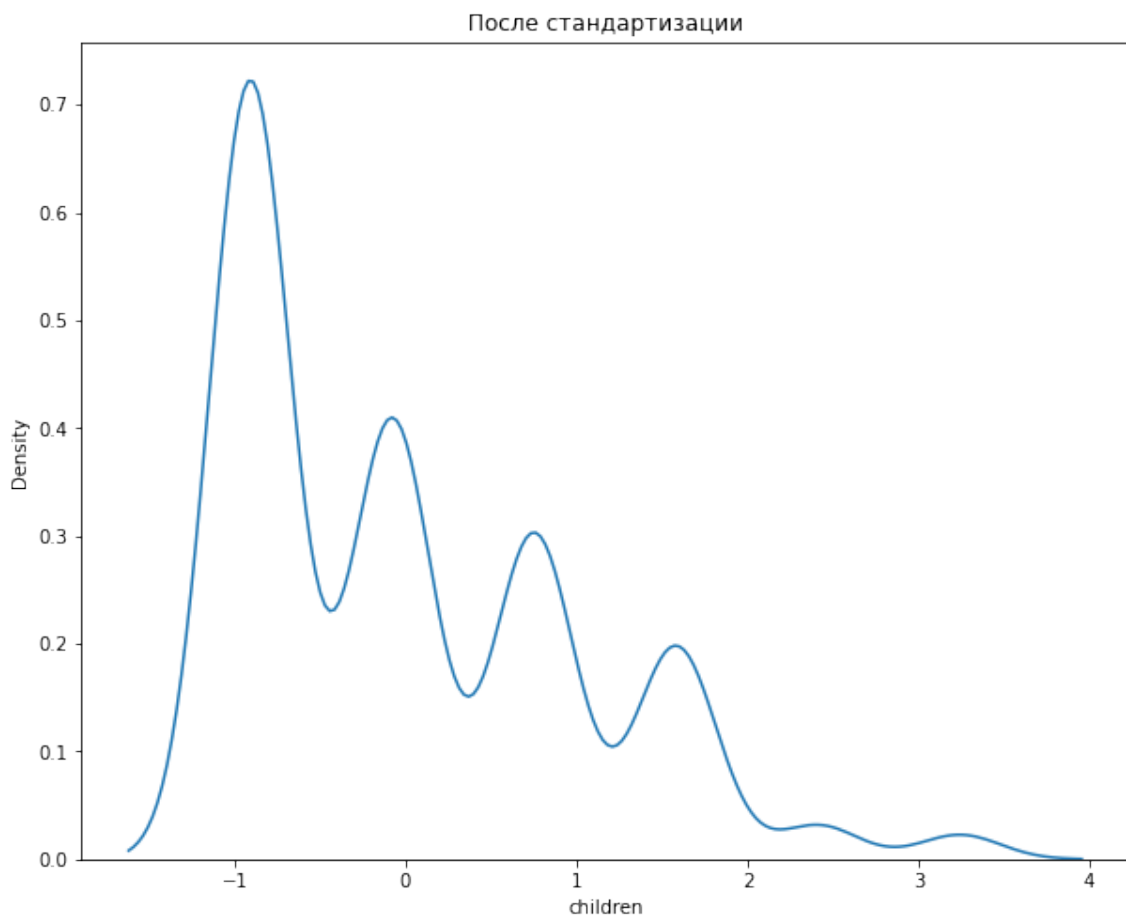


Рис. 30: Признак **children** после стандартизации.

```

83
84
85
86 # y_train=data.pop('charges')
87 X_train = data.copy()
88 for i in range(0 , len(X_train['bmi'])):
89     X_train['bmi'][i] = X_train['bmi'][i] * X_train['bmi'][i]
90
91 # X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))
92
93 import statsmodels.api as sm
94 X_train_sm = sm.add_constant(X_train)
95
96 bmi_sq= sm.OLS(y_train, X_train_sm).fit()
97

```

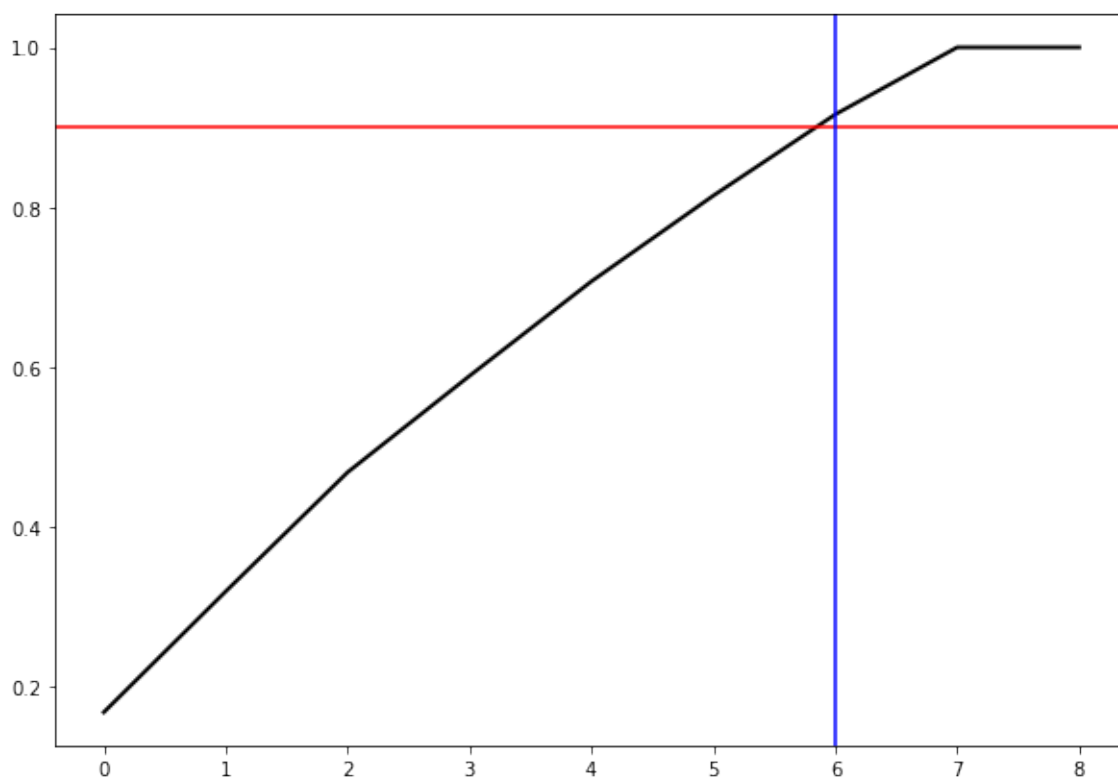


Рис. 31: Визуализация зависимости качества описываемой дисперсии от количество описывающих признаков

```

98 bmi_sq.summary()
99
100
101 # y_train=data.pop('charges')
102 X_train = data.copy()
103 for i in range(0 , len(X_train['bmi'])):
104     X_train['bmi'][i] = X_train['bmi'][i] * X_train['bmi'][i] *
105         X_train['bmi'][i]
106 # X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))
107
108 import statsmodels.api as sm
109 X_train_sm = sm.add_constant(X_train)
110
111 bmi_sq= sm.OLS(y_train, X_train_sm).fit()
112
113 bmi_sq.summary()
114

```

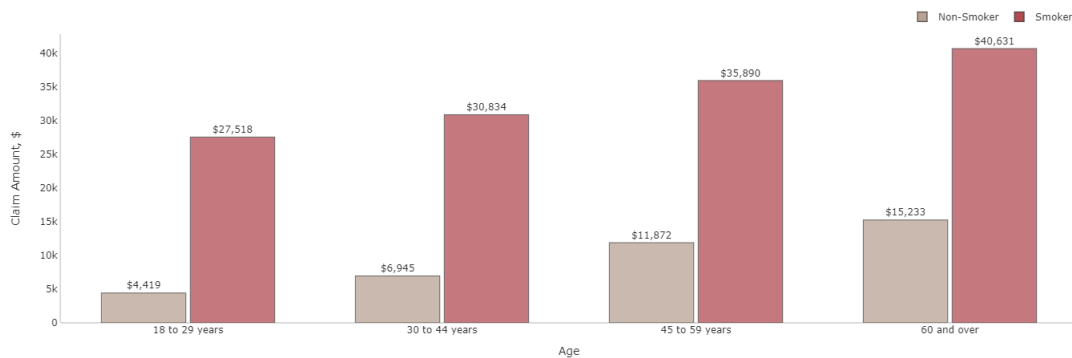


Рис. 32: Визуализация распределения стоимости медицинской страховки среди курящих и некурящих разных возрастов.

```

115
116 # y_train=data.pop('charges')
117 X_train = data.copy()
118 for i in range(0 , len(X_train['age'])):
119     X_train['age'][i] = X_train['age'][i] * X_train['age'][i] *
120     X_train['age'][i]
121 # X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))
122
123 import statsmodels.api as sm
124 X_train_sm = sm.add_constant(X_train)
125
126 bmi_sq= sm.OLS(y_train, X_train_sm).fit()
127
128 bmi_sq.summary()
129
130 # y_train=data.pop('charges')
131 X_train = data.copy()
132 for i in range(0 , len(X_train['age'])):
133     X_train['age'][i] = pow(X_train['age'][i], 0.4)
134
135 # X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))
136
137 import statsmodels.api as sm
138 X_train_sm = sm.add_constant(X_train)
139
140 bmi_sq= sm.OLS(y_train, X_train_sm).fit()
141
142 bmi_sq.summary()
143
144

```

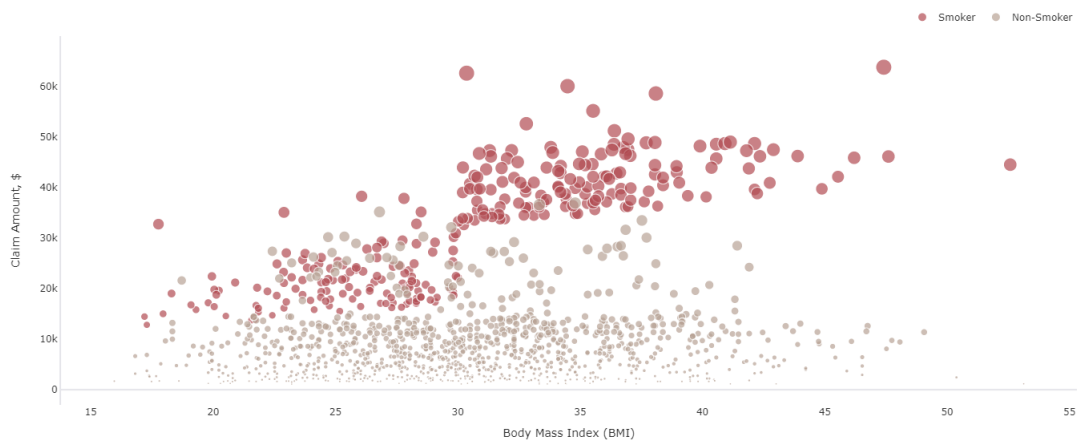


Рис. 33: Визуализация распределения стоимости медицинской страховки среди курящих и некурящих разных возрастов.

```

145 # y_train=data.pop('charges')
146 X_train = data.copy()
147 for i in range(0 , len(X_train['age'])):
148     X_train['age'][i] = pow(X_train['age'][i], 0.8)
149
150 # X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))
151
152 import statsmodels.api as sm
153 X_train_sm = sm.add_constant(X_train)
154
155 bmi_sq= sm.OLS(y_train, X_train_sm).fit()
156
157 bmi_sq.summary()
158
159
160 # y_train=data.pop('charges')
161 X_train = data.copy()
162 for i in range(0 , len(X_train['age'])):
163     X_train['age'][i] = pow(X_train['age'][i], 1.2)
164
165 # X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))
166
167 import statsmodels.api as sm
168 X_train_sm = sm.add_constant(X_train)
169
170 bmi_sq= sm.OLS(y_train, X_train_sm).fit()
171
172 bmi_sq.summary()

```

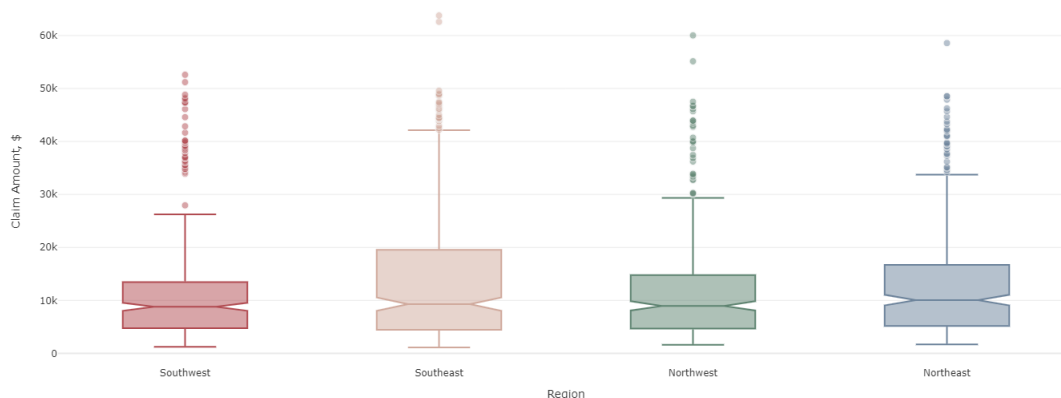



Рис. 34: Визуализация стоимости медицинской страховки к каждому региону

```

173
174 # y_train=data.pop('charges')
175 X_train = data.copy()
176 for i in range(0 , len(X_train['age'])):
177     X_train['age'][i] = pow(X_train['age'][i], 1.6)
178
179 # X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))
180
181 import statsmodels.api as sm
182 X_train_sm = sm.add_constant(X_train)
183
184 bmi_sq= sm.OLS(y_train, X_train_sm).fit()
185
186 bmi_sq.summary()
187
188 # y_train=data.pop('charges')
189 X_train = data.copy()
190 for i in range(0 , len(X_train['bmi'])):
191     X_train['bmi'][i] = pow(X_train['bmi'][i], 0.4)
192
193 # X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))
194
195 import statsmodels.api as sm
196 X_train_sm = sm.add_constant(X_train)
197
198 bmi_sq= sm.OLS(y_train, X_train_sm).fit()
199
200 bmi_sq.summary()

```

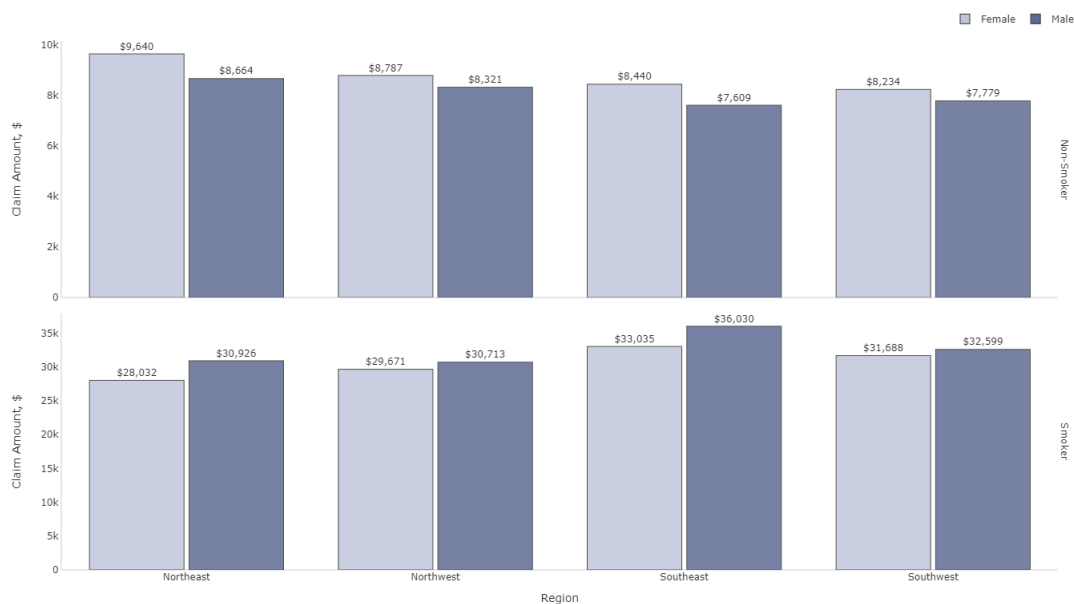


Рис. 35: Визуализация распределения стоимости медицинской страховки.

```

201
202 # y_train=data.pop('charges')
203 X_train = data.copy()
204 for i in range(0 , len(X_train['bmi'])):
205     X_train['bmi'][i] = pow(X_train['bmi'][i], 0.8)
206
207 # X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))
208
209 import statsmodels.api as sm
210 X_train_sm = sm.add_constant(X_train)
211
212 bmi_sq= sm.OLS(y_train, X_train_sm).fit()
213
214 bmi_sq.summary()
215
216
217 # y_train=data.pop('charges')
218 X_train = data.copy()
219 for i in range(0 , len(X_train['bmi'])):
220     X_train['bmi'][i] = pow(X_train['bmi'][i], 1.2)
221
222 # X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))
223

```

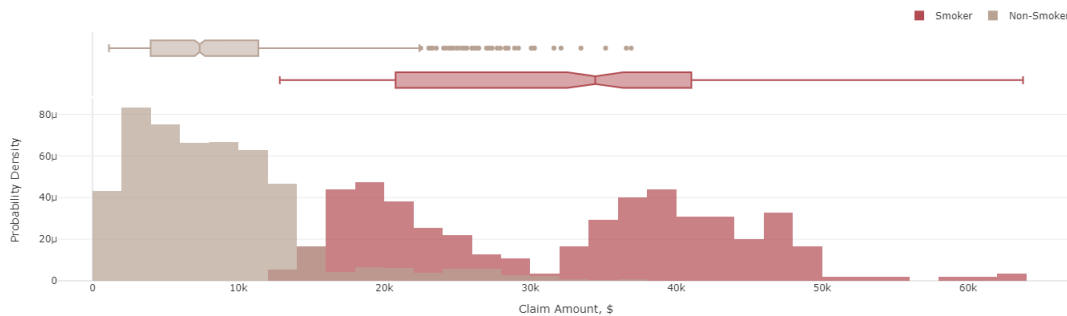


Рис. 36: Визуализация распределения стоимости медицинской страховки среди курящих и некурящих.

```

224 import statsmodels.api as sm
225 X_train_sm = sm.add_constant(X_train)
226
227 bmi_sq= sm.OLS(y_train, X_train_sm).fit()
228
229 bmi_sq.summary()
230
231
232
233 # y_train=data.pop('charges')
234 X_train = data.copy()
235 for i in range(0, len(X_train['bmi'])):
236     X_train['bmi'][i] = pow(X_train['bmi'][i], 1.6)
237
238 # X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))
239
240 import statsmodels.api as sm
241 X_train_sm = sm.add_constant(X_train)
242
243 bmi_sq= sm.OLS(y_train, X_train_sm).fit()
244
245 bmi_sq.summary()
246
247
248 # y_train=data.pop('charges')
249 X_train = data.copy()
250 for i in range(0, len(X_train['bmi'])):
251     X_train['bmi'][i] = pow(X_train['bmi'][i], 1.6)
252     X_train['age'][i] = pow(X_train['age'][i], 0.4)
253 # X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))
254
255 import statsmodels.api as sm

```

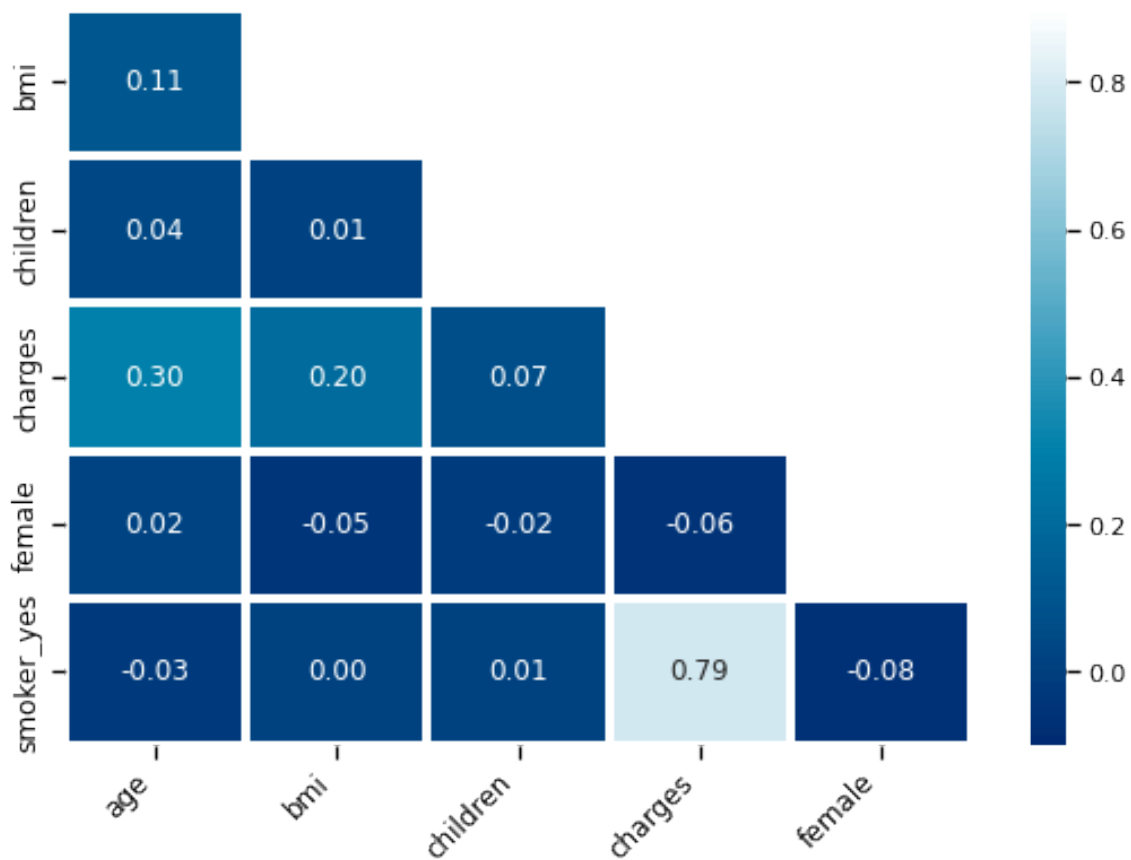


Рис. 37: Корреляционная матрица.

```

256 X_train_sm = sm.add_constant(X_train)
257
258 bmi_sq= sm.OLS(y_train, X_train_sm).fit()
259
260 bmi_sq.summary()
261
262
263 # y_train=data.pop('charges')
264 X_train = data.copy()
265 for i in range(0, len(X_train['bmi'])):
266     X_train['bmi'][i] = pow(X_train['bmi'][i], 1.6)
267     X_train['age'][i] = pow(X_train['age'][i], 0.8)
268 # X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))
269
270 import statsmodels.api as sm
271 X_train_sm = sm.add_constant(X_train)
272

```

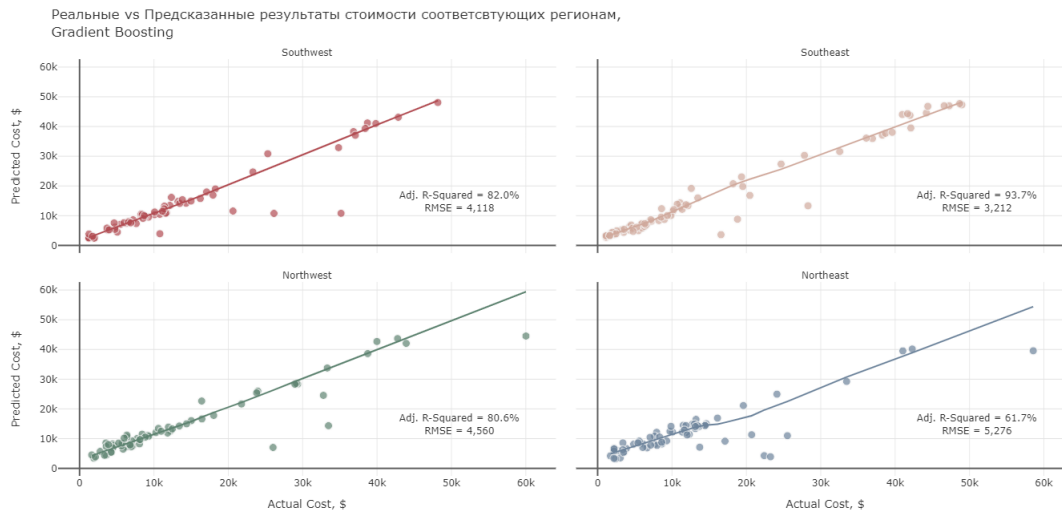


Рис. 38: Результат работы модели.

```

273 bmi_sq= sm.OLS(y_train, X_train_sm).fit()
274
275 bmi_sq.summary()
276
277
278
279 # y_train=data.pop('charges')
280 X_train = data.copy()
281 for i in range(0, len(X_train['bmi'])):
282     X_train['bmi'][i] = pow(X_train['bmi'][i], 1.6)
283     X_train['age'][i] = pow(X_train['age'][i], 1.2)
284 # X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))
285
286 import statsmodels.api as sm
287 X_train_sm = sm.add_constant(X_train)
288
289 bmi_sq= sm.OLS(y_train, X_train_sm).fit()
290
291 bmi_sq.summary()
292
293
294
295 # y_train=data.pop('charges')
296 X_train = data.copy()
297 for i in range(0, len(X_train['bmi'])):
298     X_train['bmi'][i] = pow(X_train['bmi'][i], 1.6)
299     X_train['age'][i] = pow(X_train['age'][i], 1.6)

```

```

300 # X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))
301
302 import statsmodels.api as sm
303 X_train_sm = sm.add_constant(X_train)
304
305 bmi_sq= sm.OLS(y_train, X_train_sm).fit()
306
307 bmi_sq.summary()
308
309
310
311 # y_train=data.pop('charges')
312 X_train = data.copy()
313 for i in range(0 , len(X_train['bmi'])):
314     X_train['bmi'][i] = pow(X_train['bmi'][i], 1.2)
315     X_train['age'][i] = pow(X_train['age'][i], 0.4)
316 # X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))
317
318 import statsmodels.api as sm
319 X_train_sm = sm.add_constant(X_train)
320
321 bmi_sq= sm.OLS(y_train, X_train_sm).fit()
322
323 bmi_sq.summary()
324
325
326
327 # y_train=data.pop('charges')
328 X_train = data.copy()
329 for i in range(0 , len(X_train['bmi'])):
330     X_train['bmi'][i] = pow(X_train['bmi'][i], 1.2)
331     X_train['age'][i] = pow(X_train['age'][i], 0.8)
332 # X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))
333
334 import statsmodels.api as sm
335 X_train_sm = sm.add_constant(X_train)
336
337 bmi_sq= sm.OLS(y_train, X_train_sm).fit()
338
339 bmi_sq.summary()
340
341
342
343
344
345
346 # y_train=data.pop('charges')
347 X_train = data.copy()
348 for i in range(0 , len(X_train['bmi'])):

```

```

349     X_train['bmi'][i] = pow(X_train['bmi'][i], 1.2)
350     X_train['age'][i] = pow(X_train['age'][i], 0.8)
351 # X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))
352
353 import statsmodels.api as sm
354 X_train_sm = sm.add_constant(X_train)
355
356 bmi_sq= sm.OLS(y_train, X_train_sm).fit()
357
358 bmi_sq.summary()
359
360
361
362 # y_train=data.pop('charges')
363 X_train = data.copy()
364 for i in range(0 , len(X_train['bmi'])):
365     X_train['bmi'][i] = pow(X_train['bmi'][i], 1.2)
366     X_train['age'][i] = pow(X_train['age'][i], 1.6)
367 # X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))
368
369 import statsmodels.api as sm
370 X_train_sm = sm.add_constant(X_train)
371
372 bmi_sq= sm.OLS(y_train, X_train_sm).fit()
373
374 bmi_sq.summary()
375
376
377
378
379
380 # y_train=data.pop('charges')
381 X_train = data.copy()
382 for i in range(0 , len(X_train['bmi'])):
383     X_train['bmi'][i] = pow(X_train['bmi'][i], 0.8)
384     X_train['age'][i] = pow(X_train['age'][i], 1.6)
385 # X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))
386
387 import statsmodels.api as sm
388 X_train_sm = sm.add_constant(X_train)
389
390 bmi_sq= sm.OLS(y_train, X_train_sm).fit()
391
392 bmi_sq.summary()
393
394
395 # y_train=data.pop('charges')
396 X_train = data.copy()
397 for i in range(0 , len(X_train['bmi'])):

```

```

398     X_train['bmi'][i] = pow(X_train['bmi'][i], 0.8)
399     X_train['age'][i] = pow(X_train['age'][i], 1.2)
400 # X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))
401
402 import statsmodels.api as sm
403 X_train_sm = sm.add_constant(X_train)
404
405 bmi_sq= sm.OLS(y_train, X_train_sm).fit()
406
407 bmi_sq.summary()
408
409
410 # y_train=data.pop('charges')
411 X_train = data.copy()
412 for i in range(0 , len(X_train['bmi'])):
413     X_train['bmi'][i] = pow(X_train['bmi'][i], 0.8)
414     X_train['age'][i] = pow(X_train['age'][i], 0.8)
415 # X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))
416
417 import statsmodels.api as sm
418 X_train_sm = sm.add_constant(X_train)
419
420 bmi_sq= sm.OLS(y_train, X_train_sm).fit()
421
422 bmi_sq.summary()
423
424
425
426 # y_train=data.pop('charges')
427 X_train = data.copy()
428 for i in range(0 , len(X_train['bmi'])):
429     X_train['bmi'][i] = pow(X_train['bmi'][i], 0.8)
430     X_train['age'][i] = pow(X_train['age'][i], 0.4)
431 # X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))
432
433 import statsmodels.api as sm
434 X_train_sm = sm.add_constant(X_train)
435
436 bmi_sq= sm.OLS(y_train, X_train_sm).fit()
437
438 bmi_sq.summary()
439
440
441
442
443
444
445 # y_train=data.pop('charges')
446 X_train = data.copy()

```



```

447 for i in range(0 , len(X_train['age'])):
448     X_train['age'][i] = X_train['age'][i] * X_train['age'][i]
449
450
451 import statsmodels.api as sm
452 X_train_sm = sm.add_constant(X_train)
453
454 age_sq= sm.OLS(y_train, X_train_sm).fit()
455
456 age_sq.summary()
457
458
459
460
461
462 # y_train=data.pop('charges')
463 X_train = data.copy()
464 for i in range(0 , len(X_train['age'])):
465     X_train['age'][i] = X_train['age'][i] * X_train['age'][i]
466     X_train['bmi'][i] = X_train['bmi'][i] * X_train['bmi'][i]
467
468
469 import statsmodels.api as sm
470 X_train_sm = sm.add_constant(X_train)
471
472 bmi_age_sq= sm.OLS(y_train, X_train_sm).fit()
473
474 bmi_age_sq.summary()
475
476 # y_train=data.pop('charges')
477 X_train = data.copy()
478 for i in range(0 , len(X_train['age'])):
479     X_train['age'][i] = X_train['age'][i] * X_train['age'][i]
480     X_train['bmi'][i] = X_train['bmi'][i] * X_train['bmi'][i]
481
482
483 import statsmodels.api as sm
484 X_train_sm = sm.add_constant(X_train)
485
486 lm_5= sm.OLS(y_train, X_train_sm).fit()
487 lm_5.summary()
488
489 # y_train=data.pop('charges')
490
491 X_train = data.copy()
492 for i in range(0 , len(X_train['age'])):
493     X_train['age'][i] = math.log(X_train['age'][i],10)
494
495

```

```

496
497 import statsmodels.api as sm
498 X_train_sm = sm.add_constant(X_train)
499
500 age_lg= sm.OLS(y_train, X_train_sm).fit()
501 age_lg.summary()
502
503
504 # y_train=data.pop('charges')
505 import math
506 X_train = data.copy()
507 for i in range(0 , len(X_train['bmi'])):
508     X_train['bmi'][i] = math.log(X_train['bmi'][i],10)
509
510
511
512 import statsmodels.api as sm
513 X_train_sm = sm.add_constant(X_train)
514
515 bmi_lg= sm.OLS(y_train, X_train_sm).fit()
516 bmi_lg.summary()
517
518 # y_train=data.pop('charges')
519
520 X_train = data.copy()
521 for i in range(0 , len(X_train['bmi'])):
522     X_train['bmi'][i] = math.log(X_train['bmi'][i],10)
523     X_train['age'][i] = math.log(X_train['age'][i],10)
524
525
526 import statsmodels.api as sm
527 X_train_sm = sm.add_constant(X_train)
528
529 bmi_age_lg= sm.OLS(y_train, X_train_sm).fit()
530 bmi_age_lg.summary()
531
532
533 X_train = data.copy()
534 for i in range(0 , len(X_train['bmi'])):
535     X_train['bmi'][i] = math.log(X_train['bmi'][i],10)
536     X_train['age'][i] = X_train['age'][i] * X_train['age'][i]
537
538
539 import statsmodels.api as sm
540 X_train_sm = sm.add_constant(X_train)
541
542 lm_16= sm.OLS(y_train, X_train_sm).fit()
543 lm_16.summary()
544

```

```

545 # y_train=data.pop('charges')
546
547
548 X_train = data.loc[:, data.columns.isin(['bmi', 'smoker_yes',])]]
549
550 import statsmodels.api as sm
551 X_train_sm = sm.add_constant(X_train)
552
553 lm_9= sm.OLS(y_train, X_train_sm).fit()
554 lm_9.summary()
555
556 # y_train=data.pop('charges')
557
558 X_train = data.loc[:, data.columns.isin(['age', 'smoker_yes',])]]
559
560
561 import statsmodels.api as sm
562 X_train_sm = sm.add_constant(X_train)
563
564 age_smoke= sm.OLS(y_train, X_train_sm).fit()
565 age_smoke.summary()
566
567
568 # y_train=data.pop('charges')
569 X_train = data.copy()
570 for i in range(0 , len(X_train['smoker_yes'])):
571     X_train['smoker_yes'][i] = X_train['smoker_yes'][i] * X_train['
        smoker_yes'][i]
572
573
574 import statsmodels.api as sm
575 X_train_sm = sm.add_constant(X_train)
576
577 lm_11= sm.OLS(y_train, X_train_sm).fit()
578 lm_11.summary()
579
580 # y_train=data.pop('charges')
581 X_train = data.loc[:, data.columns.isin(['age', 'bmi',])]]
582
583
584 import statsmodels.api as sm
585 X_train_sm = sm.add_constant(X_train)
586
587 lm_12= sm.OLS(y_train, X_train_sm).fit()
588 lm_12.summary()
589
590
591
592 # y_train=data.pop('charges')

```

```

593 y_train=data.pop('smoker_yes')
594 # X_train = [data['female'].copy(), data['bmi'].copy()]
595 X_train = data.loc[:, data.columns.isin(['age', 'female', 'bmi', '
    children', 'charges', 'northeast', 'northwest', 'southeast', '
    southwest'])]
596
597 import statsmodels.api as sm
598 X_train_sm = sm.add_constant(X_train)
599
600 lm_13= sm.OLS(y_train, X_train_sm).fit()
601 lm_13.summary()
602
603
604 # y_train=data.pop('charges')
605 y_train=data.pop('bmi')
606 # X_train = [data['female'].copy(), data['bmi'].copy()]
607 X_train = data.loc[:, data.columns.isin(['age', 'female', '
    smoker_yes', 'children', 'charges', 'northeast', 'northwest', '
    southeast', 'southwest'])]
608
609 import statsmodels.api as sm
610 X_train_sm = sm.add_constant(X_train)
611
612 lm_20= sm.OLS(y_train, X_train_sm).fit()
613 lm_20.summary()
614
615
616 # Dataframe to save results
617
618
619 regions = ins.region.unique()
620 s = StandardScaler()
621 # feat_importance=pd.DataFrame()
622 actuals=[]
623 preds=[]
624 rmse=[]
625 r2_scores=[]
626 adj_r2_scores=[]
627
628 for i in regions:
629
630     # Filter data by region
631     print("\nRegion: {}".format(i))
632     ins_df = ins[ins.region==i]
633     X=ins_df.drop(['charges', 'region'], axis=1)
634     y=ins_df.charges
635
636     # Add polynomial features
637     pf = PolynomialFeatures(degree=2, include_bias=False)

```

```

638 X_pf = pd.DataFrame(data=pf.fit_transform(X), columns=pf.
get_feature_names(X.columns))
639
640 # Create training and test sets
641 X_train, X_test, y_train, y_test = train_test_split(X_pf, y,
test_size=0.2, random_state=1)
642
643 # Scale features
644 X_train_scaled = s.fit_transform(X_train)
645 X_test_scaled = s.transform(X_test)
646
647 # PCA
648 pca = PCA(.95)
649 X_train_pca=pca.fit_transform(X_train_scaled)
650 X_test_pca=pca.transform(X_test_scaled)
651 print("Number of Principal Components = {}".format(pca.
n_components_))
652 print("Train Shape:{} {} Test Shape:{} {}".format(X_train_pca.
shape, y_train.shape, X_test_pca.shape, y_test.shape))
653
654 # Linear Regression
655 lr = LinearRegression().fit(X_train_pca, y_train)
656 y_pred=lr.predict(X_test_pca)
657 rmse=np.sqrt(mean_squared_error(y_test, y_pred)).round(2)
658 r2=r2_score(y_test, y_pred)
659 adj_r2 = 1 - (1-r2)*(len(y_test)-1)/(len(y_test)-X_test_pca.
shape[1]-1)
660
661 actuals.append(pd.Series(y_test, name='actuals').reset_index())
662 preds.append(pd.Series(y_pred, name='preds').reset_index(drop=
True))
663 rmse.append(rmse)
664 r2_scores.append(r2)
665 adj_r2_scores.append(adj_r2)
666
667 # feat_importance["Importance_"+str(i)]=lr.feature_importances_
668
669 print("Test Error (RMSE) = {:,}".format(rmse))
670 print("R-Squared = {:.2f}%, Adjusted R-Squared = {:.2f}%".format
(r2*100, adj_r2*100))
671 if i != 'Northeast':
672     print("-----")
673
674 # Plot results
675 for i in range(0,4):
676     actuals[i].loc[:, 'index']=regions[i]
677 actual = pd.concat([actuals[i] for i in range(4)], axis = 0)
678 pred = pd.concat([preds[i] for i in range(4)], axis = 0)
679 df = pd.concat([actual, pred], axis=1).reset_index(drop=True)

```

```

680 col = ["#B14B51", '#D0A99C', '#5D8370', '#6C839B']
681 fig = px.scatter(df, x="actuals", y="preds", color="index",
682                 trendline="ols", height=700,
683                 title="Actual vs Predicted Insurance Costs by
684                       Region,<br>Linear Regression with Principal Component Analysis",
685                 color_discrete_sequence=col, opacity=0.7, facet_col
686                 ='index', facet_col_wrap=2)
687
688 fig.for_each_annotation(lambda a: a.update(text=a.text.split("=")
689 [-1]))
690 fig.add_annotation(text="Adj. R-Squared = {:.1f}%<br>RMSE = {:.0f}"
691 .format(adj_r2_scores[0]*100,rmses[0]),
692                 x=51e3,y=15e3, row=2,col=1, showarrow=False)
693 fig.add_annotation(text="Adj. R-Squared = {:.1f}%<br>RMSE = {:.0f}"
694 .format(adj_r2_scores[1]*100,rmses[1]),
695                 x=51e3,y=15e3, row=2,col=2, showarrow=False)
696 fig.add_annotation(text="Adj. R-Squared = {:.1f}%<br>RMSE = {:.0f}"
697 .format(adj_r2_scores[2]*100,rmses[2]),
698                 x=51e3,y=15e3, row=1,col=1, showarrow=False)
699 fig.add_annotation(text="Adj. R-Squared = {:.1f}%<br>RMSE = {:.0f}"
700 .format(adj_r2_scores[3]*100,rmses[3]),
701                 x=51e3,y=15e3, row=1,col=2, showarrow=False)
702
703 fig.update_traces(hovertemplate="Actual Cost: %{x:$,.2f}<br>
704                       Predicted Cost: %{y:$,.2f}",
705                       marker=dict(size=10, line=dict(width=1,color="#
706                       F7F7F7"))),
707                       selector=dict(mode="markers"), showlegend=False)
708 fig.update_xaxes(title="Actual Cost, $", row=1)
709 fig.update_xaxes(showgrid=True, gridwidth=1, gridcolor='#EAEAEA',
710                 zeroline=True, zerolinewidth=2, zerolinecolor='#5
711                 E5E5E')
712 fig.update_yaxes(title="Predicted Cost, $", col=1)
713 fig.update_yaxes(showgrid=True, gridwidth=1, gridcolor='#E3E3E3',
714                 zeroline=True, zerolinewidth=2, zerolinecolor='#5
715                 E5E5E')
716 fig.update_layout(font_color="#303030", paper_bgcolor="white",
717                 plot_bgcolor="white")
718 fig.show()
719
720 actuals=[]
721 preds=[]
722 rmses=[]
723 r2_scores=[]
724 adj_r2_scores=[]
725 feat_importance=pd.DataFrame()

```

```

716 regions = ins.region.unique()
717 s = StandardScaler()
718 col = ["#B14B51", '#D0A99C', '#5D8370', '#6C839B']
719
720 for i in regions:
721
722     # Разделение датасета по региону
723     ins_df = ins[ins.region==i]
724     X=ins_df.drop(['charges', 'region'], axis=1)
725     y=ins_df.charges
726
727
728
729     pf = PolynomialFeatures(degree=2, include_bias=False)
730     X_pf = pd.DataFrame(data=pf.fit_transform(X), columns=pf.
get_feature_names(X.columns))
731
732     # Распределение на выборки
733     X_train, X_test, y_train, y_test = train_test_split(X_pf, y,
test_size=0.2, random_state=1)
734     X_train = pd.DataFrame(X_train, columns = X_pf.columns)
735     X_test = pd.DataFrame(X_test, columns = X_pf.columns)
736     actuals.append(pd.Series(y_test, name='actuals').reset_index())
737     print("\nRegion: {}\n".format(i))
738     print("Train Shape:{} {} Test Shape:{} {}".format(X_train.shape
, y_train.shape, X_test.shape, y_test.shape))
739
740
741     X_train = pd.DataFrame(data=s.fit_transform(X_train), columns=
X_pf.columns)
742     X_test = pd.DataFrame(data=s.transform(X_test), columns=X_pf.
columns)
743
744
745     grid = {'learning_rate': [0.0001, 0.001, 0.01, 0.1, 0.25, 0.5],
746             'n_estimators': [int(x) for x in np.linspace(start =
200, stop = 1000, num = 5)],
747             'subsample': [0.5, 0.8, 1],
748             'min_samples_split': [2, 5, 10],
749             'min_samples_leaf': [1, 2, 4],
750             'max_depth': [int(x) for x in np.linspace(2, 10, num =
5)],
751             'max_features': [None, 'sqrt']}
752     xgb=GradientBoostingRegressor(random_state=21)
753     xgb_cv=RandomizedSearchCV(estimator=xgb, param_distributions=
grid, scoring='neg_mean_squared_error',
754                               n_iter=100, cv=3, random_state=21,
n_jobs=-1)
755     xgb_cv.fit(X_train, y_train)

```

```

756     y_pred=xgb_cv.predict(X_test)
757     preds.append(pd.Series(y_pred, name='preds').reset_index(drop=
True))
758     rmse=np.sqrt(mean_squared_error(y_test, y_pred)).round(2)
759     r2=r2_score(y_test, y_pred)
760     adj_r2 = 1 - (1-r2)*((len(y_test)-1)/(len(y_test)-X_test.shape
[1]-1))
761     rmses.append(rmse)
762     r2_scores.append(r2)
763     adj_r2_scores.append(adj_r2)
764
765
766     feat_importance["Importance_"+str(i)]=xgb_cv.best_estimator_.
feature_importances_
767
768     print("Test Error (RMSE) = {:,}".format(rmse))
769     print("R-Squared = {:.2f}%, Adjusted R-Squared = {:.2f}%".format
(r2*100, adj_r2*100))
770     if i != 'Northeast':
771         print("-----")
772
773 # Plot results
774 for i in range(0,4):
775     actuals[i].loc[:, 'index']=regions[i]
776 actual = pd.concat([actuals[i] for i in range(4)], axis = 0)
777 pred = pd.concat([preds[i] for i in range(4)], axis = 0)
778 df = pd.concat([actual, pred], axis=1).reset_index(drop=True)
779
780 fig = px.scatter(df, x="actuals", y="preds", color="index",
trendline="lowess", height=700,
781                 title="Реальные vs Предсказанные результаты стоимос
ти соответствующих регионам,<br>Gradient Boosting",
782                 color_discrete_sequence=col, opacity=0.7, facet_col
='index', facet_col_wrap=2)
783
784 fig.for_each_annotation(lambda a: a.update(text=a.text.split("=")
[-1]))
785 fig.add_annotation(text="Adj. R-Squared = {:.1f}%<br>RMSE = {:.0f}"
.format(adj_r2_scores[0]*100,rmses[0]),
786                 x=51e3,y=15e3, row=2,col=1, showarrow=False)
787 fig.add_annotation(text="Adj. R-Squared = {:.1f}%<br>RMSE = {:.0f}"
.format(adj_r2_scores[1]*100,rmses[1]),
788                 x=51e3,y=15e3, row=2,col=2, showarrow=False)
789 fig.add_annotation(text="Adj. R-Squared = {:.1f}%<br>RMSE = {:.0f}"
.format(adj_r2_scores[2]*100,rmses[2]),
790                 x=51e3,y=15e3, row=1,col=1, showarrow=False)
791 fig.add_annotation(text="Adj. R-Squared = {:.1f}%<br>RMSE = {:.0f}"
.format(adj_r2_scores[3]*100,rmses[3]),
792                 x=51e3,y=15e3, row=1,col=2, showarrow=False)

```



```

793
794 fig.update_traces(hovertemplate="Actual Cost: %{x:$.2f}<br>
      Predicted Cost: %{y:$.2f}",
795                   marker=dict(size=10, line=dict(width=1, color="#
      F7F7F7"))),
796                   selector=dict(mode="markers"), showlegend=False)
797 fig.update_xaxes(title="Actual Cost, $", row=1)
798 fig.update_xaxes(showgrid=True, gridwidth=1, gridcolor='#EAEAEA',
799                   zeroline=True, zerolinewidth=2, zerolinecolor='#5
      E5E5E')
800 fig.update_yaxes(title="Predicted Cost, $", col=1)
801 fig.update_yaxes(showgrid=True, gridwidth=1, gridcolor='#E3E3E3',
802                   zeroline=True, zerolinewidth=2, zerolinecolor='#5
      E5E5E')
803 fig.update_layout(font_color="#303030", paper_bgcolor="white",
      plot_bgcolor="white")
804 fig.show()
805
806
807
808 col=sns.color_palette("magma", 20).as_hex()[::-1]
809 feat_importance.set_index(X_train.columns, inplace=True)
810 ft=pd.DataFrame({"Средняя значимость":feat_importance.mean(axis=1)})
811 plot_df=ft.nlargest(20, columns="Средняя значимость").sort_values(by
      ="Средняя значимость", ascending=False)
812 fig = px.bar(plot_df, x="Средняя значимость", y=plot_df.index, text=
      "Средняя значимость", height=700,
813               color=plot_df.index, width=700, opacity=.8,
      color_discrete_sequence=col)
814 fig.update_traces(texttemplate='%{text:.2f}', textposition='outside'
      ,
815                   marker_line=dict(width=1, color='#3F3B3A'),
      showlegend=False,
816                   hovertemplate='Значимость = <b>%{x:.2}</b>')
817 fig.update_layout(title_text='Значимость факторов при оценке стоимос
      ти медицинской страховки',
818                   coloraxis_showscale=False, yaxis_title="",
      font_color="#303030", yaxis_linecolor="#D8D8D8",
819                   xaxis=dict(title="Средняя значимость", showgrid=
      True, showline=True,
820                               linecolor="#9A9A9A", gridcolor="#F5F5F5
      "))
821

```

Листинг 11: Полный код решения Практической работы №6

Таблица 43: Характеристики модели: *data5_fin*

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-0.61121	0.15321	-3.989	7.20e-05	***
log(age)	-0.18727	0.04557	-4.110	4.35e-05	***
h_educ	0.71432	0.09195	7.769	2.31e-14	***
log(dur)	0.16612	0.03971	4.183	3.18e-05	***
I(<i>satisfy</i> ²)	0.25840	0.10182	2.538	0.011333	*
of	0.40250	0.11037	3.647	0.000282	***
gov	0.23492	0.13147	1.787	0.074321	.

Residual standard error: 1.068 on 839 degrees of freedom

Multiple R-squared: 0.1319 Adjusted R-squared: 0.1257

F-statistic 21.25 on 6 and 839 DF p-value: < 2.2e-16

Таблица 44: Описание переменных текущего набора данных

Название переменной	Описание
CLIENTNUM	Уникальный идентификатор клиента
Attrition_Flag	Активность клиента
Customer_Age	Возраст клиента в годах
Gender	М=Мужчина, F=Женщина
Education_Level	Уровень образования клиента
Marital_Status	Состоит ли респондент в браке
Income_Category	Категория годового дохода владельца счета
Credit_Limit	Кредитный лимит по Кредитной карте
Total_Trans_Amt	Общая сумма транзакции (за последние 12 месяцев)
Avg_Open_To_Buy	Открыта кредитная линия на покупку

Таблица 45: Представление начальных данных

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520
5	31	female	25.740	0	no	southeast	3756.62160
6	46	female	33.440	1	no	southeast	8240.58960

Таблица 46: Представление начальных данных							
#	Column	Non-Null	Count	Dtype	smoker	region	charges
0	age	1338	non-null	int64	yes	southwest	16884.92400
1	sex	1338	non-null	object	no	southeast	1725.55230
2	bmi	1338	non-null	float64	no	southeast	4449.46200
3	children	1338	non-null	int64	no	northwest	21984.47061
4	smoker	1338	non-null	object	no	northwest	3866.85520
5	region	1338	non-null	object	no	southeast	3756.62160
6	charges	1338	non-null	float64	no	southeast	8240.58960
dtypes:					float64(2),	int64(2),	object(3)

Таблица 47:	
southeast	364
southwest	325
northwest	325
northeast	324
Name: region, dtype: int64	

Таблица 48: Представление данных				
	Column	Non-Null	Count	Dtype
0	age	1338	non-null	int64
1	sex	1338	non-null	object
2	bmi	1338	non-null	float64
3	children	1338	non-null	int64
4	smoker	1338	non-null	object
5	charges	1338	non-null	float64
6	northeast	1338	non-null	uint8
7	northwest	1338	non-null	uint8
8	southeast	1338	non-null	uint8
9	southwest	1338	non-null	uint8
dtypes: float64 int64(2), object(2), uint8(4)				

Таблица 49: Проверка данных на наличие аномальных значений

	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

Таблица 50: Результат работы функции data.corr()

	age	bmi	children	charges	northeast	northwest	southeast	southwest
age	1.000000	0.109272	0.042469	0.299008	0.002475	-0.000407	-0.011642	0.010016
bmi	0.109272	1.000000	0.012759	0.198341	-0.138156	-0.135996	0.270025	-0.006205
children	0.042469	0.012759	1.000000	0.067998	-0.022808	0.024806	-0.023066	0.021914
charges	0.299008	0.198341	0.067998	1.000000	0.006349	-0.039905	0.073982	-0.043210
northeast	0.002475	-0.138156	-0.022808	0.006349	1.000000	-0.320177	-0.345561	-0.320177
northwest	-0.000407	-0.135996	0.024806	-0.039905	-0.320177	1.000000	-0.346265	-0.320829
southeast	-0.011642	0.270025	-0.023066	0.073982	-0.345561	-0.346265	1.000000	-0.346265
southwest	0.010016	-0.006205	0.021914	-0.043210	-0.320177	-0.320829	-0.346265	1.000000

Значимость факторов при оценке стоимости медицинской страховки

