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 _____ Ю.И.Худак
«09» февраля 2022 г.

СОГЛАСОВАНО
Руководитель практики от кафедры:
«09» февраля 2022 г. _____ (Петрусевич Д.А.)
(подпись) (фамилия и инициалы)

Задание получил:
«09»февраля 2022 г. _____ (Сиразетдинов Р.Д.)
(подпись) (фамилия и инициалы)

ИНСТРУКТАЖ ПРОВЕДЕН:

Вид мероприятия	ФИО ответственного, подпись, дата	ФИО студента, подпись, дата
Охрана труда	Петрусевич Д.А. _____ «09» февраля 2022 г.	Сиразетдинов Р.Д. _____ «09» февраля 2022 г.
Техника безопасности	Петрусевич Д.А. _____ «09» февраля 2022 г.	Сиразетдинов Р.Д. _____ «09» февраля 2022 г.
Пожарная безопасность	Петрусевич Д.А. _____ «09» февраля 2022 г.	Сиразетдинов Р.Д. _____ «09» февраля 2022 г.
Правила внутреннего распорядка	Петрусевич Д.А. _____ «09» февраля 2022 г.	Сиразетдинов Р.Д. _____ «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

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 умееют 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$Examination) # 16.48936
5
6       # дисперсия(сумма квадратов отклоний всех значений от средне
    го) столбцов
7       #Catholic, Agriculture, Examination
8       var(data$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_Arg = lm(Catholic~Agriculture, data)
3           model_Arg
4           summary(model_Arg)
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 умееют 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 реггрессором и описывемой переменной есть, но эта связь недостаточно сильная- только 2 звездочки, значит вероятность того, что посчитанное в модели значение y обусловленно моделью и не сильно отличается 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 построенны две модели: *model_ Agr* - модель завсимости переменной *Catholic* от регрессора *Agriculture* и *model_ Ex* - модель завсимотси 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* отрицальный, т.е. люди, которые получали образование выше начальной школы в меньшей степени относились к католикам. Относились ли они к православной церкви или другим комуннам сказать нельзя, т.к. для этого нужно проводисть 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	Продвежение сотрудников по службе

2.1 Решение

2.1.1 Проверка отсутствия линейной зависимости между регрессорами

Для начала проверим зависимость между регрессорами, если некоторые будут зависить друг от другу, 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.

```
model_auxiliary_2 = lm(raises ~ advance, attitude)
summary(model_auxiliary_2)

model_auxiliary_3 = lm(critical ~ advance, data)
summary(model_auxiliary_3)

```

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		

```
model_all = lm(rating ~ raises + critical + advance, data)
summary(model_all)

no_critical_model = lm(rating ~ raises + advance, data)
summary(no_critical_model)

no_raises_model = lm(rating ~ critical + advance, data)
summary(no_raises_model)

```

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

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

Листинг 7: Построение линейных зависимостей.

Поскольку построенных моделей достаточно много, здесь лишь остановимся на некоторых комментариях, которые дают общее представление о логике посроения последующий 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 будет 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 показатль 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 *raiting* нельзя вполной 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 Стьедента, которое может быть вычислено с помощью функций языка R.

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

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

```
3
```

Листинг 8: Расчитаем *t-критерий Стьедента.*

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 Стьедента: 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(raises))*
   - значение коэффициента: 55.004
   - стандартная ошибка: 13.119
   - t-критерий Стьедента: 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 Стьедента: 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(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(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
- доверитильный 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.

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

```

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(raises)) + I(log(advance))). Но доверительный интервал получился достаточно большим

$$\approx$$

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 обработака 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 обратки данных с помощью языка программирования *R* см. в разделе Приложение.

3.1.2 Построение линейной регрессии зароботной 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*.

```
 ds_2 = select(ds, salary, age, sex, h_educ, city_status, dur,
wed1, wed2, wed3, satisfy, of, gov)

 # модель_1 - 'model_def'
 model_def = lm(data = ds_2, salary ~ age + sex + h_educ + city_
status +
                  dur + wed1 + wed2 + wed3 + satisfy + of + gov)
 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

```
      vif(model_def)
      age          sex       h_educ  city_status          dur
wed1
1.252570     1.138695     1.071094     1.026265     1.080268
2.425066
     wed2         wed3      satisfy           of          gov
2.025865     1.920659     1.035515     1.037693     1.031553

```

Листинг 11: Проверка регрессоров на линейную зависимоть.

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 моделе.

3.1.3 Преобразование регрессоров построенной модели

Построим некоторые другие модели, эксперементируя с регрессорами, и выберем из них наилучшую. Поскольку построенных моделей очень много, здесь приведем характеристики лучших моделей, характеристики прочих можно увидель 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 коэффицинты перед регрессороми 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	1.49e-14	***
h_educ	0.63077	0.08806	7.163 1.73e-12 ***	1.73e-12	***
city_status	0.46811	0.07522	6.223 7.68e-10 ***	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 стистически зарабатывают меньше. В каждой из моделей регрессоры *of* - наличие официального трудоустройста, *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:

```
data5_1 = subset(ds_2, sex = 1)
data5_2 = subset(data5_1, wed3 = 1)
data5_fin = subset(data5_2, h_educ = 0)

model_5_1 = lm(data = data5_fin, salary ~ log(age) + city_status +
                 log(dur) + I(satisfy^2) + of + gov)
summary(model_5_1)

```

Листинг 12: Подмножество 1.

Вывод: по полученным данным(см. таблицу 18) можно предположить, что заработок респондентов огранниченых на «подмножестве 1» сильно зависит от возраста - с увееличением 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:

```
data5_1 = subset(ds_2, sex = 1)
data5_2 = subset(data5_1, city_status = 1)
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($satisfy^2$)	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
5   model_5_2 = lm(data = data5_fin, salary ~ log(age) + h_educ +
6                  log(dur) + I(satisfy^2) + of + gov)
7   summary(model_5_2)
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($satisfy^2$)	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 огранниченых на «подмножестве

2» так же сильно зависит от возраста - с увеелиечением 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 даннных 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	M=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 пуктов вы можете найти в разделе Приложение в полном коде решенея. Здесь же мы приведем полученные результаты(см. рисунки 1, 3).

Вернемся к задаче классификации. Построим классификатор методом опорных векторов.

```
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)
svm_clf = svm.SVC()

# Поиск наилучших параметров для классфикации методом опорных вектор
    ов
svm_param = {'C': [0.1, 1, 10, 100],
             'gamma': ['scale', 'auto'],
             'kernel': ['rbf', 'poly', 'sigmoid'],
             'probability':[True],
             'class_weight':['balanced', 'none']}
```

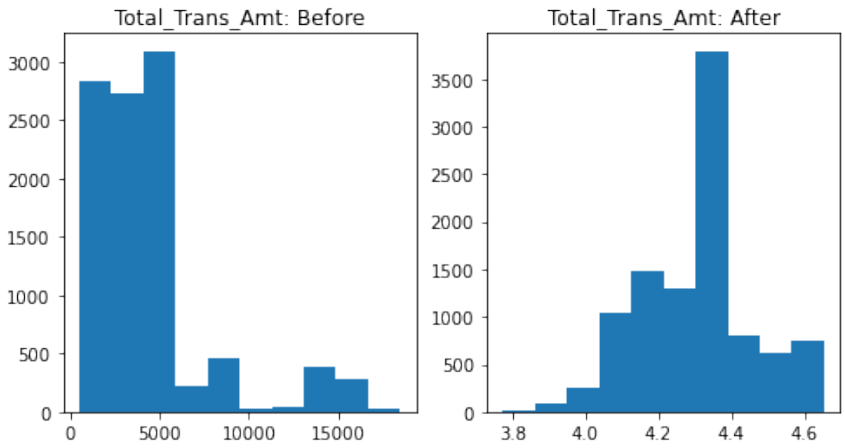


Рис. 1: Результат ноормализации признака Total_Trans_Amt.

```
svm_grid = GridSearchCV(svm_clf, svm_param, n_jobs=-1, cv=5, verbose=1)
svm_grid.fit(X_train, y_train)
print("Best param: ", svm_grid.best_params_)
svm_pred = svm_grid.predict(X_test)
svm_pred_prob = svm_grid.predict_proba(X_test)[:,1]
svm_pred_df = pd.DataFrame({"pred":svm_pred, "prob":svm_pred_prob, "actual":y_test})
# mannually adjust pred based on prob
thresh = np.quantile(svm_pred_prob, (1-y_train.mean()))
svm_pred_df['pred'] = (svm_pred_df['prob']>thresh).astype(int)
svm_pred_df

X, y = std_bank_df.iloc[:,1:], std_bank_df.iloc[:,0]


```

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

Полуичли, что лучшие параметры для данного классификатора

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

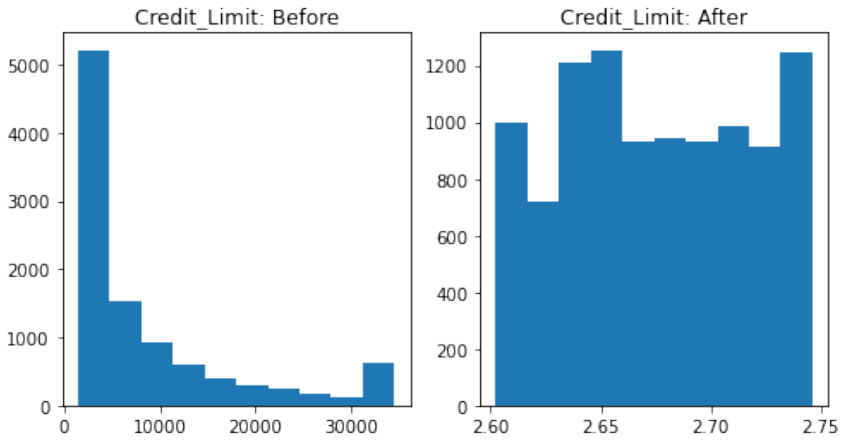


Рис. 2: Результат ноормализации признака Credit_Limit.

```
estimator = svm.SVC(C=10,
                    class_weight='balanced',
                    gamma='scale',
                    kernel='rbf',
                    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 постороения линейной регрессии, но не стоит расчитывать на полноценную точность. Теперь построим классификатор типа Случайный Лес.

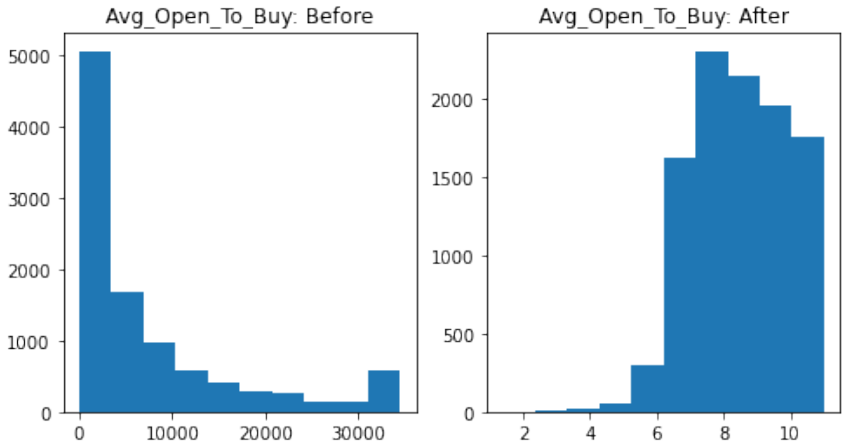


Рис. 3: Результат ноормализации признака Avg_Open_To_Buy.

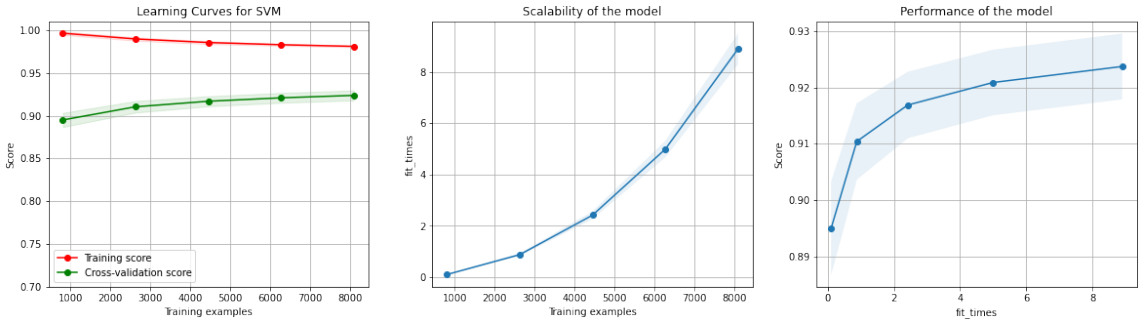


Рис. 4: Результат классификатора SVM.

```
rf = RandomForestClassifier()
n_estimators = [100, 200, 300]
max_features = [0.5, 0.25, 'log2', 'sqrt']
max_depth = [100, 200, 'none']
min_samples_split = [2, 5, 10]
min_samples_leaf = [1, 2, 4]
bootstrap = [True, False]
class_weight = ['balanced', 'none']
rf_param = {'n_estimators': n_estimators,
            'max_features': max_features,
            'max_depth': max_depth,
            'min_samples_split': min_samples_split,
            'min_samples_leaf': min_samples_leaf,
            'bootstrap': bootstrap,
            'class_weight': class_weight}
rf_grid = GridSearchCV(rf, rf_param, n_jobs=-1, cv=5, verbose=1)
```

```
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.

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

Листинг 17: Реализация классификатора Случайный Лес.

Получим следующие результаты:

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

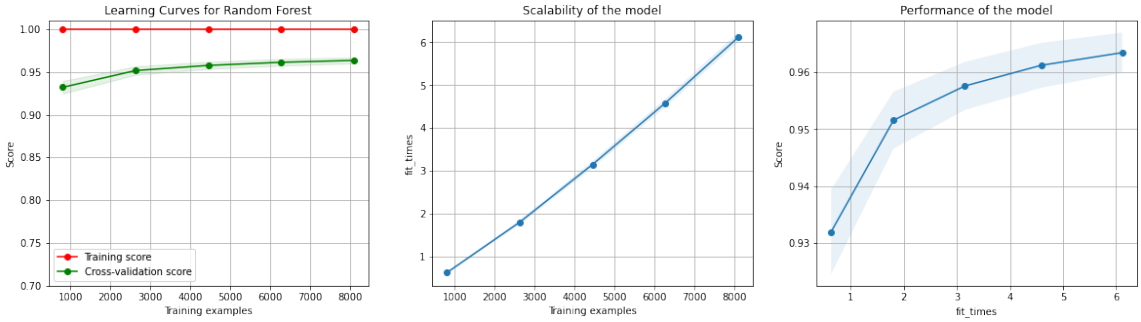


Рис. 5: Результат классификатора Random Forest.

- recall: 0.7494549627079747

Оказалось, что точность второго классификатора лучше(см. рисунки 4,5). Из всех названным 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 решаюших деревьев, восполняя недостаток точности большим количеством, в этой задаче классификации показала себя с хорошей стороны, в то время как, разбиение объектов гиперплоскостями не боло 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

Name: region, dtype: int64

- **smoker** - является ли бенефициар курильщиком
- **region** - район, в котором проживает бенефициар
- **charges** - индивудуальные расходы, оплачиваемые медецинским 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 числвые признаки **age**, **children**, **bmi**, **charges**.

Обработаем категориальные признаки.

```
region = pd.get_dummies(data['region'])
data = pd.concat((data, region), axis=1)
data = data.loc[:, data.columns.isin(['age', 'sex', 'bmi', 'children
    ' , '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 представленны 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 предетеся их отдельно обрабатывать.

5.1.4 Определение и обработка анамальных значений и выбрасов

- 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 медецинской 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 желательно должен быть в рамках от 18.5 до 24.9, хотя из таблицы видно, что даже у четверти клиентов нет таких показателей. Среднее значение значение этого показателя 30.7, что превышает норму более чем на 5 пунктов. В целом можно сказть, что большинство клиентов имееют ожирение. Давайте посмотрим на это более детально.

• Сравним рост индекса массы тела с ростом возраста и с ростом индивидуальных расходов по медецинской страховке.

```
    plt.figure(figsize=(12,10), dpi= 80)
    sns.heatmap(data.corr(), xticklabels=data.corr().columns,
   yticklabels=data.corr().columns, cmap='RdYlGn', center=0, annot=
   True)

    plt.xticks(fontsize=12)
    plt.yticks(fontsize=12)
    plt.show()

```

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

• В итоге, см. рис 8 и 7 и таблицу 26, никакой простой зависимотси роста индекса тела от возрасти или от медецинских расходов выявить не удалось, это значит, что нет повышения или понижения возраста или медецинских расходов связванных индксом массы тела. Т.е. другими словами, практически каждый клиет имеет индекс массы тела выше нормы, но медецинские 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

```
data['sex'] = np.where(data['sex'] == 'female' , 0, data['sex'])
data['sex'] = np.where(data['sex'] == 'male' , 1, data['sex'])

data['smoker'] = np.where(data['smoker'] == 'no' , 0, data['smoker'
   ])
```

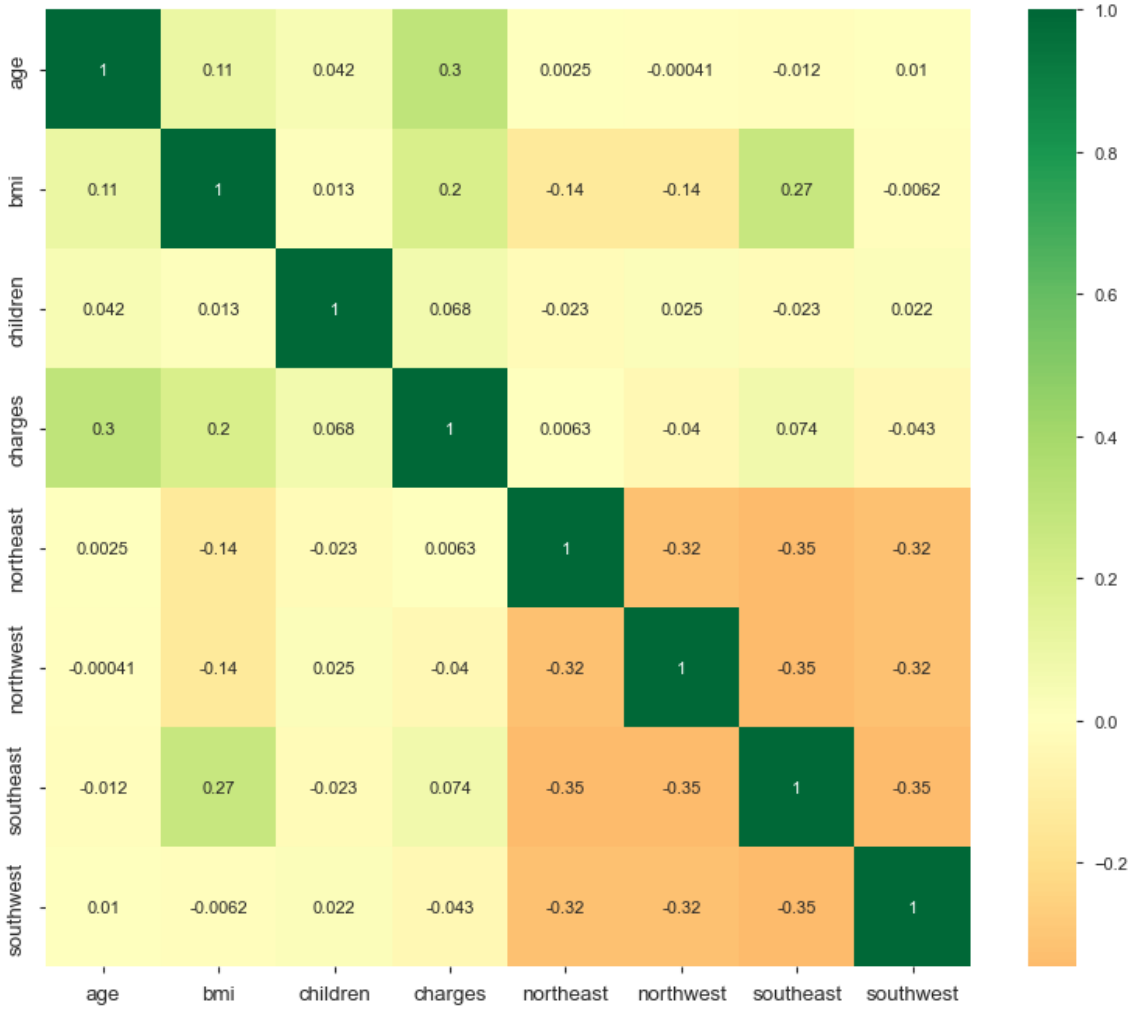


Рис. 6: Корреляция признаков датасета.

```
data['smoker'] = np.where(data['smoker'] == 'yes' , 1, data['smoker'
    ])


from sklearn import preprocessing
import seaborn as sns # Для графиков

# Инициализируем стандартизатор
s_scaler = preprocessing.StandardScaler()

# Копируем исходный датасет
data_s = s_scaler.fit_transform(data)
```

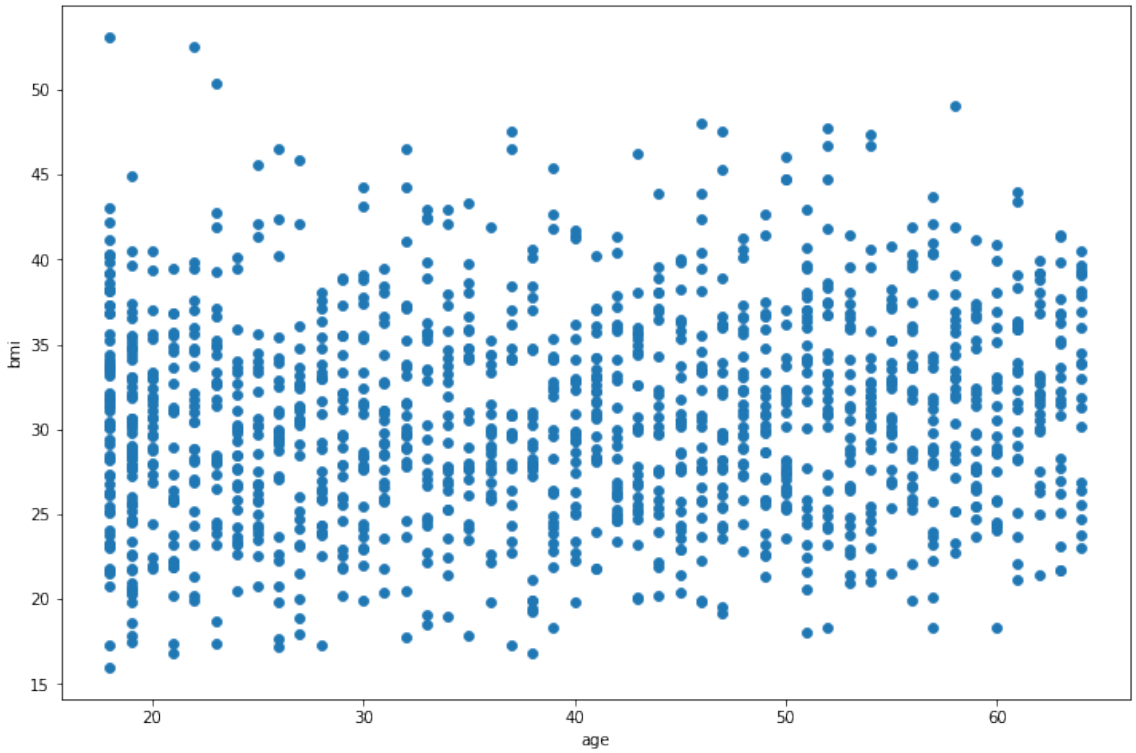


Рис. 7: График зависимости индекса массы тела от медецинких расходов.

```
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. 7 - 7

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

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

```

Листинг 21: Ислючение признака charges из данных.

5.1.7 Разбиение данных на выборки

Поскольу параметр **test_size** имеет значение 0.3, а все объекты разделяются тренировачную и обучаемую выборки, то количество объектов в тренировачной 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 иструментов библиотеки **pandas**.

```
    data.corr()

```

Листинг 22: Проверка корреляции признаков между собой.

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

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

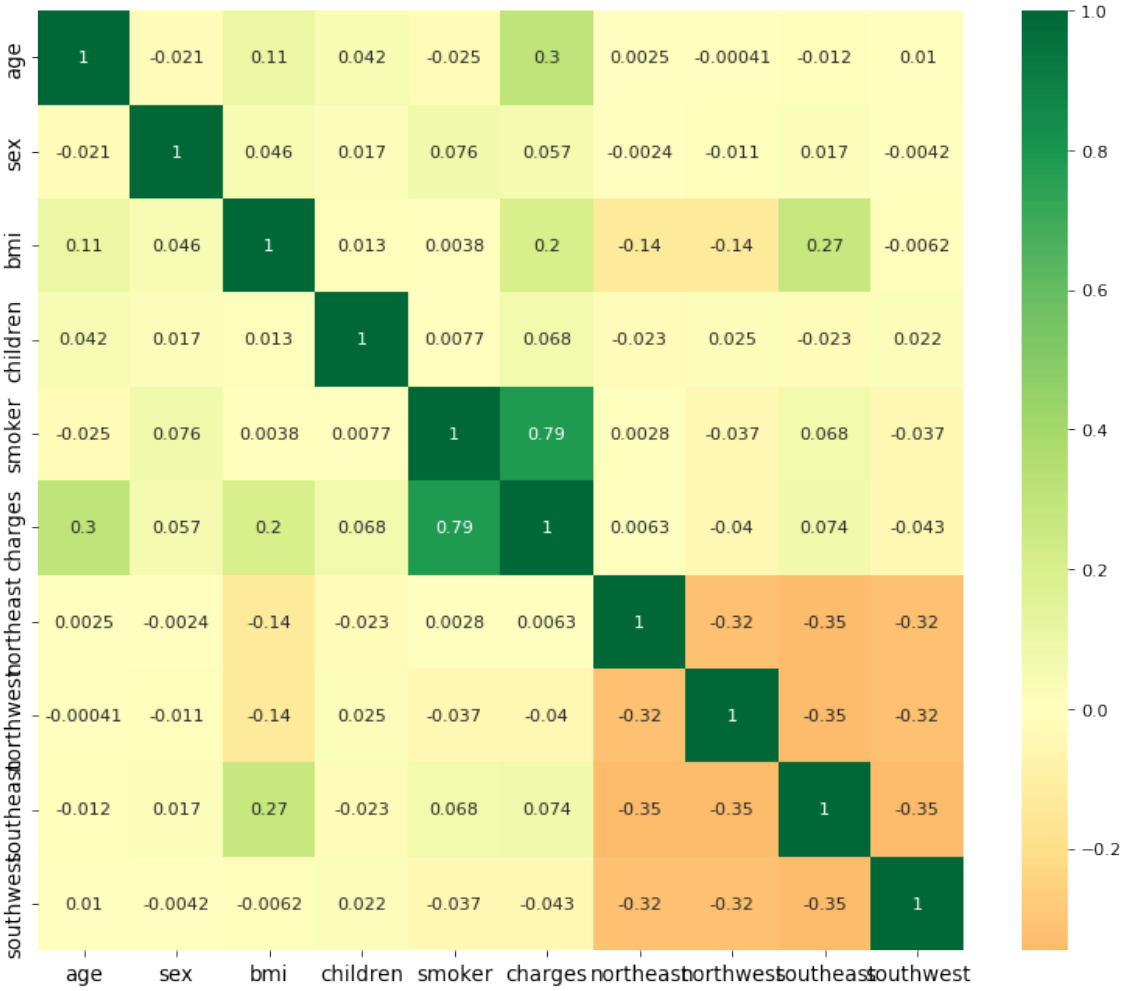


Рис. 8: Корреляция признаков датасета.

Теперь разделим данные на обучающую и тестовую выборки.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(train, target,
    test_size = 0.25, random_state = 42)

N_train, _ = X_train.shape
N_test,  _ = X_test.shape
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 предстваленых в таблице 26, исключая столбец с выбранным целевым признаком, воспользуемся методом главных компонент.

```
pca = PCA()
pca.fit(X_train)
X_pca = pca.transform(X_train)
for i, component in enumerate(pca.components_):
    print("{} component: {}% of initial variance".format(i + 1,
          round(100 * pca.explained_variance_ratio_[i], 2)))
    print(" + ".join("%.3f x %s" % (value, name)
                     for value, name in zip(component,train.columns)
   ))
    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, медициские страховки самые дорогие. Что на первый взгляд очень странно. Проведя дополнительные исследования в интернете, обнаживается, 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.

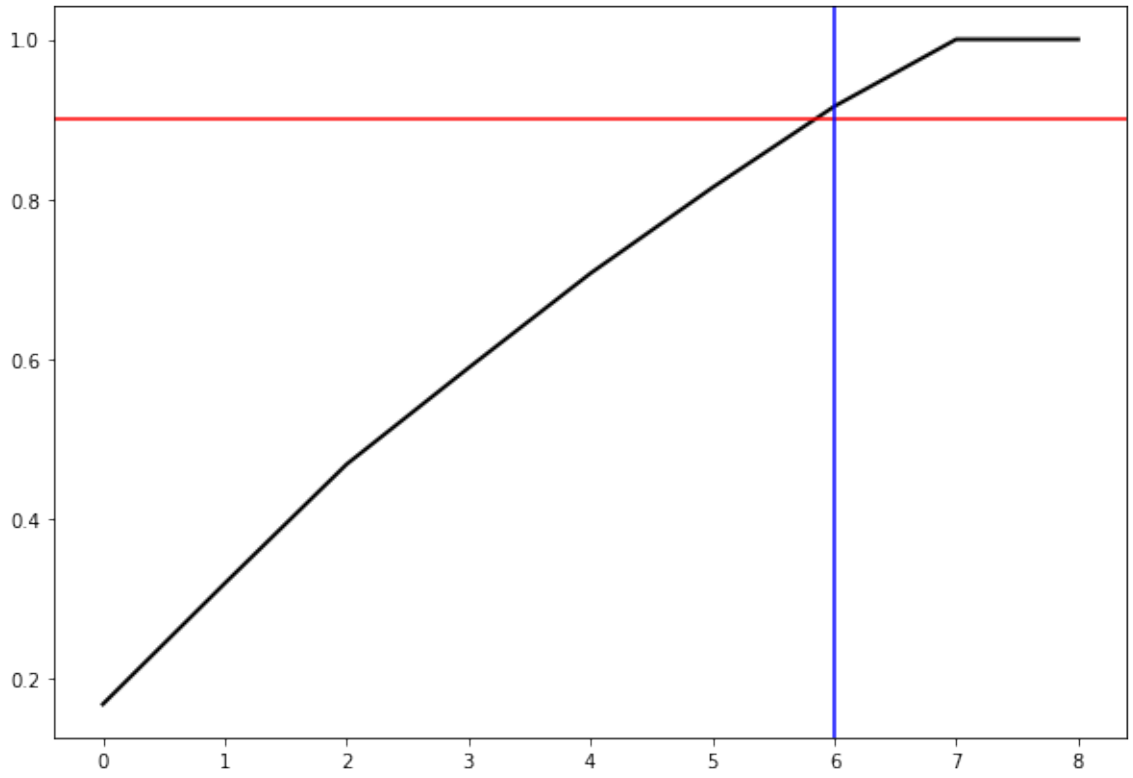


Рис. 9: Визуализация зависимости качества описываемой дисперсии от количество описывающих признаков.

6 Практическая работа №6

В практической работе №5 мы провели начальную обработку данных и с помощью метода главных компонент продемонстрировали сильную линейную зависимости 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 компонеты, оказываеющие наибольшее влияние на стоимость.

6.1 Решение

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

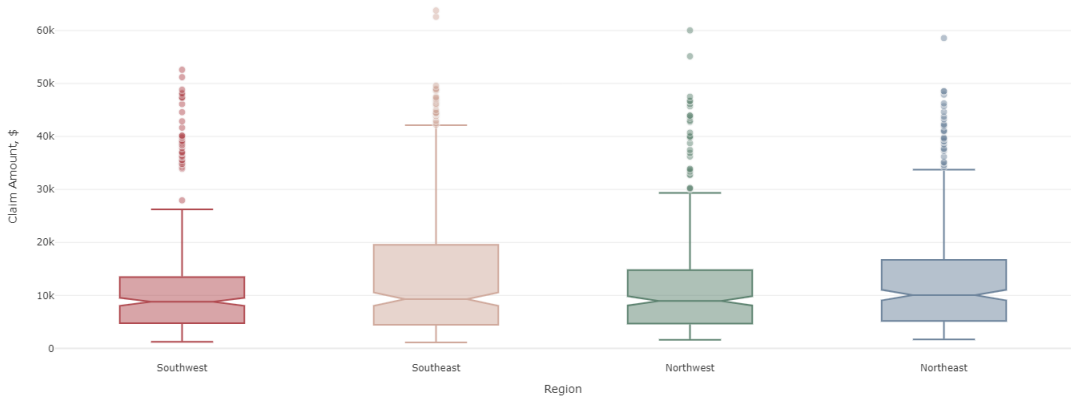


Рис. 10: Визуализация стоимости медицинской страховки к каждом регионе.

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 медицискую страховку.

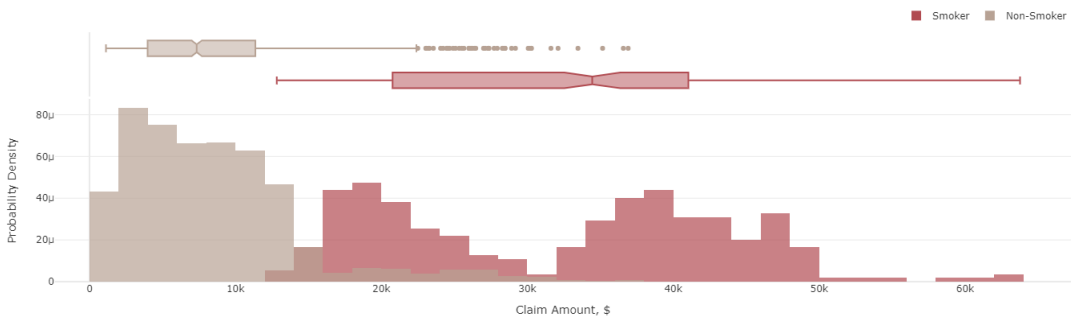


Рис. 11: Визуализация распределения стоимости медицинской страховки среди курящих и некурящих.

Здесь же (см. рис 11) можно заметить весьма важную деталь. Стоимость медициской 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 жещин 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 медецинской страховки для мужчины выше, чем у женщины. Кроме того, стоит отбратить внимание на стоимость медицинской страховки у курящих и нет. Первые плятят больше в **4-5** раз!

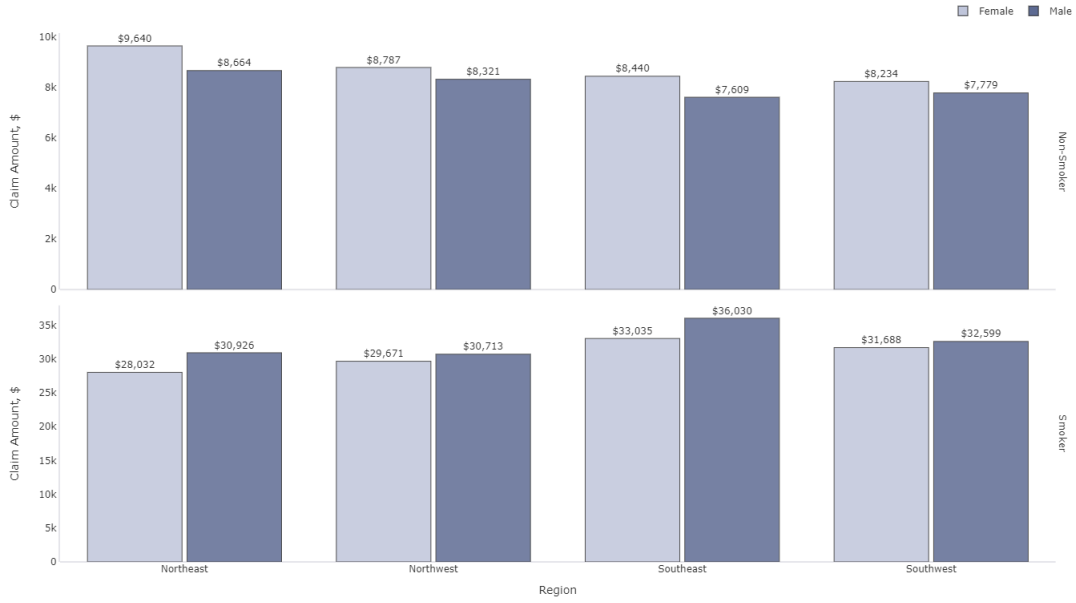


Рис. 12: Визуализация распределения стоимости медицинской страховки.

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

• Как выяснилось в практической работе №5 есть существенная зависимось 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 ленейных регрессий.

Для начала построим зависимость от всех параметров.

```
y_train=data.pop('charges')
X_train = data.copy()

import statsmodels.api as sm
X_train_sm = sm.add_constant(X_train)

all= sm.OLS(y_train, X_train_sm).fit()

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 имееют отрицательные значения коэффициентов в построенной модели, что, на самамо деле, противоречит тому, что было получено ранее через анализ зависимости описываемой переменно и регрессорами, отвечающими за регион.

Распределим полученные результаты по регионам см. рис. 13.

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

```
    actuals[i].loc[:,'index']=regions[i]
actual = pd.concat([actuals[i] for i in range(4)], axis = 0)
pred = pd.concat([preds[i] for i in range(4)], axis = 0)
df = pd.concat([actual, pred], axis=1).reset_index(drop=True)
col = ["#B14B51", '#D0A99C', '#5D8370','#6C839B']
fig = px.scatter(df, x="actuals", y="preds", color="index",
    trendline="ols", height=700,
                 title="Actual vs Predicted Insurance Costs by
    Region,<br>Linear Regression with Principal Component Analysis",
                 color_discrete_sequence=col, opacity=0.7, facet_col
    ='index', facet_col_wrap=2)

fig.for_each_annotation(lambda a: a.update(text=a.text.split("=")
    [-1]))
fig.add_annotation(text="Adj. R-Squared = {:.1f}%<br>RMSE = {:,.0f}"
    .format(adj_r2_scores[0]*100,rmses[0]),
                   x=51e3,y=15e3, row=2,col=1, showarrow=False)
fig.add_annotation(text="Adj. R-Squared = {:.1f}%<br>RMSE = {:,.0f}"
    .format(adj_r2_scores[1]*100,rmses[1]),
                   x=51e3,y=15e3, row=2,col=2, showarrow=False)
fig.add_annotation(text="Adj. R-Squared = {:.1f}%<br>RMSE = {:,.0f}"
    .format(adj_r2_scores[2]*100,rmses[2]),
                   x=51e3,y=15e3, row=1,col=1, showarrow=False)
fig.add_annotation(text="Adj. R-Squared = {:.1f}%<br>RMSE = {:,.0f}"
    .format(adj_r2_scores[3]*100,rmses[3]),
                   x=51e3,y=15e3, row=1,col=2, showarrow=False)

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 линейнх 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	bmi^2, age^2	0.752	0.750
age_lg	log(age,2)	0.664	0.662
bmi_lg	log(bmi,2)	0.751	0.750
bmi_age_lg	log({age,bmi},2)	0.665	0.663
lm_15	log(age,2),bmi^2	0.660	0.659
lm_16	**log(bmi,2),age^2**	**0.754**	**0.753**
bmi_smoke	bmi+smoke_yes	0.658	0.657
age_smoke	age+smoke_yes	0.721	0.721
smk_sq	smoke_yes^2	0.751	0.749
lm_12	age + bmi	0.117	0.116

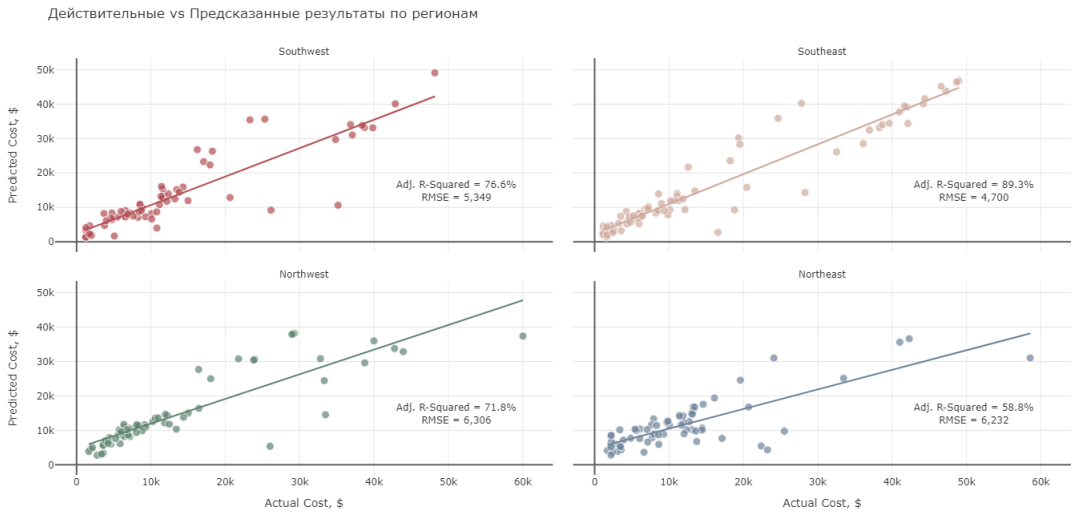


Рис. 13: Распределение результатов модели **all** по регионам.

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

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

```
    X_train = pd.DataFrame(data=s.fit_transform(X_train), columns=
    X_pf.columns)
X_test = pd.DataFrame(data=s.transform(X_test), columns=X_pf.columns
    )


grid = {'learning_rate': [0.0001, 0.001, 0.01, 0.1, 0.25, 0.5],
        'n_estimators': [int(x) for x in np.linspace(start = 200,
    stop = 1000, num = 5)],
        'subsample': [0.5, 0.8, 1],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4],
        'max_depth': [int(x) for x in np.linspace(2, 10, num = 5)],
        'max_features': [None, 'sqrt']}
xgb=GradientBoostingRegressor(random_state=21)
xgb_cv=RandomizedSearchCV(estimator=xgb, param_distributions=grid,
    scoring='neg_mean_squared_error',
                          n_iter=100, cv=3, random_state=21, n_jobs
    =-1)
xgb_cv.fit(X_train, y_train)
y_pred=xgb_cv.predict(X_test)
```

```
preds.append(pd.Series(y_pred, name='preds').reset_index(drop=True))
rmse=np.sqrt(mean_squared_error(y_test, y_pred)).round(2)
r2=r2_score(y_test, y_pred)
adj_r2 = 1 - (1-r2)*(len(y_test)-1)/(len(y_test)-X_test.shape[1]-1)
rmses.append(rmse)
r2_scores.append(r2)
adj_r2_scores.append(adj_r2)

```

Листинг 27: Постоение модели на основе градиентного бустинга.

Посмотрим на результаты.

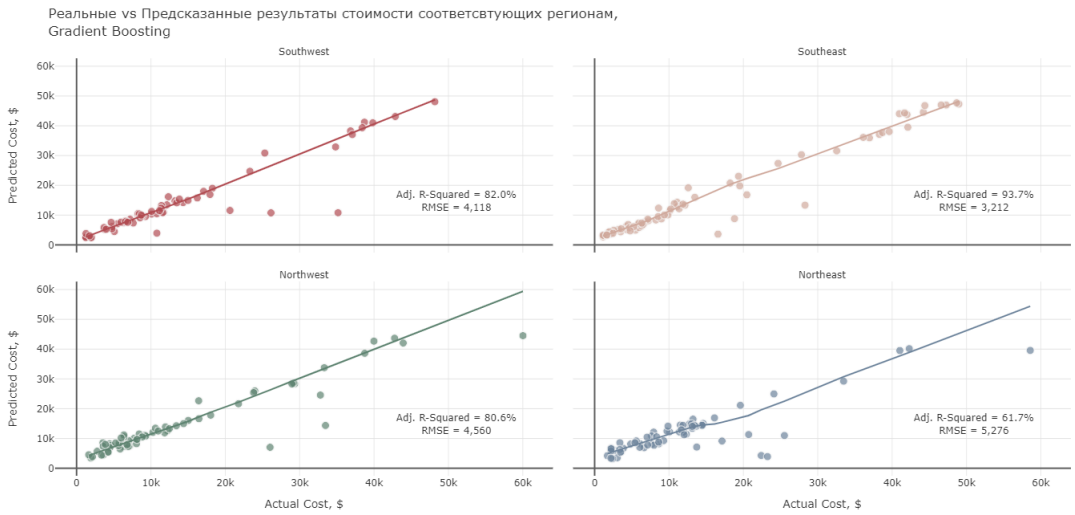


Рис. 14: Визуализация результатов постоенной модели *xgb_cv*.

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 счиать, что построенная с помощью градиентного спуска модель удачная.

Теперь мы приведем усредненые значения значимости отдельных параметров и их композиций при оценке стоимости медицинской страховки(см. рис 15, которые были использованы в модели $\mathbf{xgb}_{cv}$.

Полный код решения см. в разделе Приложение).

Вывод. В представленной практической работе №6 мы выяснили, какие из параметров оказывают наиболее зна-

чимое воздействие на рост стоимости медициской страховки. Как оказалось, наибольшая средняя цена медицинской страховки представлена в северо-восточном регионе, а наибольшая стоимость страховки в юго-восточном. Курящие люди, имеющие ожирение плятят за страховки значительно больше остальных, и разница стоимости страховки для курящих и нет весьма значительная. Вторым и третьим по значимости оказались признаки возрасти и индекс массы тела, которые так же достаточно сильно оказывают влияние на стоимость медицинской страховки. Чем старше или чем больше у него лишнего веса, тем дороже ему обходится медицинская страховка. Однако признак, определяющий количество детей наоборот не вносит существенной разницы в стоимость, точно также как и гендер клиента.

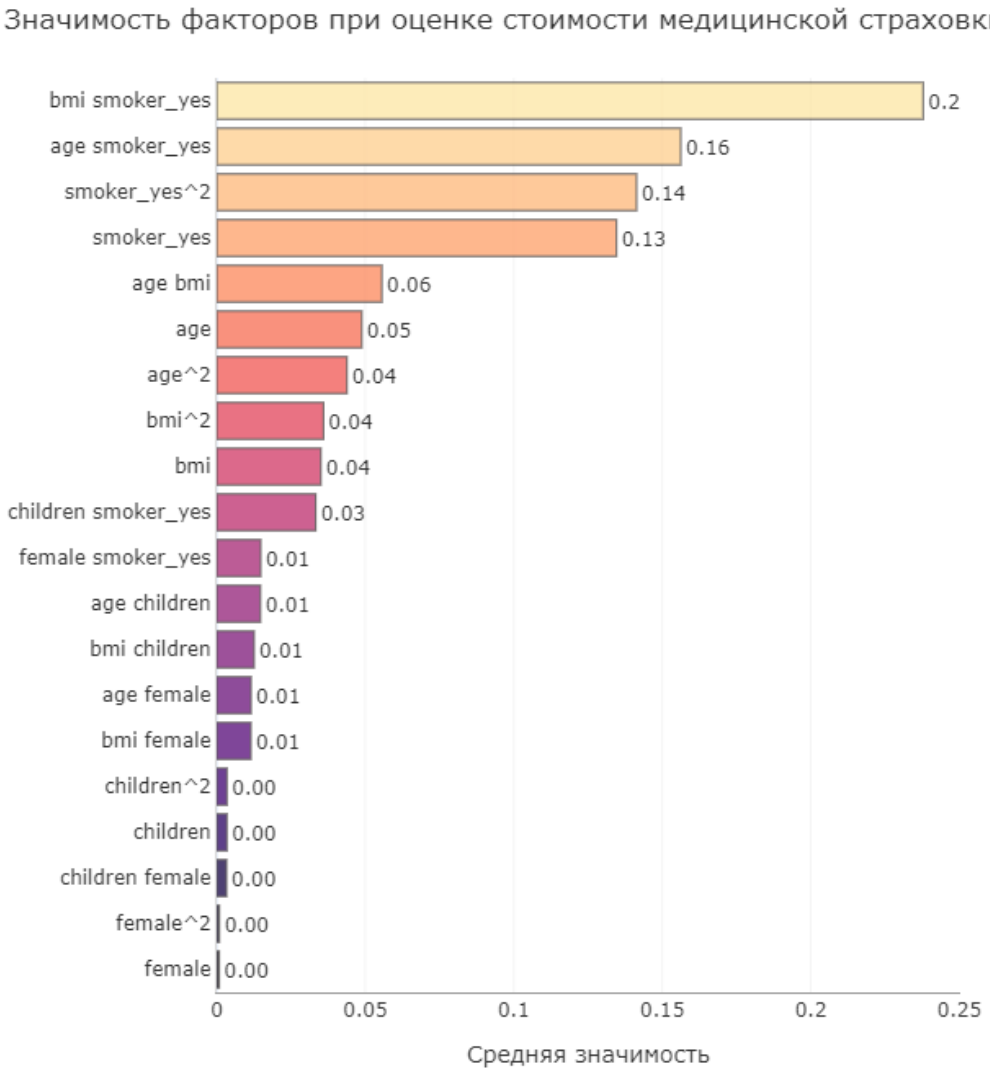


Рис. 15: Визуализация распределения стоимости медицинской страховки.

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.**

7 Приложение

```
output model_Agr
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.8313    13.8966   0.276   0.7840
Agriculture   0.7365     0.2508   2.937   0.0052 **
---
Residual standard error: 38.63 on 45 degrees of freedom
Multiple R-squared:  0.1609,  Adjusted R-squared:  0.1422
F-statistic: 8.627 on 1 and 45 DF,  p-value: 0.005204

################################################
output model_Ex
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 ***
---

Residual standard error: 34.56 on 45 degrees of freedom
Multiple R-squared:  0.328, Adjusted R-squared:  0.3131
F-statistic: 21.97 on 1 and 45 DF,  p-value: 2.588e-05

```

Листинг 1: Практическая работа 1: Результаты построенных моделей

```
      library("lmtest")
      library("GGally")
      data = swiss # insertion base data
      summary(data)

      #-------------------------------------------------------
      # среднее значение столбцов Catholic, Agriculture,
Examination
      mean(data$Catholic)    # 41.14383
      mean(data$Agriculture) # 50.65957
      mean(data$Examination) # 16.48936

      # дисперсия(сумма квадратов отклоний всех значений от средне
го) столбцов
      #Catholic, Agriculture, Examination
      var(data$Catholic)    # 1739.295
      var(data$Agriculture) # 515.7994
      var(data$Examination) # 63.64662

      # СКО(среднеквадратическое отклонение столбцов Catholic,
Agriculture, Examination
```

```
sd(data$Catholic)    # 41.70485
sd(data$Agriculture) # 22.71122
sd(data$Examination) # 7.977883
#-------------------------------------------------------------


# model_Agr: y- Catholic ~ x1- Agriculture
model_Arg = lm(Catholic~Agriculture, data)
model_Arg
summary(model_Arg)

model_Ex = lm(Catholic~Examination, data)
model_Ex
summary(model_Ex)

```

Листинг 2: Практическая работа 1: Полный код программы

```
model_auxiliary_2 = lm(raises ~ advance, attitude)
summary(model_auxiliary_2)

model_auxiliary_3 = lm(critical ~ advance, data)
summary(model_auxiliary_3)

```

Листинг 3: Построение линейных зависимостей

```
model_all = lm(rating ~ raises + critical + advance, data)
summary(model_all)

no_critical_model = lm(rating ~ raises + advance, data)
summary(no_critical_model)

no_raises_model = lm(rating ~ critical + advance, data)
summary(no_raises_model)

```

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 $\sim$ 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 $\sim$ 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 ∼ 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 ∼ 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 ∼ 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 ∼ 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($critical^2$)	-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

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($critical^2$)	-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

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($critical^2$)	-9.771e-04	1.412e-03	-0.692	0.495141	
I($advance^2$)	-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($advance^2$)	-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($raises^2$)	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($advance^2$)	-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
2       library("lmtest")
3       library("GGally")
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 +
    advance, data)
31      summary(model_all_log_1)
32
33      model_all_log_2 = lm(rating ~ raises + I(log(critical)) +
    advance, data)
34      summary(model_all_log_2)
35
36      model_all_log_3 = lm(rating ~ raises + I(log(advance)) +
    critical, data)
37      summary(model_all_log_3)
38
39      model_all_log_4 = lm(rating ~ raises + I(log(advance)) + I(
    log(critical)), data)
40      summary(model_all_log_4)
41
42      no_critical_model_log_1 = lm(rating ~ raises + I(log(advance
    )), data)
43      summary(no_critical_model_log_1)
44
```

```
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}- перрессоры
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

```
> vif(model_def)
      age          sex       h\_educ city\_status          dur
     wed1
1.252570     1.138695     1.071094     1.026265     1.080268
2.425066
     wed2         wed3      satisfy           of          gov
2.025865     1.920659     1.035515     1.037693     1.031553

```

Листинг 6: Проверка регрессоров на линейную зависимоть

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	***

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	***

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	1.49e-14	***
h_educ	0.63077	0.08806	7.163 1.73e-12 ***	1.73e-12	***
city_status	0.46811	0.07522	6.223 7.68e-10 ***	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 ***	< 2e-16	***
city_status	0.46811	0.07522	6.223 7.68e-10 ***	< 2e-16	***
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 30: Характеристики модели: *model_7*

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	***
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	*

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(age^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(dur^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(age^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(dur^2)	0.009851	0.003183	3.095	0.00198	**
wed3	0.012800	0.034397	0.372	0.70982	***
I($satisfy^2$)	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($satisfy^2$)	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	*

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(dur^2)	0.009629	0.003676	2.620	0.008855	**
I($satisfy^2$)	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	***

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,
      pj6.2, pj1.1.1, pj11.1, pj23)
13
14    # исключение строк со значением NA
15    ds = na.omit(ds)
16
17    ################################################
18    # Семейное положение
19    ds["wed"] = ds$p_marst
20    ds["wed"] = lapply(ds["wed"], as.character)
21    ds$wed1 = 0
22
23    ds$wed1[which(ds$wed == '2')] <- 1
24    ds$wed1[which(ds$wed == '6')] <- 1
25    ds$wed1 = as.numeric(ds$wed1)
26
27    ds["wed2"] = ds$p_marst
28    ds$wed2 = 0
29
30    ds$wed2[which(ds$wed == '4')] <- 1
31    ds$wed2[which(ds$wed == '5')] <- 1
32    ds$wed2 = as.numeric(ds$wed2)
33
34    ds["wed3"] = ds$p_marst
35    ds$wed3 = 0
36
37    ds$wed3[which(ds$wed == '1')] <- 1
38    # ds$wed3[which(ds$wed == '3')] <- 1
39    ds$wed3 = as.numeric(ds$wed3)
40
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 зависимотсти переменной ‘salary‘
    от
141   # всех введеных регрессоров
142
143   ds_2 = select(ds, salary, age, sex, h_educ, city_status, dur,
```

```
wed1, wed2, wed3, satisfy, of, gov)

 # модель_1 - ‘model_def‘
 model_def = lm(data = ds_2, salary ~ age + sex + h_educ + city_status +
           dur + wed1 + wed2 + wed3 + satisfy + of + gov)
 summary(model_def)
 vif(model_def)

 # модель_2 - ‘model_2‘
 model_2 = lm(data = ds_2, salary ~ age + sex + h_educ + city_status +
           dur + wed2 + wed3 + satisfy + of + gov)
 summary(model_2)
 vif(model_2)

 # модель_3 - ‘model_3‘
 model_3 = lm(data = ds_2, salary ~ age + sex + h_educ + city_status +
         dur + wed3 + satisfy + of + gov)
 summary(model_3)
 vif(model_3)

 # модель_4 - ‘model_4‘
 model_4 = lm(data = ds_2, salary ~ log(age) + sex + h_educ + city_status +
         dur + wed3 + satisfy + of + gov)
 summary(model_4)
 vif(model_4)

 # модель_5 - ‘model_5‘
 model_5 = lm(data = ds_2, salary ~ log(age) + sex + h_educ + city_status +
         log(dur) + wed3 + satisfy + of + gov)
 summary(model_5)
 vif(model_5)

 # модель_6 - ‘model_6‘
 model_6 = lm(data = ds_2, salary ~ log(age) + sex + h_educ + city_status +
         log(dur) + wed3 + satisfy + of + gov)
 summary(model_6)
 vif(model_6)

 # модель_7 - ‘model_7‘
 model_7 = lm(data = ds_2, salary ~ I(age^2) + sex + h_educ + city_status +
         log(dur) + wed3 + satisfy + of + gov)
 summary(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 +
    city_status +
189             I(dur^2) + wed3 + satisfy + of + gov)
190     summary(model_8)
191     vif(model_8)
192
193     # модель_9 - ‘model_9‘
194     model_9 = lm(data = ds_2, salary ~ I(age^2) + sex + h_educ +
    city_status +
195             I(dur^2) + wed3 + I(satisfy^2) + of + gov)
196     summary(model_9)
197     vif(model_9)
198
199     # модель_10 - ‘model_10‘
200     model_10 = lm(data = ds_2, salary ~ I(age^2) + sex + h_educ +
    city_status +
201             log(dur) + wed3 + I(satisfy^2) + of + gov)
202     summary(model_10)
203     vif(model_10)
204
205     # модель_11 - ‘model_11‘
206     model_11 = lm(data = ds_2, salary ~ I(age^2) + sex + h_educ +
    city_status +
207             log(dur) + wed3 + I(satisfy^2) + of + gov)
208     summary(model_11)
209     vif(model_11)
210
211     # модель_12 - ‘model_12‘
212     model_12 = lm(data = ds_2, salary ~ I(age^0.1) + sex + h_educ +
    city_status +
213              dur + I(satisfy^2) + of + gov)
214     summary(model_12)
215     vif(model_12)
216
217     # модель_13 - ‘model_13‘
218     model_13 = lm(data = ds_2, salary ~ I(age^0.5) + sex + h_educ +
    city_status +
219              dur + I(satisfy^2) + of + gov)
220     summary(model_13)
221     vif(model_13)
222
223     # модель_14 - ‘model_14‘
224     model_14 = lm(data = ds_2, salary ~ log(age) + sex + h_educ +
    city_status +
225              log(dur) + I(satisfy^2) + of + gov)
226     summary(model_14)
```

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

```
    vif(model_21)

    # модель_22 - ‘model_22‘
    model_22 = lm(data = ds_2, salary ~ age + sex + h_educ + city_
   status +
             log(dur) + satisfy + of + gov)
    summary(model_22)
    vif(model_22)

    ################################################


    # Не вступавшие в брак мужчины, без высшего образования;

    data5_1 = subset(ds_2, sex = 1)
    data5_2 = subset(data5_1, wed3 = 1)
    data5_fin = subset(data5_2, h_educ = 0)

    model_5_1 = lm(data = data5_fin, salary ~ log(age) + city_status
    +
              log(dur) + I(satisfy^2) + of + gov)
    summary(model_5_1)


    ################################################

    # Городские жители, мужчины состоящие в браке

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

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


```

Листинг 7: Практическая работа №3: Полный код программы

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, GridSearchCV,
   RandomizedSearchCV, learning_curve, ShuffleSplit, cross_val_score
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	*

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($satisfy^2$)	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

```
from sklearn.metrics import f1_score, roc_auc_score, roc_curve
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.pipeline import make_pipeline
# from pandas_profiling import ProfileReport
from sklearn.impute import KNNImputer

from sklearn.ensemble import RandomForestClassifier

import warnings
warnings.filterwarnings("ignore")

from sklearn import svm

from scipy.special import boxcox1p
from scipy.stats import boxcox_normmax
pd.set_option('display.max_columns', 50)
pd.set_option('display.max_rows', 100)

bank_df = pd.read_csv("C:\\Users\\averu\\Documents\\git_local\\
    programming-practice\\IDA-practice-4\\BankChurners.csv")
bank_df

bank_df['Marital_Status'] = np.where(bank_df['Marital_Status'] == '
    Married' , False, bank_df['Marital_Status'])
bank_df['Marital_Status'] = np.where(bank_df['Marital_Status'] == '
    Single' , True, bank_df['Marital_Status'])
bank_df['Marital_Status'] = np.where(bank_df['Marital_Status'] == '
    Divorced' , True, bank_df['Marital_Status'])
bank_df['Marital_Status'] = np.where(bank_df['Marital_Status'] == '
    Unknown' , True, bank_df['Marital_Status'])

# Выбрасываем некоторые признаки
drop_cols = list(bank_df.iloc[:,[0,-1,-2, -3, 6]].columns)
bank_df.drop(columns=drop_cols, inplace=True)

# Трансформация категориального признака к целочисленному типу `int`
bank_df['Attrition_Flag'] = (bank_df['Attrition_Flag']=='Attrited
    Customer').astype(int)

bank_df.describe()

# Contacts_Count_12_mon
by_contact_df = bank_df.groupby('Contacts_Count_12_mon')['
    Attrition_Flag'].mean()
by_contact_df.plot(kind='bar', ylabel='Attrited Ratio')

# Months_Inactive_12_mon
by_inactive_df = bank_df.groupby('Months_Inactive_12_mon')['
    Attrition_Flag'].mean()
```

```
by_inactive_df.plot(kind='bar', ylabel='Attrited Ratio')

# Education level
by_edu_df = bank_df.groupby('Education_Level')['Attrition_Flag'].
    mean()
by_edu_df.plot(kind='bar', ylabel='Attrited Ratio')

# Income_Category
by_income_df = bank_df.groupby('Income_Category')['Attrition_Flag'].
    mean()
by_income_df.plot(kind='bar', ylabel='Attrited Ratio')

# Gender
by_gender_df = bank_df.groupby('Gender')['Attrition_Flag'].mean()
by_gender_df.plot(kind='bar', ylabel='Attrited Ratio')

# Card_Category
by_card_df = bank_df.groupby('Card_Category')['Attrition_Flag'].mean
    ()
by_card_df.plot(kind='bar', ylabel='Attrited Ratio')

# Credit Limit
print("Avg attrition of customers with minimun credit limit: ",
      bank_df[bank_df['Credit_Limit']==bank_df['Credit_Limit'].min()
    ]['Attrition_Flag'].mean())
print("Avg attrition of customers with more credit limit: ",
      bank_df[bank_df['Credit_Limit']!=bank_df['Credit_Limit'].min()
    ]['Attrition_Flag'].mean())

# Total_Revolving_Bal
print("Avg attrition of customers with 0 revolving balance: ",
      bank_df[bank_df['Total_Revolving_Bal']==0]['Attrition_Flag'].
    mean())
print("Avg attrition of customers with more revolving balance: ",
      bank_df[bank_df['Total_Revolving_Bal']!=0]['Attrition_Flag'].
    mean())


tmp_bank_df = bank_df.copy()

le_ls = []
for col in ['Education_Level', 'Income_Category']:
    le = LabelEncoder()
    tmp_bank_df[col] = le.fit_transform(tmp_bank_df[col])
    keys = le.classes_
    values = le.transform(le.classes_)
    dictionary = dict(zip(keys, values))
    le_ls.append(le)
    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']
```

```
trans_bank_df = bank_df.copy()
for col in skewed_col:
    plt.figure(figsize=(8, 4))
    plt.subplot(1, 2, 1)
    plt.hist(bank_df[col])
    plt.title(f"{col}: Before")
    trans_bank_df[col] = boxcox1p(bank_df[col], boxcox_normmax(bank_df[col] + 1))
    plt.subplot(1, 2, 2)
    plt.hist(trans_bank_df[col])
    plt.title(f"{col}: After")

# Стандартизация признаков
std = StandardScaler()
std_df = pd.DataFrame(std.fit_transform(trans_bank_df.iloc[:,1:]),
                      index = trans_bank_df.index,
                      columns = trans_bank_df.columns[1:])
std_bank_df = pd.concat([trans_bank_df.iloc[:,0], std_df], axis=1)

# Reference: https://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html
def plot_learning_curve(
    estimator,
    title,
    X,
    y,
    axes=None,
    ylim=None,
    cv=None,
    n_jobs=None,
    train_sizes=np.linspace(0.1, 1.0, 5),
):
    """
    Generate 3 plots: the test and training learning curve, the training
    samples vs fit times curve, the fit times vs score curve.

    Parameters
    ----------
    estimator : estimator instance
        An estimator instance implementing `fit` and `predict` methods which
        will be cloned for each validation.

    title : str
        Title for the chart.

    X : array-like of shape (n_samples, n_features)
        Training vector, where ``n_samples`` is the number of
```

```
    samples and
177         ``n_features`` is the number of features.
178
179     y : array-like of shape (n_samples) or (n_samples, n_features)
180         Target relative to ``X`` for classification or regression;
181         None for unsupervised learning.
182
183     axes : array-like of shape (3,), default=None
184         Axes to use for plotting the curves.
185
186     ylim : tuple of shape (2,), default=None
187         Defines minimum and maximum y-values plotted, e.g. (ymin,
    ymax).
188
189     cv : int, cross-validation generator or an iterable, default=
    None
190         Determines the cross-validation splitting strategy.
191         Possible inputs for cv are:
192
193           - None, to use the default 5-fold cross-validation,
194           - integer, to specify the number of folds.
195           - :term:`CV splitter`,
196           - An iterable yielding (train, test) splits as arrays of
    indices.
197
198         For integer/None inputs, if ``y`` is binary or multiclass,
199         :class:`StratifiedKFold` used. If the estimator is not a
    classifier
200         or if ``y`` is neither binary nor multiclass, :class:`KFold`
     is used.
201
202         Refer :ref:`User Guide <cross_validation>` for the various
203         cross-validators that can be used here.
204
205     n_jobs : int or None, default=None
206         Number of jobs to run in parallel.
207         ``None`` means 1 unless in a :obj:`joblib.parallel_backend`
    context.
208         ``-1`` means using all processors. See :term:`Glossary <
    n_jobs>`
209         for more details.
210
211     train_sizes : array-like of shape (n_ticks,)
212         Relative or absolute numbers of training examples that will
    be used to
213         generate the learning curve. If the ``dtype`` is float, it
    is regarded
214         as a fraction of the maximum size of the training set (that
    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 классфикации методом опорных вектор
    ов
303 svm_param = {'C': [0.1, 1, 10, 100],
```

```
              'gamma': ['scale', 'auto'],
              'kernel': ['rbf', 'poly', 'sigmoid'],
              'probability':[True],
              'class_weight':['balanced', 'none']}
svm_grid = GridSearchCV(svm_clf, svm_param, n_jobs=-1, cv=5, verbose
    =1)
svm_grid.fit(X_train, y_train)
print("Best param: ", svm_grid.best_params_)
svm_pred = svm_grid.predict(X_test)
svm_pred_prob = svm_grid.predict_proba(X_test)[:,1]
svm_pred_df = pd.DataFrame({"pred":svm_pred, "prob":svm_pred_prob, "
    actual":y_test})
# mannually adjust pred based on prob
thresh = np.quantile(svm_pred_prob, (1-y_train.mean()))
svm_pred_df['pred'] = (svm_pred_df['prob']>thresh).astype(int)
svm_pred_df

X, y = std_bank_df.iloc[:,1:], std_bank_df.iloc[:,0]


estimator = svm.SVC(C=10,
                    class_weight='balanced',
                    gamma='scale',
                    kernel='rbf',
                    probability=True)
estimator.fit(X_train, y_train)

plt.show()

print( "accuracy:"+str(np.average(cross_val_score(estimator, X_test,
    y_test, scoring=  'accuracy'))))
print(       "f1:"+str(np.average(cross_val_score(estimator, X_test,
    y_test, scoring=  'f1'))))
print("precision:"+str(np.average(cross_val_score(estimator, X_test,
    y_test, scoring=  'precision'))))
print(   "recall:"+str(np.average(cross_val_score(estimator, X_test,
    y_test, scoring=  'recall'))))


# Random Forest

rf = RandomForestClassifier()
n_estimators = [100, 200, 300]
max_features = [0.5, 0.25, 'log2', 'sqrt']
max_depth = [100, 200, 'none']
min_samples_split = [2, 5, 10]
min_samples_leaf = [1, 2, 4]
bootstrap = [True, False]
class_weight = ['balanced', 'none']
```

```
rf_param = {'n_estimators': n_estimators,
            'max_features': max_features,
            'max_depth': max_depth,
            'min_samples_split': min_samples_split,
            'min_samples_leaf': min_samples_leaf,
            'bootstrap': bootstrap,
            'class_weight': class_weight}
rf_grid = GridSearchCV(rf, rf_param, n_jobs=-1, cv=5, verbose=1)
rf_grid.fit(X_train, y_train)



estimator = RandomForestClassifier(bootstrap=False,
                                   class_weight='balanced',
                                   max_depth=200,
                                   max_features=0.25,
                                   min_samples_leaf=1,
                                   min_samples_split=2,
                                   n_estimators=300)
estimator.fit(X_train, y_train)

plt.show()

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

Листинг 8: Полный код решения Практической работы №4

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

data = pd.read_csv('C:\\Users\\averu\\Documents\\git_local\\
   programming-practice\\IDA-practice-5\\insurance.csv')

data.shape
data.head(7)
data.info()

data['sex'].value_counts()
data['smoker'].value_counts()
data['region'].value_counts()
```

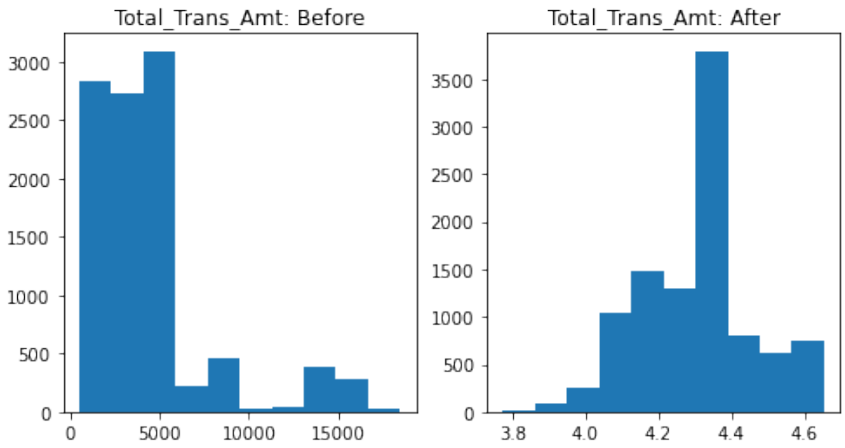


Рис. 16: Результат ноормализации признака Total_Trans_Amt.

```
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()
```

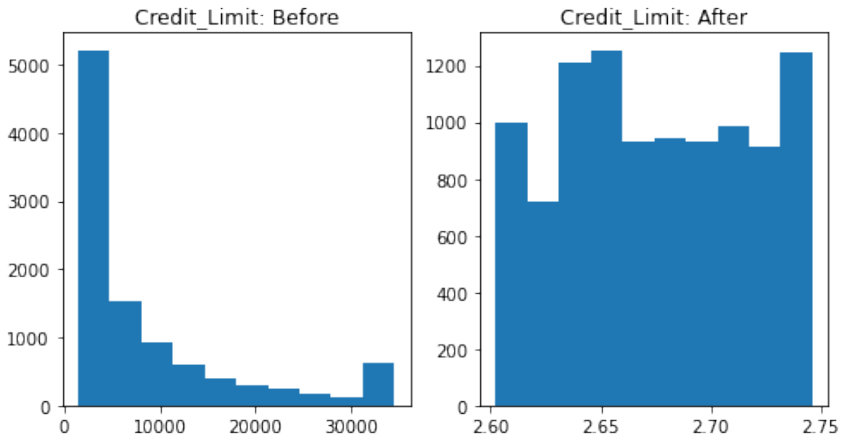


Рис. 17: Результат ноормализации признака Credit_Limit.

```
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'])
```

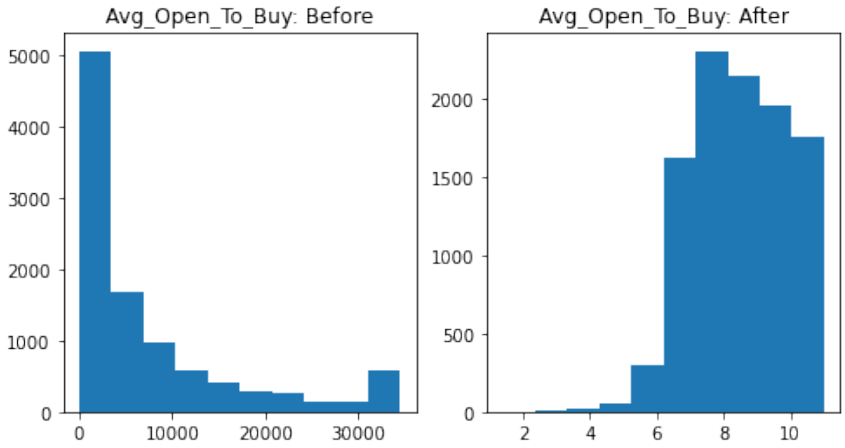


Рис. 18: Результат ноормализации признака Avg_Open_To_Buy.

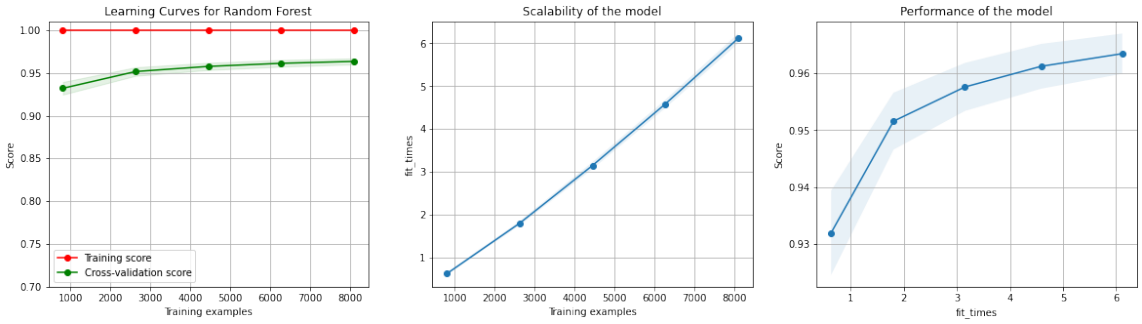


Рис. 19: Результат классификатора Random Forest.

```
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')
```

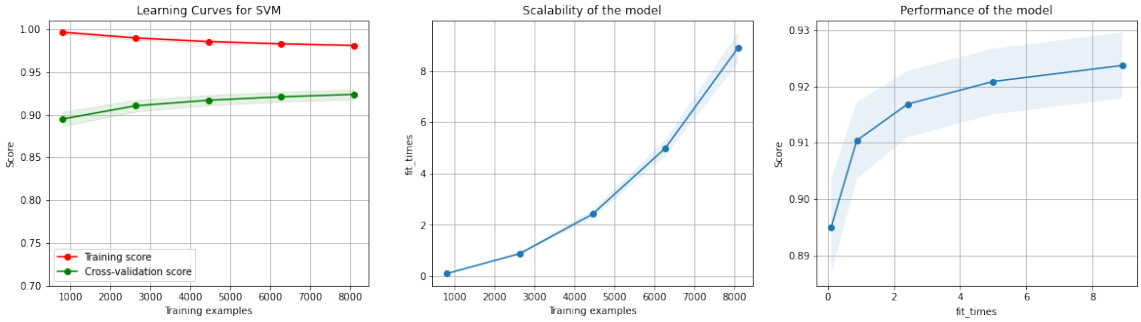


Рис. 20: Результат классификатора SVM.

```
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,
        yticklabels=data.corr().columns, cmap='RdYlGn', center=0, annot=
        True)
103 plt.xticks(fontsize=12)
104 plt.yticks(fontsize=12)
105 plt.show()
106
107 sns.heatmap(data.corr(),annot=True)
108
109
110 from sklearn.model_selection import train_test_split
111 X_train, X_test, y_train, y_test = train_test_split(train, target,
        test_size = 0.25, random_state = 42)
112
113 N_train, _ = X_train.shape
114 N_test,  _ = X_test.shape
115 print (N_train, N_test)
```

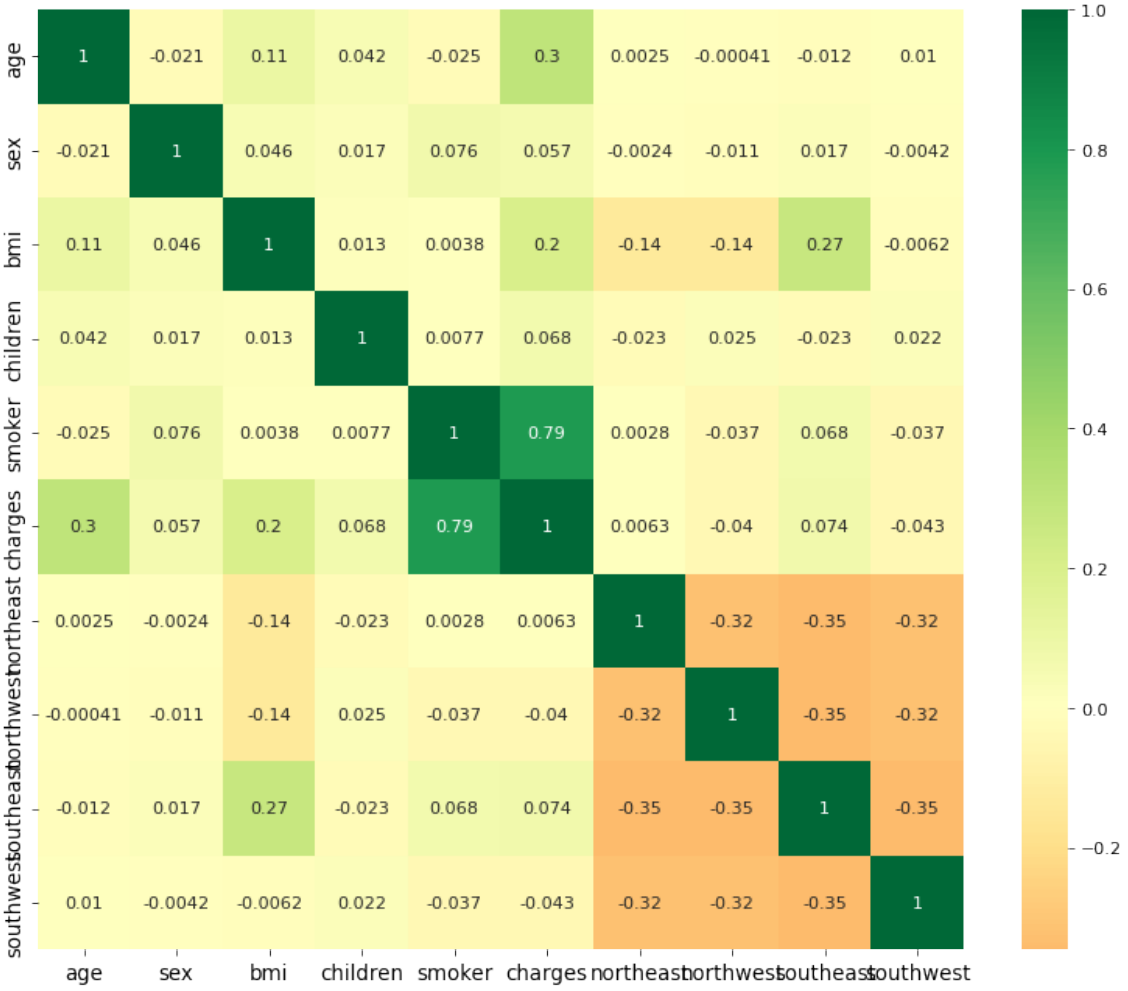


Рис. 21: Корреляция признаков датасета.

```
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     print('\n')
```

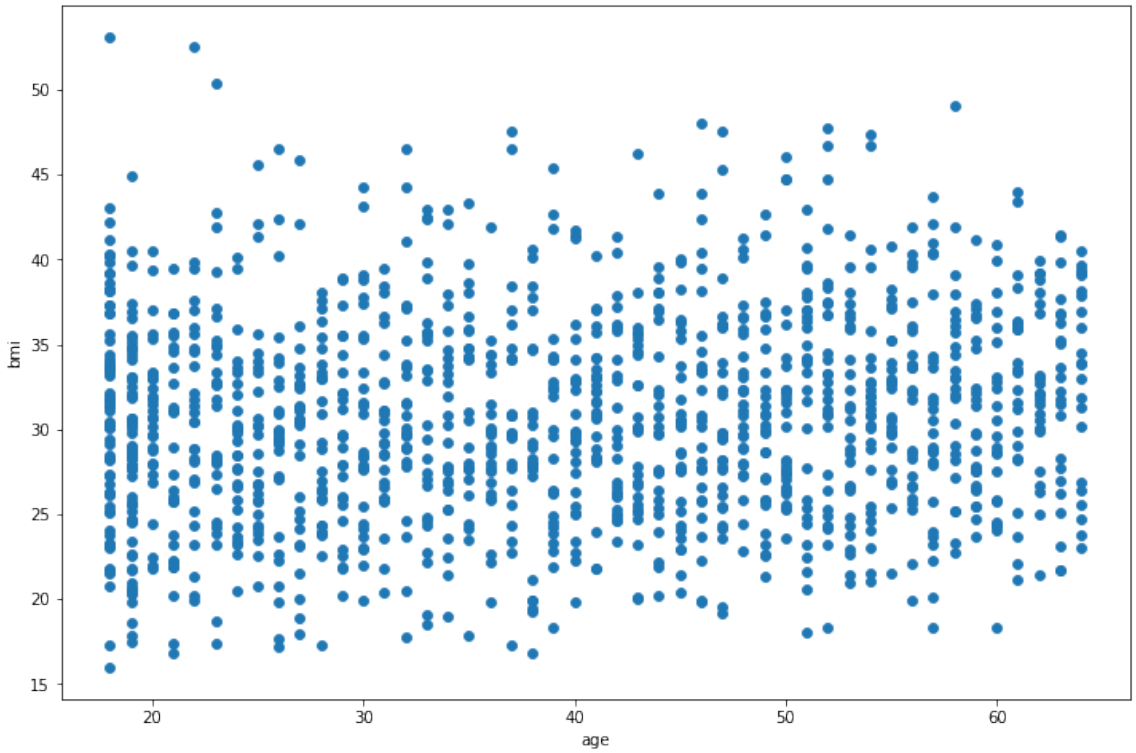


Рис. 22: График зависимости индекса массы тела от медецинких расходов.

```
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
   ', '#6C839B'])
8 fig.update_traces(marker=dict(size=9, opacity=0.5, line=dict(width
   =1,color="#F7F7F7")), showlegend=False)
9 fig.update_layout(font_color="#303030", xaxis_title='Region',
   yaxis_title='Claim Amount, $',
```

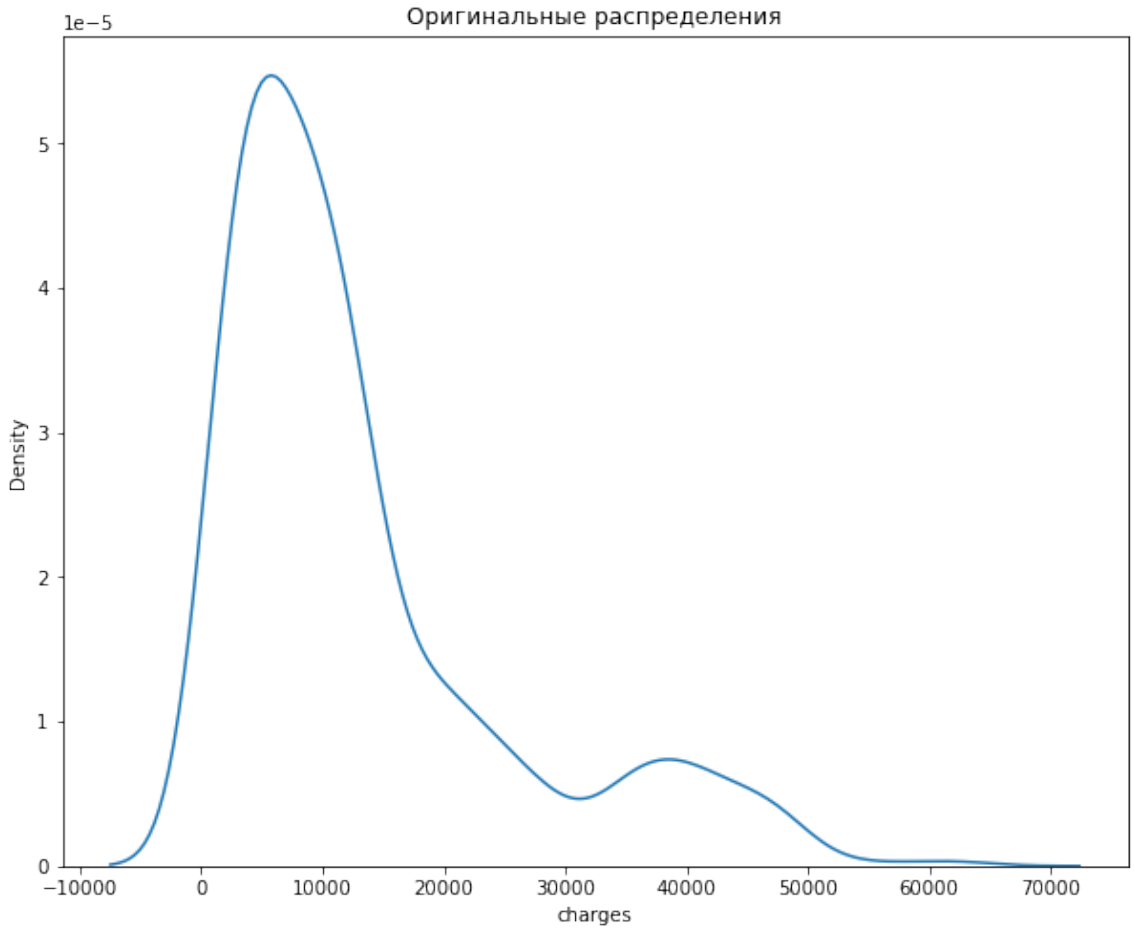


Рис. 23: Признак **charges** до стандартизации.

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

```

Листинг 10: Распределение стоимости медицинской страховки в зависимости от региона

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
from plotly.offline import plot, iplot, init_notebook_mode
```

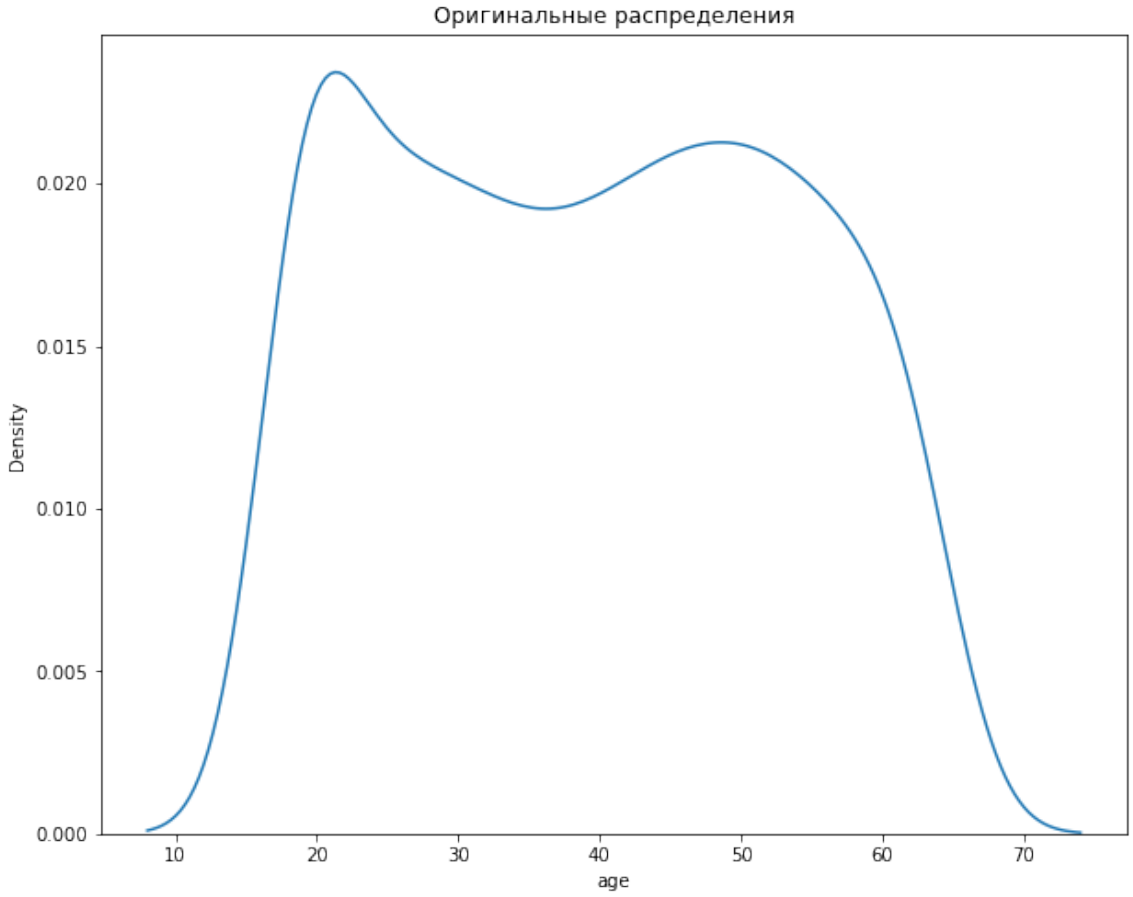


Рис. 24: Признак **age** до стандартизации.

```
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.decomposition import PCA
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import train_test_split,
    RandomizedSearchCV
from sklearn.metrics import r2_score, mean_squared_error
import warnings
warnings.filterwarnings("ignore")
import math

# Выгрузка датасета
ins = pd.read_csv('C:\\Users\\averu\\Documents\\git_local\\
    programming-practice\\IDA\\IDA-practice-5\\insurance.csv')
print("There are {:,} observations and {} columns in the data set.".
    format(ins.shape[0], ins.shape[1]))
```

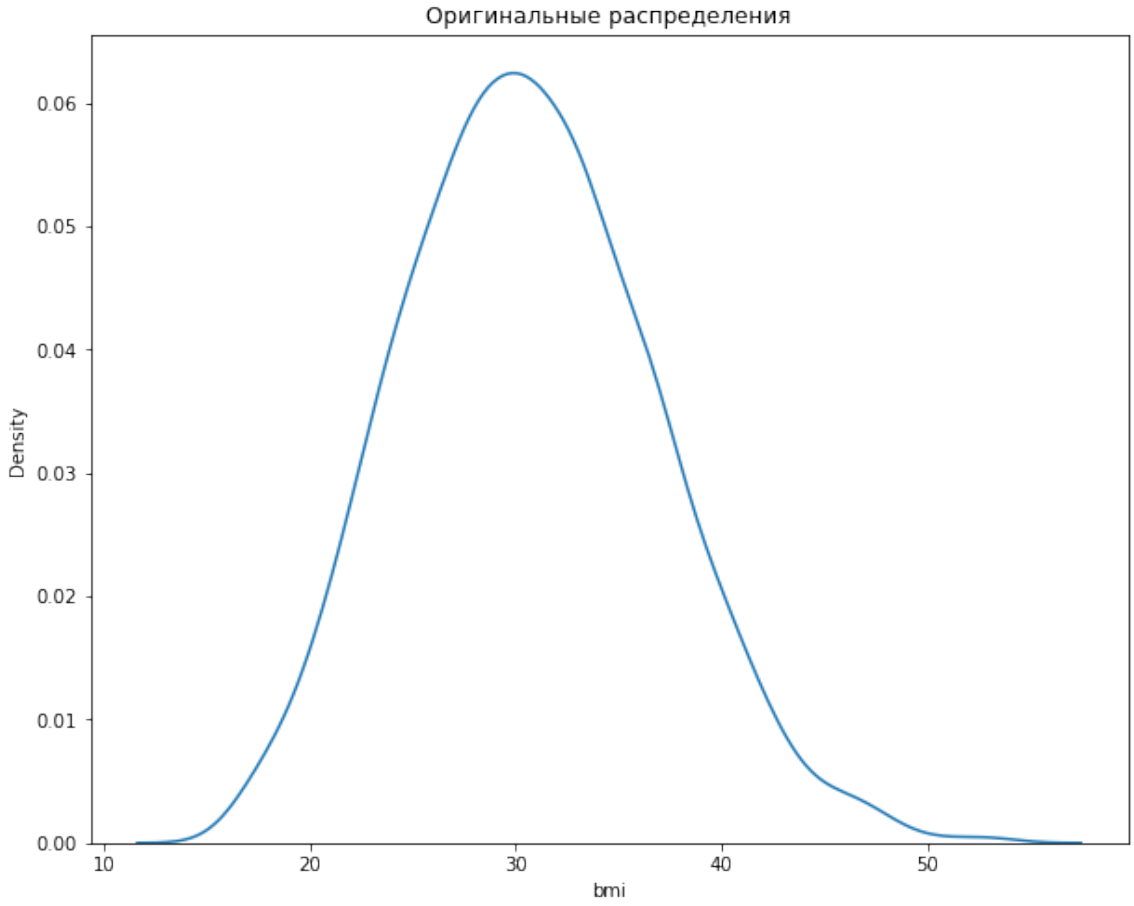


Рис. 25: Признак **bmi** до стандартизации.

```
print("There are {} missing values in the data.".format(ins.isna().
    sum().sum()))


ins['sex'] = ins['sex'].str.capitalize()
ins['smoker'] = ins['smoker'].apply(lambda x: 'Smoker' if x=='yes'
    else 'Non-Smoker')
ins['region'] = ins['region'].str.capitalize()


init_notebook_mode(connected=True)
px.defaults.template = "plotly_white"
plot_df=ins.copy()
fig = px.box(plot_df, x="region", y="charges", color="region",
             notched=True, points="outliers", height=600,
             title="",
```

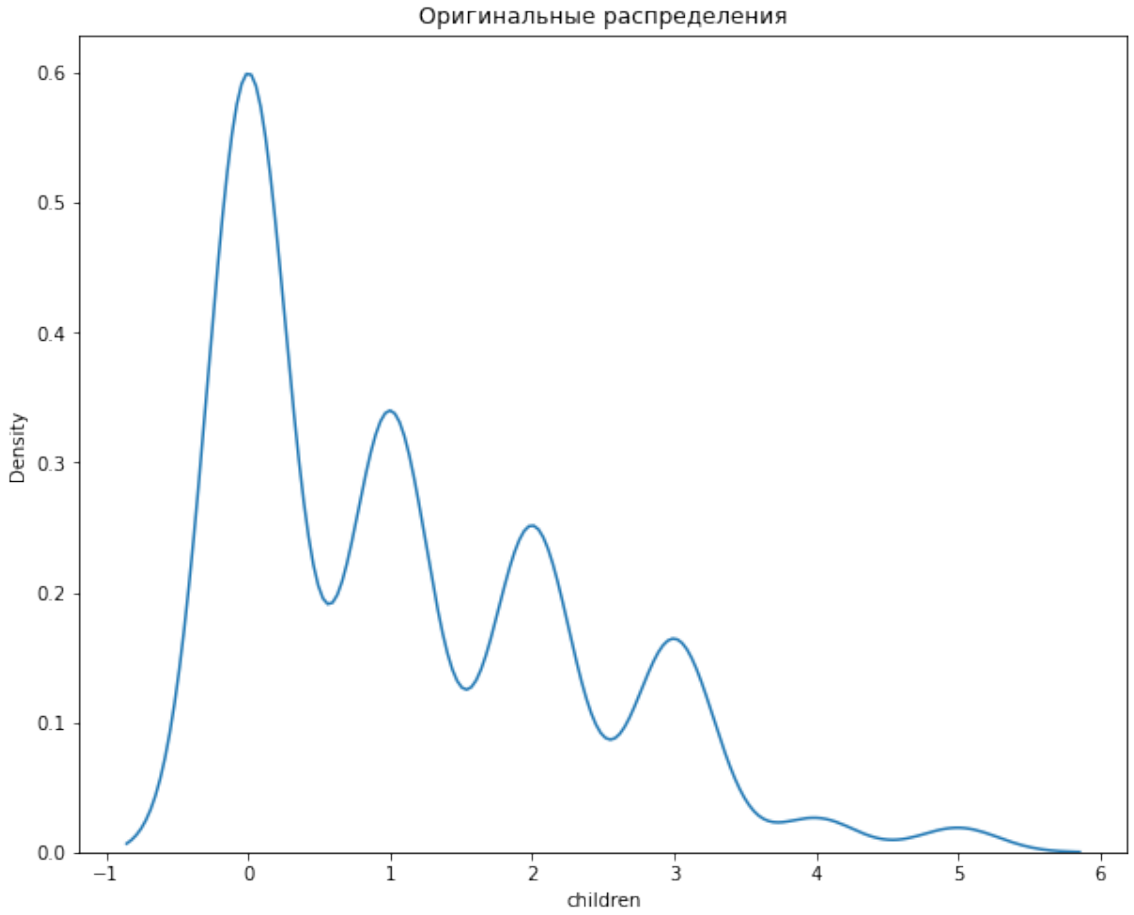


Рис. 26: Признак **children** до стандартизации.

```
            color_discrete_sequence=['#B14B51', '#D0A99C', '#5D8370', '#6C839B'])
fig.update_traces(marker=dict(size=9, opacity=0.5, line=dict(width=1,color="#F7F7F7")), showlegend=False)
fig.update_layout(font_color="#303030", xaxis_title='Region',
    yaxis_title='Claim Amount, $',
                  yaxis=dict(showgrid=True, gridwidth=1, gridcolor='#EAEAEA', zerolinecolor='#EAEAEA'))
fig.show()


ins['female'] = ins['sex'].apply(lambda x: 1 if x=='Female' else 0)
ins['smoker_yes'] = ins['smoker'].apply(lambda x: 1 if x=='Smoker' else 0)
ins.drop(['sex', 'smoker'], axis=1, inplace=True)
```

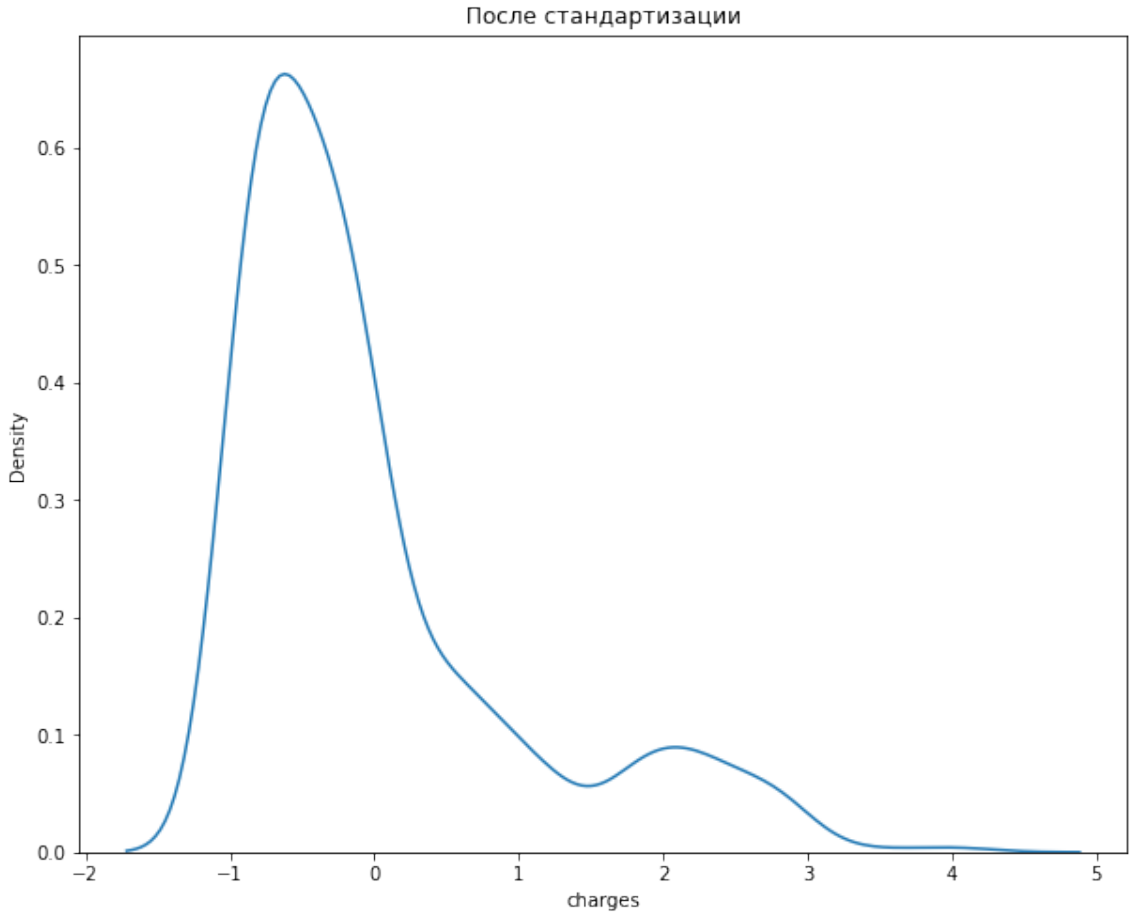


Рис. 27: Признак **charges** после стандартизации.

```
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()
```

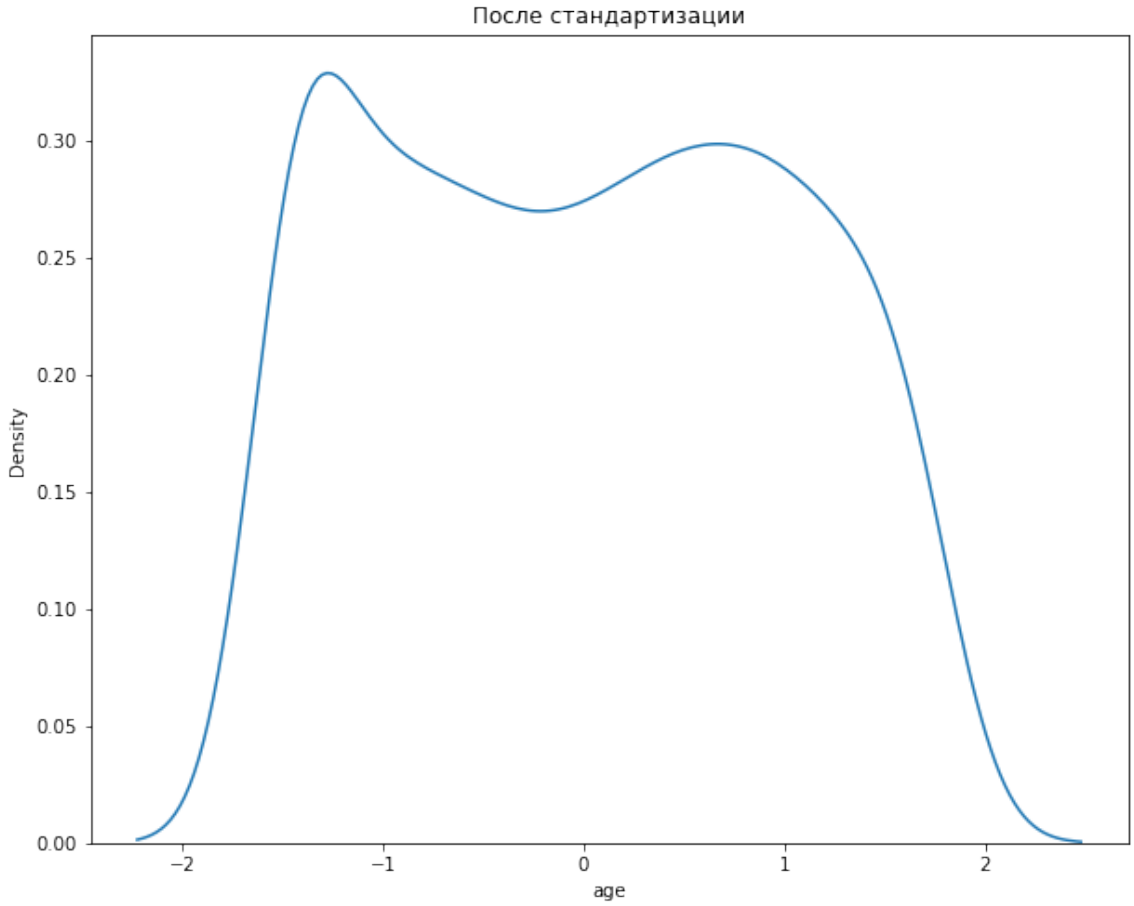


Рис. 28: Признак **age** после стандартизации.

```
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'])
```

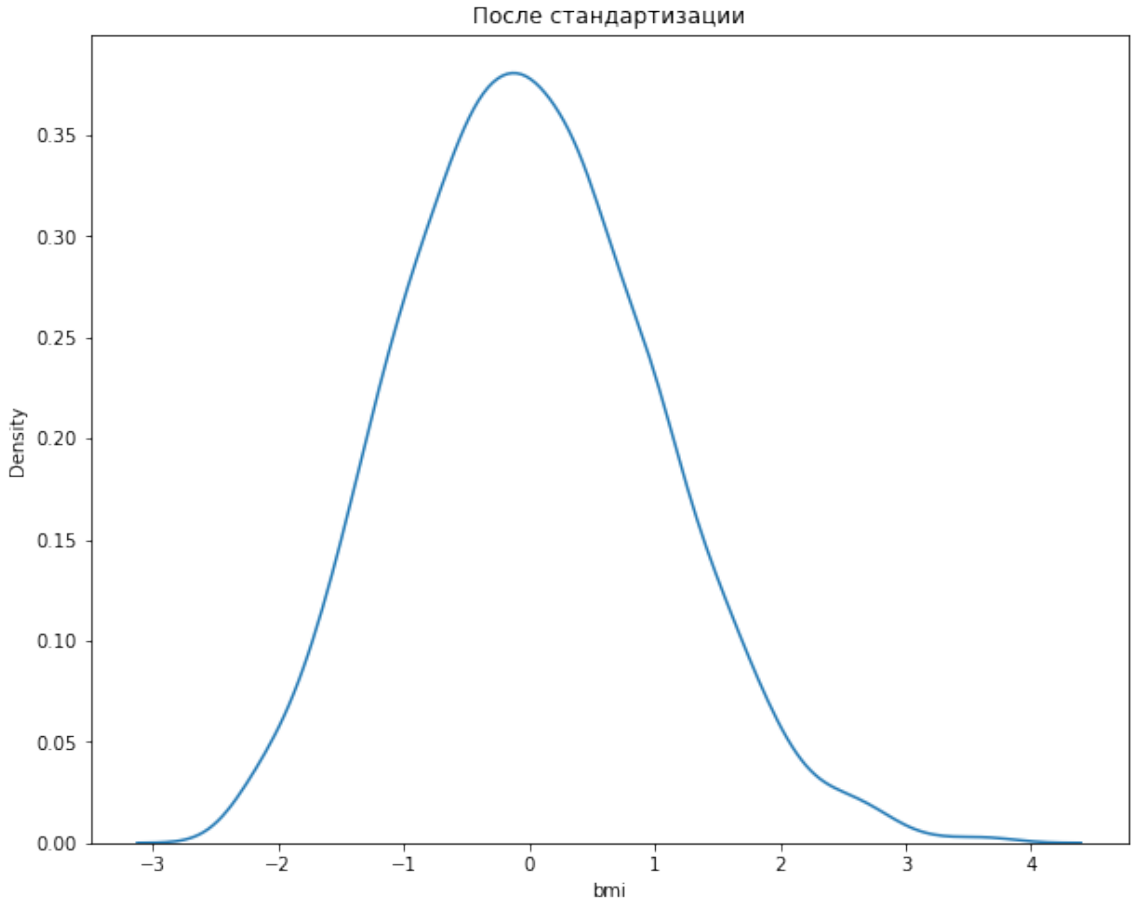


Рис. 29: Признак **bmi** после стандартизации.

```
data = pd.concat((data, region), axis=1)
data = data.loc[:, data.columns.isin(['age', 'female', 'bmi', '
    children' , 'smoker_yes', 'charges', 'northeast', 'northwest','
    southeast', 'southwest'])]

data.head(7)

y_train=data.pop('charges')
X_train = data.copy()

import statsmodels.api as sm
X_train_sm = sm.add_constant(X_train)

lm= sm.OLS(y_train, X_train_sm).fit()

lm.summary()
```

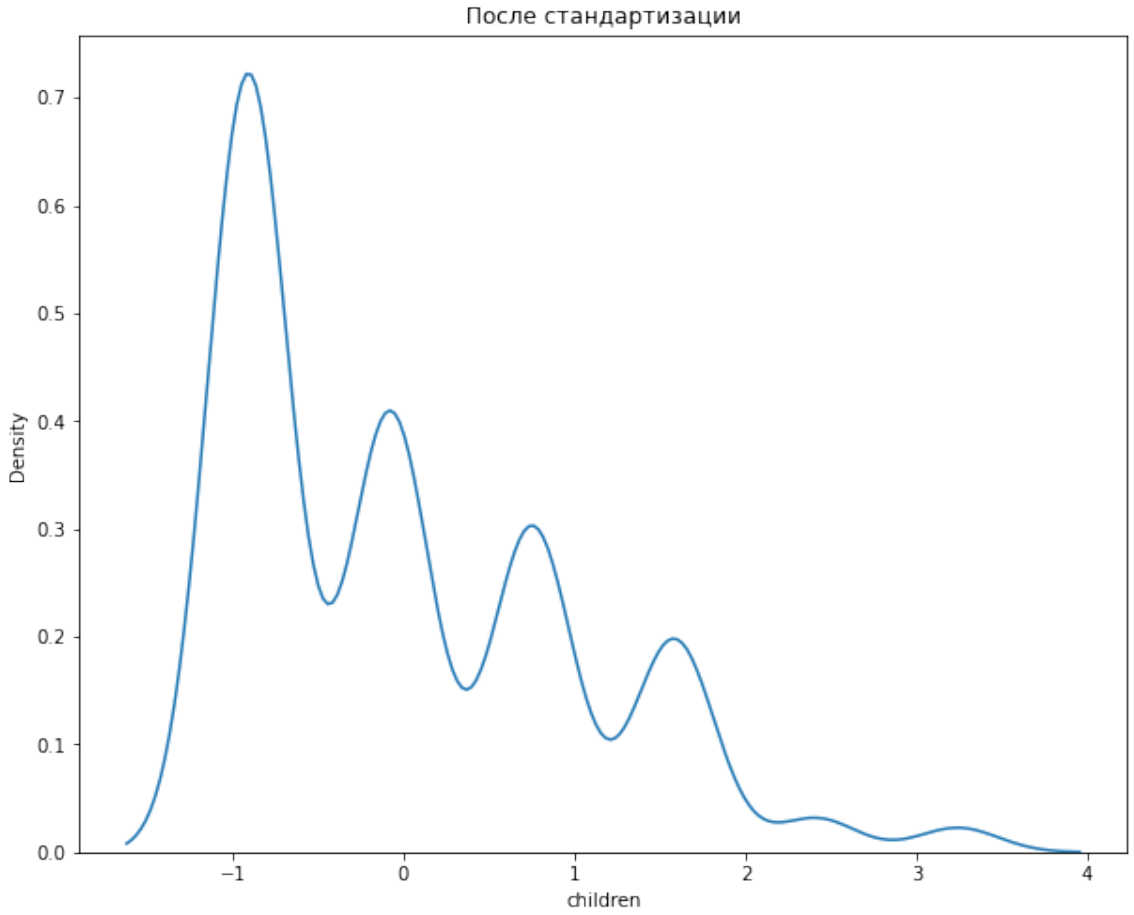


Рис. 30: Признак **children** после стандартизации.

```
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
```

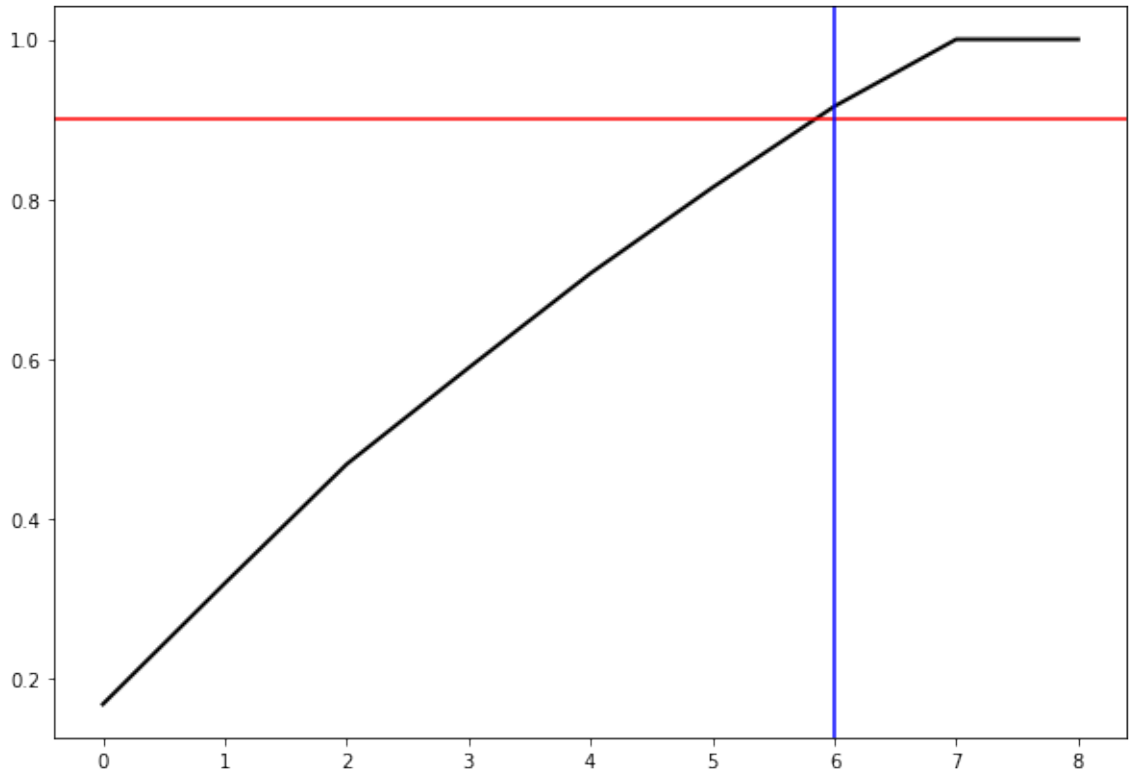


Рис. 31: Визуализация зависимости качества описываемой дисперсии от количество описывающих признаков

```
bmi_sq.summary()


# y_train=data.pop('charges')
X_train = data.copy()
for i in range(0 , len(X_train['bmi'])):
    X_train['bmi'][i] = X_train['bmi'][i] * X_train['bmi'][i] *
   X_train['bmi'][i]

# X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))

import statsmodels.api as sm
X_train_sm = sm.add_constant(X_train)

bmi_sq= sm.OLS(y_train, X_train_sm).fit()

bmi_sq.summary()

```

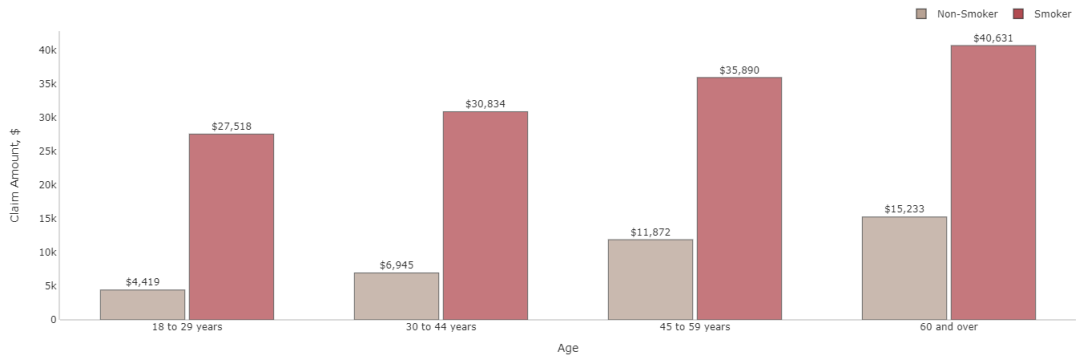


Рис. 32: Визуализация распределения стоимости медицинской страховки среди курящих и некурящих разных возрастов.

```
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] *
    X_train['age'][i]
120
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
```

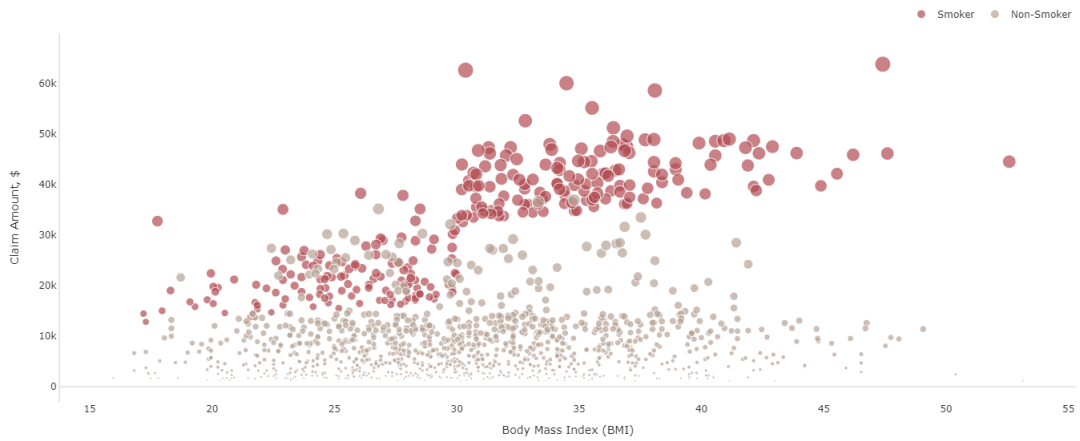


Рис. 33: Визуализация распределения стоимости медицинской страховки среди курящих и некурящих разных возрастов.

```
# y_train=data.pop('charges')
X_train = data.copy()
for i in range(0 , len(X_train['age'])):
    X_train['age'][i] = pow(X_train['age'][i], 0.8)

# X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))

import statsmodels.api as sm
X_train_sm = sm.add_constant(X_train)

bmi_sq= sm.OLS(y_train, X_train_sm).fit()

bmi_sq.summary()


# y_train=data.pop('charges')
X_train = data.copy()
for i in range(0 , len(X_train['age'])):
    X_train['age'][i] = pow(X_train['age'][i], 1.2)

# X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))

import statsmodels.api as sm
X_train_sm = sm.add_constant(X_train)

bmi_sq= sm.OLS(y_train, X_train_sm).fit()

bmi_sq.summary()
```

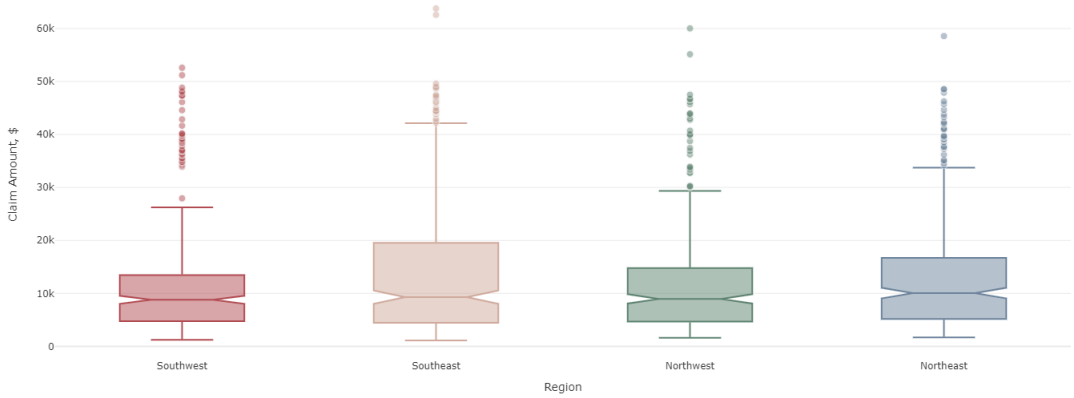


Рис. 34: Визуализация стоимости медицинской страховки к каждом регионе

```
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()
```

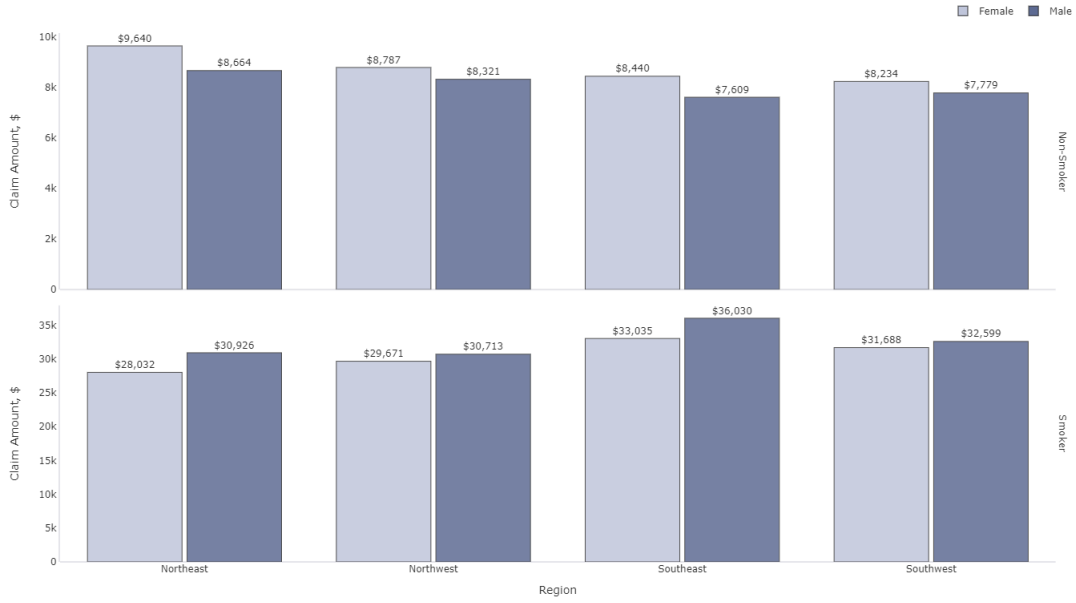


Рис. 35: Визуализация распределения стоимости медицинской страховки.

```
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
```

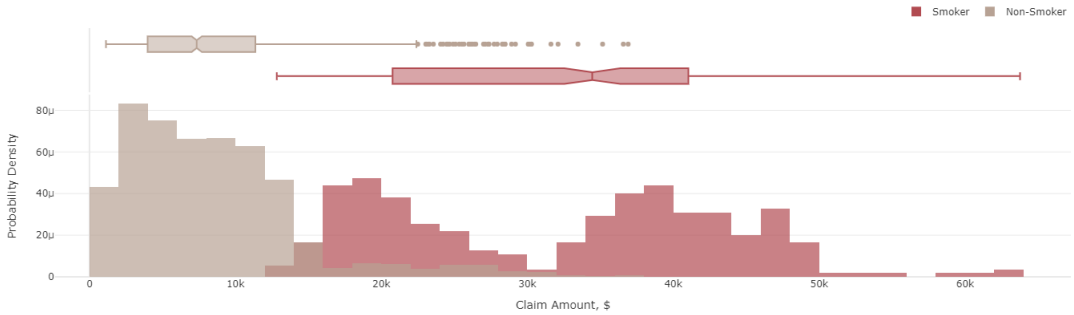


Рис. 36: Визуализация распределения стоимости медицинской страховки среди курящих и некурящих.

```
import statsmodels.api as sm
X_train_sm = sm.add_constant(X_train)

bmi_sq= sm.OLS(y_train, X_train_sm).fit()

bmi_sq.summary()



# y_train=data.pop('charges')
X_train = data.copy()
for i in range(0 , len(X_train['bmi'])):
    X_train['bmi'][i] = pow(X_train['bmi'][i], 1.6)

# X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))

import statsmodels.api as sm
X_train_sm = sm.add_constant(X_train)

bmi_sq= sm.OLS(y_train, X_train_sm).fit()

bmi_sq.summary()


# y_train=data.pop('charges')
X_train = data.copy()
for i in range(0 , len(X_train['bmi'])):
    X_train['bmi'][i] = pow(X_train['bmi'][i], 1.6)
    X_train['age'][i] = pow(X_train['age'][i], 0.4)
# X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))

import statsmodels.api as sm
```

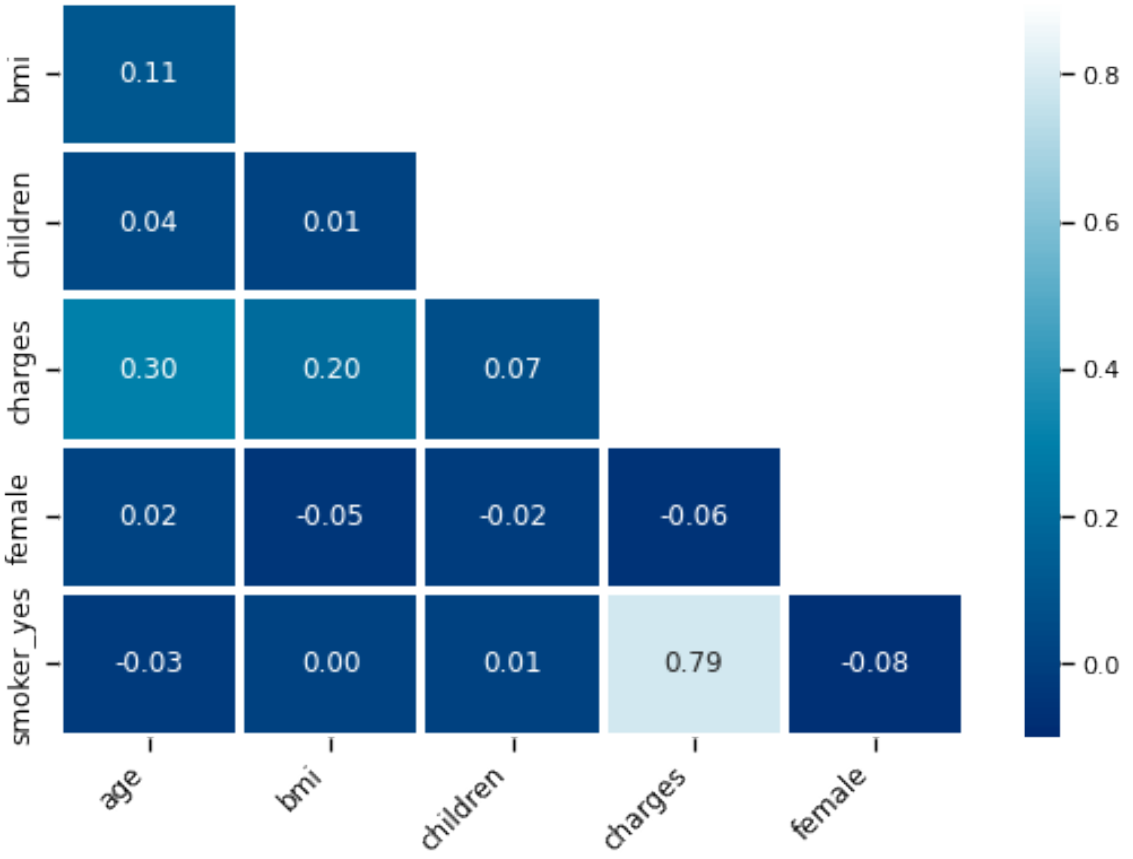


Рис. 37: Коррялицонная матрица.

```
X_train_sm = sm.add_constant(X_train)

bmi_sq= sm.OLS(y_train, X_train_sm).fit()

bmi_sq.summary()


# y_train=data.pop('charges')
X_train = data.copy()
for i in range(0 , len(X_train['bmi'])):
    X_train['bmi'][i] = pow(X_train['bmi'][i], 1.6)
    X_train['age'][i] = pow(X_train['age'][i], 0.8)
# X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))

import statsmodels.api as sm
X_train_sm = sm.add_constant(X_train)

```

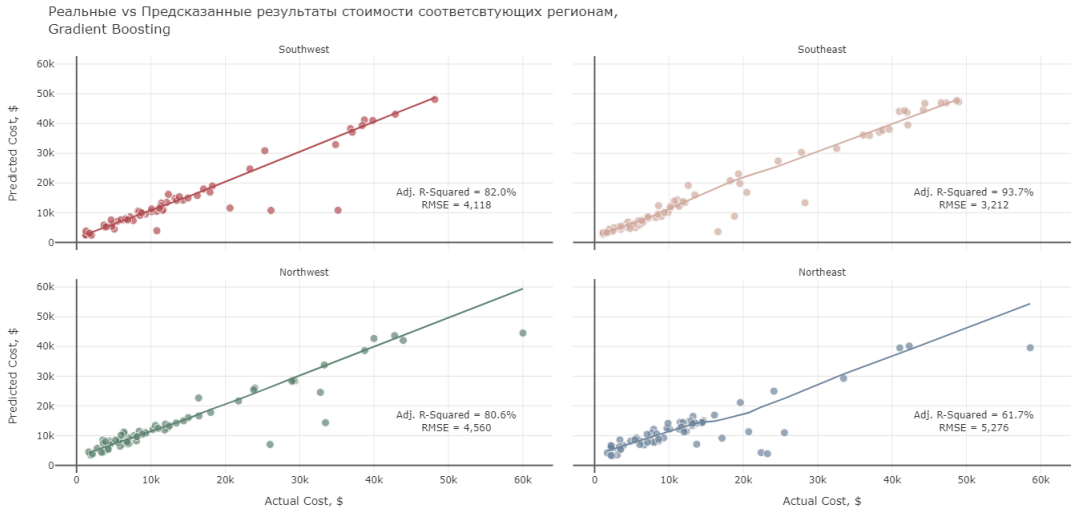


Рис. 38: Результат работы модели.

```
bmi_sq= sm.OLS(y_train, X_train_sm).fit()

bmi_sq.summary()



# y_train=data.pop('charges')
X_train = data.copy()
for i in range(0 , len(X_train['bmi'])):
    X_train['bmi'][i] = pow(X_train['bmi'][i], 1.6)
    X_train['age'][i] = pow(X_train['age'][i], 1.2)
# X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))

import statsmodels.api as sm
X_train_sm = sm.add_constant(X_train)

bmi_sq= sm.OLS(y_train, X_train_sm).fit()

bmi_sq.summary()



# y_train=data.pop('charges')
X_train = data.copy()
for i in range(0 , len(X_train['bmi'])):
    X_train['bmi'][i] = pow(X_train['bmi'][i], 1.6)
    X_train['age'][i] = pow(X_train['age'][i], 1.6)
```

```
# X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))

import statsmodels.api as sm
X_train_sm = sm.add_constant(X_train)

bmi_sq= sm.OLS(y_train, X_train_sm).fit()

bmi_sq.summary()



# y_train=data.pop('charges')
X_train = data.copy()
for i in range(0 , len(X_train['bmi'])):
    X_train['bmi'][i] = pow(X_train['bmi'][i], 1.2)
    X_train['age'][i] = pow(X_train['age'][i], 0.4)
# X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))

import statsmodels.api as sm
X_train_sm = sm.add_constant(X_train)

bmi_sq= sm.OLS(y_train, X_train_sm).fit()

bmi_sq.summary()



# y_train=data.pop('charges')
X_train = data.copy()
for i in range(0 , len(X_train['bmi'])):
    X_train['bmi'][i] = pow(X_train['bmi'][i], 1.2)
    X_train['age'][i] = pow(X_train['age'][i], 0.8)
# X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))

import statsmodels.api as sm
X_train_sm = sm.add_constant(X_train)

bmi_sq= sm.OLS(y_train, X_train_sm).fit()

bmi_sq.summary()









# y_train=data.pop('charges')
X_train = data.copy()
for i in range(0 , len(X_train['bmi'])):
```

```
    X_train['bmi'][i] = pow(X_train['bmi'][i], 1.2)
    X_train['age'][i] = pow(X_train['age'][i], 0.8)
# X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))

import statsmodels.api as sm
X_train_sm = sm.add_constant(X_train)

bmi_sq= sm.OLS(y_train, X_train_sm).fit()

bmi_sq.summary()



# y_train=data.pop('charges')
X_train = data.copy()
for i in range(0 , len(X_train['bmi'])):
    X_train['bmi'][i] = pow(X_train['bmi'][i], 1.2)
    X_train['age'][i] = pow(X_train['age'][i], 1.6)
# X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))

import statsmodels.api as sm
X_train_sm = sm.add_constant(X_train)

bmi_sq= sm.OLS(y_train, X_train_sm).fit()

bmi_sq.summary()







# y_train=data.pop('charges')
X_train = data.copy()
for i in range(0 , len(X_train['bmi'])):
    X_train['bmi'][i] = pow(X_train['bmi'][i], 0.8)
    X_train['age'][i] = pow(X_train['age'][i], 1.6)
# X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))

import statsmodels.api as sm
X_train_sm = sm.add_constant(X_train)

bmi_sq= sm.OLS(y_train, X_train_sm).fit()

bmi_sq.summary()


# y_train=data.pop('charges')
X_train = data.copy()
for i in range(0 , len(X_train['bmi'])):
```

```
    X_train['bmi'][i] = pow(X_train['bmi'][i], 0.8)
    X_train['age'][i] = pow(X_train['age'][i], 1.2)
# X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))

import statsmodels.api as sm
X_train_sm = sm.add_constant(X_train)

bmi_sq= sm.OLS(y_train, X_train_sm).fit()

bmi_sq.summary()


# y_train=data.pop('charges')
X_train = data.copy()
for i in range(0 , len(X_train['bmi'])):
    X_train['bmi'][i] = pow(X_train['bmi'][i], 0.8)
    X_train['age'][i] = pow(X_train['age'][i], 0.8)
# X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))

import statsmodels.api as sm
X_train_sm = sm.add_constant(X_train)

bmi_sq= sm.OLS(y_train, X_train_sm).fit()

bmi_sq.summary()



# y_train=data.pop('charges')
X_train = data.copy()
for i in range(0 , len(X_train['bmi'])):
    X_train['bmi'][i] = pow(X_train['bmi'][i], 0.8)
    X_train['age'][i] = pow(X_train['age'][i], 0.4)
# X_train['bmi'] = list(map(lambda x: x, X_train['bmi']))

import statsmodels.api as sm
X_train_sm = sm.add_constant(X_train)

bmi_sq= sm.OLS(y_train, X_train_sm).fit()

bmi_sq.summary()









# y_train=data.pop('charges')
X_train = data.copy()
```

```
for i in range(0 , len(X_train['age'])):
    X_train['age'][i] = X_train['age'][i] * X_train['age'][i]


import statsmodels.api as sm
X_train_sm = sm.add_constant(X_train)

age_sq= sm.OLS(y_train, X_train_sm).fit()

age_sq.summary()







# y_train=data.pop('charges')
X_train = data.copy()
for i in range(0 , len(X_train['age'])):
    X_train['age'][i] = X_train['age'][i] * X_train['age'][i]
    X_train['bmi'][i] = X_train['bmi'][i] * X_train['bmi'][i]


import statsmodels.api as sm
X_train_sm = sm.add_constant(X_train)

bmi_age_sq= sm.OLS(y_train, X_train_sm).fit()

bmi_age_sq.summary()

# y_train=data.pop('charges')
X_train = data.copy()
for i in range(0 , len(X_train['age'])):
    X_train['age'][i] = X_train['age'][i] * X_train['age'][i]
    X_train['bmi'][i] = X_train['bmi'][i] * X_train['bmi'][i]


import statsmodels.api as sm
X_train_sm = sm.add_constant(X_train)

lm_5= sm.OLS(y_train, X_train_sm).fit()
lm_5.summary()

# y_train=data.pop('charges')

X_train = data.copy()
for i in range(0 , len(X_train['age'])):
    X_train['age'][i] = math.log(X_train['age'][i],10)


```

```
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
```

```
# y_train=data.pop('charges')


X_train = data.loc[:, data.columns.isin(['bmi', 'smoker_yes',])]

import statsmodels.api as sm
X_train_sm = sm.add_constant(X_train)

lm_9= sm.OLS(y_train, X_train_sm).fit()
lm_9.summary()

# y_train=data.pop('charges')

X_train = data.loc[:, data.columns.isin(['age', 'smoker_yes',])]


import statsmodels.api as sm
X_train_sm = sm.add_constant(X_train)

age_smoke= sm.OLS(y_train, X_train_sm).fit()
age_smoke.summary()


# y_train=data.pop('charges')
X_train = data.copy()
for i in range(0 , len(X_train['smoker_yes'])):
    X_train['smoker_yes'][i] = X_train['smoker_yes'][i] * X_train['
   smoker_yes'][i]


import statsmodels.api as sm
X_train_sm = sm.add_constant(X_train)

lm_11= sm.OLS(y_train, X_train_sm).fit()
lm_11.summary()

# y_train=data.pop('charges')
X_train = data.loc[:, data.columns.isin(['age', 'bmi',])]


import statsmodels.api as sm
X_train_sm = sm.add_constant(X_train)

lm_12= sm.OLS(y_train, X_train_sm).fit()
lm_12.summary()



# y_train=data.pop('charges')
```

```
y_train=data.pop('smoker_yes')
# X_train = [data['female'].copy(), data['bmi'].copy()]
X_train = data.loc[:, data.columns.isin(['age', 'female', 'bmi', '
    children' , 'charges', 'northeast', 'northwest','southeast', '
    southwest'])]

import statsmodels.api as sm
X_train_sm = sm.add_constant(X_train)

lm_13= sm.OLS(y_train, X_train_sm).fit()
lm_13.summary()


# y_train=data.pop('charges')
y_train=data.pop('bmi')
# X_train = [data['female'].copy(), data['bmi'].copy()]
X_train = data.loc[:, data.columns.isin(['age', 'female', '
    smoker_yes', 'children' , 'charges', 'northeast', 'northwest','
    southeast', 'southwest'])]

import statsmodels.api as sm
X_train_sm = sm.add_constant(X_train)

lm_20= sm.OLS(y_train, X_train_sm).fit()
lm_20.summary()


# Dataframe to save results


regions = ins.region.unique()
s = StandardScaler()
# feat_importance=pd.DataFrame()
actuals=[]
preds=[]
rmses=[]
r2_scores=[]
adj_r2_scores=[]

for i in regions:

    # Filter data by region
    print("\nRegion: {}\n".format(i))
    ins_df = ins[ins.region==i]
    X=ins_df.drop(['charges', 'region'], axis=1)
    y=ins_df.charges

    # Add polynomial features
    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     rmses.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)
```

```
col = ["#B14B51", '#D0A99C', '#5D8370','#6C839B']
fig = px.scatter(df, x="actuals", y="preds", color="index",
    trendline="ols", height=700,
                 title="Actual vs Predicted Insurance Costs by
    Region,<br>Linear Regression with Principal Component Analysis",
                 color_discrete_sequence=col, opacity=0.7, facet_col
    ='index', facet_col_wrap=2)

fig.for_each_annotation(lambda a: a.update(text=a.text.split("=")
    [-1]))
fig.add_annotation(text="Adj. R-Squared = {:.1f}%<br>RMSE = {:,.0f}"
    .format(adj_r2_scores[0]*100,rmses[0]),
                   x=51e3,y=15e3, row=2,col=1, showarrow=False)
fig.add_annotation(text="Adj. R-Squared = {:.1f}%<br>RMSE = {:,.0f}"
    .format(adj_r2_scores[1]*100,rmses[1]),
                   x=51e3,y=15e3, row=2,col=2, showarrow=False)
fig.add_annotation(text="Adj. R-Squared = {:.1f}%<br>RMSE = {:,.0f}"
    .format(adj_r2_scores[2]*100,rmses[2]),
                   x=51e3,y=15e3, row=1,col=1, showarrow=False)
fig.add_annotation(text="Adj. R-Squared = {:.1f}%<br>RMSE = {:,.0f}"
    .format(adj_r2_scores[3]*100,rmses[3]),
                   x=51e3,y=15e3, row=1,col=2, showarrow=False)

fig.update_traces(hovertemplate="Actual Cost: %{x:$,.2f}<br>
    Predicted Cost: %{y:$,.2f}",
                  marker=dict(size=10, line=dict(width=1,color="#
    F7F7F7")),
                   selector=dict(mode="markers"), showlegend=False)
fig.update_xaxes(title="Actual Cost, $", row=1)
fig.update_xaxes(showgrid=True, gridwidth=1, gridcolor='#EAEAEA',
                 zeroline=True, zerolinewidth=2, zerolinecolor='#5
    E5E5E')
fig.update_yaxes(title="Predicted Cost, $", col=1)
fig.update_yaxes(showgrid=True, gridwidth=1, gridcolor='#E3E3E3',
                 zeroline=True, zerolinewidth=2, zerolinecolor='#5
    E5E5E')
fig.update_layout(font_color="#303030", paper_bgcolor="white",
    plot_bgcolor="white")
fig.show()



actuals=[]
preds=[]
rmses=[]
r2_scores=[]
adj_r2_scores=[]
feat_importance=pd.DataFrame()
```

```
regions = ins.region.unique()
s = StandardScaler()
col = ["#B14B51", '#D0A99C', '#5D8370','#6C839B']

for i in regions:

    # Разделение датасета по региону
    ins_df = ins[ins.region==i]
    X=ins_df.drop(['charges', 'region'], axis=1)
    y=ins_df.charges



    pf = PolynomialFeatures(degree=2, include_bias=False)
    X_pf = pd.DataFrame(data=pf.fit_transform(X), columns=pf.
    get_feature_names(X.columns))

    # Распределение на выборки
    X_train, X_test, y_train, y_test = train_test_split(X_pf, y,
    test_size=0.2, random_state=1)
    X_train = pd.DataFrame(X_train, columns = X_pf.columns)
    X_test = pd.DataFrame(X_test, columns = X_pf.columns)
    actuals.append(pd.Series(y_test, name='actuals').reset_index())
    print("\nRegion: {}\n".format(i))
    print("Train Shape:{} {}  Test Shape:{} {}".format(X_train.shape
    , y_train.shape, X_test.shape, y_test.shape))


    X_train = pd.DataFrame(data=s.fit_transform(X_train), columns=
    X_pf.columns)
    X_test = pd.DataFrame(data=s.transform(X_test), columns=X_pf.
    columns)


    grid = {'learning_rate': [0.0001, 0.001, 0.01, 0.1, 0.25, 0.5],
            'n_estimators': [int(x) for x in np.linspace(start =
    200, stop = 1000, num = 5)],
            'subsample': [0.5, 0.8, 1],
            'min_samples_split': [2, 5, 10],
            'min_samples_leaf': [1, 2, 4],
            'max_depth': [int(x) for x in np.linspace(2, 10, num =
    5)],
            'max_features': [None, 'sqrt']}
    xgb=GradientBoostingRegressor(random_state=21)
    xgb_cv=RandomizedSearchCV(estimator=xgb, param_distributions=
    grid, scoring='neg_mean_squared_error',
                              n_iter=100, cv=3, random_state=21,
    n_jobs=-1)
    xgb_cv.fit(X_train, y_train)
```

```
    y_pred=xgb_cv.predict(X_test)
    preds.append(pd.Series(y_pred, name='preds').reset_index(drop=True))
    rmse=np.sqrt(mean_squared_error(y_test, y_pred)).round(2)
    r2=r2_score(y_test, y_pred)
    adj_r2 = 1 - (1-r2)*(len(y_test)-1)/(len(y_test)-X_test.shape[1]-1)
    rmses.append(rmse)
    r2_scores.append(r2)
    adj_r2_scores.append(adj_r2)


    feat_importance["Importance_"+str(i)]=xgb_cv.best_estimator_.feature_importances_

    print("Test Error (RMSE) = {:,}".format(rmse))
    print("R-Squared = {:.2f}%, Adjusted R-Squared = {:.2f}%".format(r2*100, adj_r2*100))
    if i != 'Northeast':
        print("___________________________")

# Plot results
for i in range(0,4):
    actuals[i].loc[:,'index']=regions[i]
actual = pd.concat([actuals[i] for i in range(4)], axis = 0)
pred = pd.concat([preds[i] for i in range(4)], axis = 0)
df = pd.concat([actual, pred], axis=1).reset_index(drop=True)

fig = px.scatter(df, x="actuals", y="preds", color="index", trendline="lowess", height=700,
                 title="Реальные vs Предсказанные результаты стоимости соответсвующих регионам,<br>Gradient Boosting",
                 color_discrete_sequence=col, opacity=0.7, facet_col='index', facet_col_wrap=2)

fig.for_each_annotation(lambda a: a.update(text=a.text.split("=")[-1]))
fig.add_annotation(text="Adj. R-Squared = {:.1f}%<br>RMSE = {:,.0f}".format(adj_r2_scores[0]*100,rmses[0]),
                   x=51e3,y=15e3, row=2,col=1, showarrow=False)
fig.add_annotation(text="Adj. R-Squared = {:.1f}%<br>RMSE = {:,.0f}".format(adj_r2_scores[1]*100,rmses[1]),
                   x=51e3,y=15e3, row=2,col=2, showarrow=False)
fig.add_annotation(text="Adj. R-Squared = {:.1f}%<br>RMSE = {:,.0f}".format(adj_r2_scores[2]*100,rmses[2]),
                   x=51e3,y=15e3, row=1,col=1, showarrow=False)
fig.add_annotation(text="Adj. R-Squared = {:.1f}%<br>RMSE = {:,.0f}".format(adj_r2_scores[3]*100,rmses[3]),
                   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($satisfy^2$)	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	M=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: Предствавление 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: Представвление 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

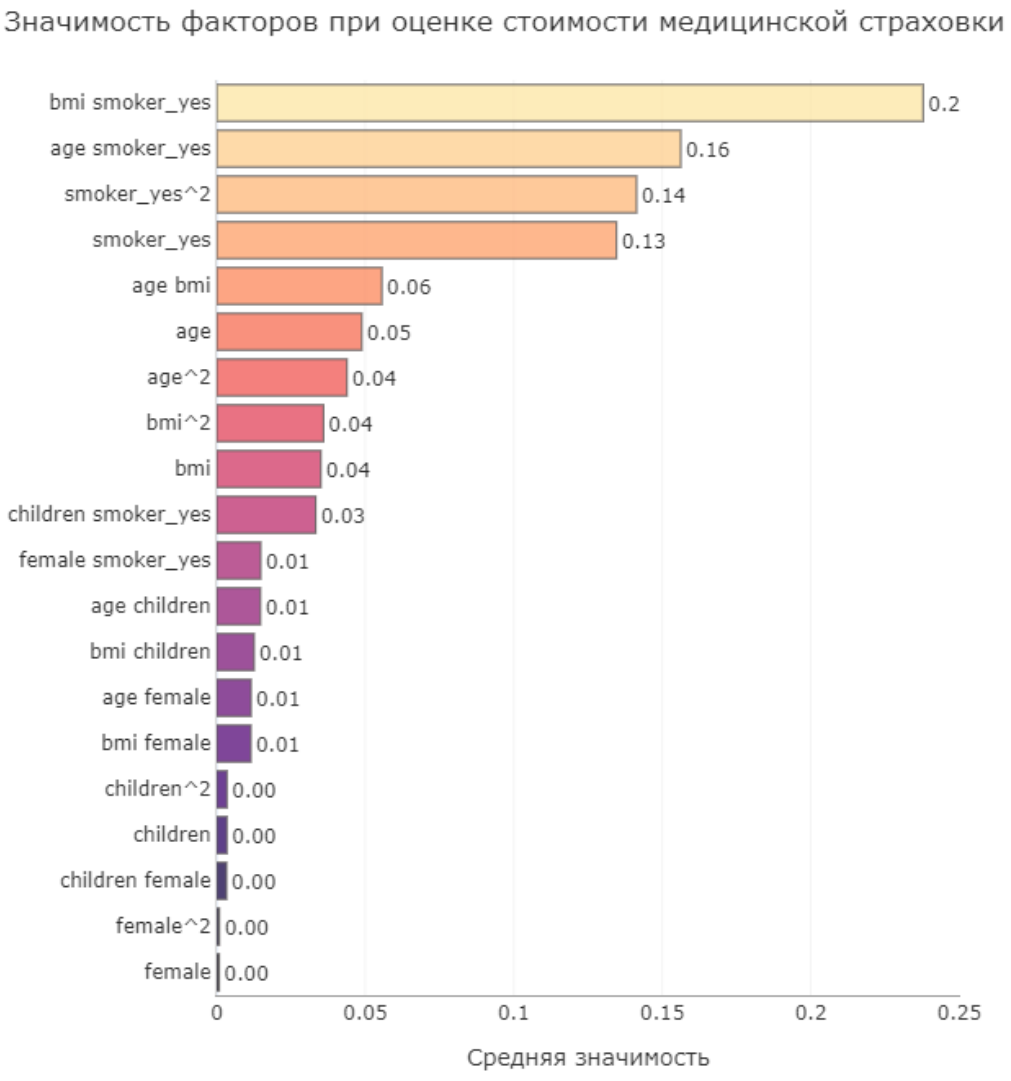
	Column	Non-Null	Count	Dtype
0	age	1338	non-null	int64
1	sex	1338	non-null	object
2	bmi	1338	non-null	float64
3	children	1338	non-null	int64
4	smoker	1338	non-null	object
5	charges	1338	non-null	float64
6	northeast	1338	non-null	uint8
7	northwest	1338	non-null	uint8
8	southeast	1338	non-null	uint8
9	southwest	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
bmi smoker_yes
age smoker_yes
smoker_yes^2
smoker_yes
age bmi
age
age^2
bmi^2
bmi
children smoker_yes
female smoker_yes
age children
bmi children
age female
bmi female
children^2
children
children female
female^2
female
0.2
0.16
0.14
0.13
0.06
0.05
0.04
0.04
0.04
0.03
0.01
0.01
0.01
0.01
0.01
0.00
0.00
0.00
0.00
0.00
0
0.05
0.1
0.15
0.2
0.25
Средняя значимость