



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: Эффективная программная реализация вычисления сплайнов Коханек-
Бартельса

приказ университета о направлении на НИР
от «9» февраля 2023 г. № 735 - С


Отчет представлен к
рассмотрению:
Студент группы КМБО-07-22

Невский В.Е.
(расшифровка подписи)
«9» февраля 2023г.

Отчет утвержден.
Допущен к защите:

Руководитель НИР от
кафедры

Парфенов Д.В.
(расшифровка подписи)
«9» июня 2023г.

Карьерная и career-ориентированная работа, Оценки
«осуждено»  /Парфенов С.В./

Москва 2023



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-07-22 института искусственного
интеллекта Невскому Владиславу Евгеньевичу

(фамилия, имя и отчество)

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

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

Должность на НИР: практикант

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

2. СОДЕРЖАНИЕ НИР:

2.1 Изучить: литературу и практические примеры по темам: 1) построение линейной регрессии, 2) использование метода главных компонент, 3) поиск и устранение линейной зависимости в данных, 4) основы нормализации данных, 5) методы классификации и кластеризации («решающее дерево», «случайный лес», «k ближайших соседей»), 6) сплайн-аппроксимация.

2.2 Практически выполнить: осуществить алгоритмическую и программную оптимизацию реализации вычисления сплайнов Коханек-Бартельса, изучить поведение сплайнов на наборах данных.

2.3 Ознакомиться: с применением сплайн-приближений, их свойствами и возможностями; построением модели данных на основе сплайна.

3. ДОПОЛНИТЕЛЬНОЕ ЗАДАНИЕ: Осуществить минимизацию объема вычислений при построении сплайн-модели данных.

4. ОГРАНИЗАЦИОННО-МЕТОДИЧЕСКИЕ УКАЗАНИЯ: выделить основные факторы, определяющие поведение сплайн-приближения и связать их с параметрами модели и особенностями набора данных; найти зависимости между гладкостью сплайн-кривой и параметрами.

Заведующий кафедрой
высшей математики

«09» февраля 2023 г.

СОГЛАСОВАНО

Руководитель НИР от кафедры:

«09» февраля 2023 г.

Задание получил:

«09» февраля 2023 г.

Ю.И. Худак

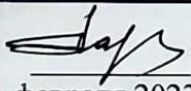
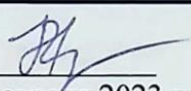
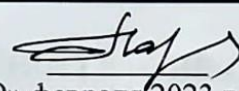
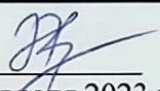
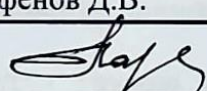
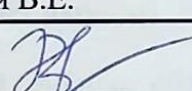
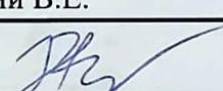
(подпись)

(Парфенов Д.В.)
(фамилия и инициалы)

(подпись)

(Невский В.Е.)
(фамилия и инициалы)

ИНСТРУКТАЖ ПРОВЕДЕН:

Вид мероприятия	ФИО ответственного, подпись, дата	ФИО студента, подпись, дата
Охрана труда	Парфенов Д.В.  «09» февраля 2023 г.	Невский В.Е.  «09» февраля 2023 г.
Техника безопасности	Парфенов Д.В.  «09» февраля 2023 г.	Невский В.Е.  «09» февраля 2023 г.
Пожарная безопасность	Парфенов Д.В.  «09» февраля 2023 г.	Невский В.Е.  «09» февраля 2023 г.
Правила внутреннего распорядка	Парфенов Д.В.  «09» февраля 2023 г.	Невский В.Е.  «09» февраля 2023 г.



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

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

студента Невского В.Е. 1 курса группы КМБО-07-22 очной формы обучения,
обучающегося по направлению подготовки 01.03.02 «Прикладная математика и
информатика»,
профиль «Математическое моделирование и вычислительная математика»

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

		Беггинг. Бустинг	<i>Восморт</i>
16	27.05.2023	Представление отчётных материалов по НИР и их защита. Передача обобщённых материалов на кафедру для архивного хранения	<i>Восморт.</i>
		Зачётная аттестация	<i>Восморт.</i>

Согласовано:

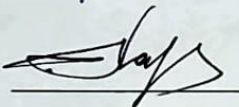
Заведующий кафедрой



/ ФИО /

Худак Ю.И.

Руководитель НИР от
кафедры



/ ФИО /

Парфенов Д.В.

Обучающийся



/ ФИО /

Невский В.Е.

Оглавление

Задача 1	9
1.1 Оценка среднего значения, дисперсии и СКО.....	9
1.2 Построение зависимости вида $y=a+bx$	9
1.3 Оценка модели по коэффициенту детерминации R^2	10
1.4 Оценка на наличие взаимосвязи между объясняемой и объясняющей переменной	10
1.5 Заключение	11
Задача 2.1.....	12
2.1.1 Проверка на отсутствие линейной независимости между регрессорами	12
2.1.2 Построение линейной модели и её оценка.....	12
2.1.3 Введение в модель логарифмов и выбор наилучшей.....	13
2.1.4 Введение в модель произведений пар регрессоров и квадратов регрессоров. Выбор наилучшей по R^2	14
2.1.5 Заключение	15
Задача 2.2.....	16
2.2.1 Оценка доверительных интервалов для всех коэффициентов	16
2.2.2 Вывод о статистической гипотезе	16
2.2.3 Оценка доверительного интервала для прогноза	16
2.2.4 Заключение	18
Задача 3.....	19
3.1 Построение линейной регрессии. Оценка коэффициентов вздутия дисперсии VIF	19
3.2 Добавление логарифмов, степеней и произведений регрессоров.....	21
3.3 Выделение наилучших моделей из построенных	22
3.4 Вывод об индивидах, получающих большую зарплату.....	23
3.5 Оценка и вывод для подмножества индивидов	23
3.6 Заключение	25
Задача 4.....	26

4.1 Обработка набора данных. Выделение целевого признака. Разделение набора данных на тестовую и обучающую выборки. Построение классификатора типа SVM. Оценка точности с помощью различных метрик на тестовой выборке.	26
4.2 Построение классификатора типа Случайный Лес. Оценка его качества с помощью различных метрик. Перебор различных комбинаций гиперпараметров с помощью GridSearch.....	27
4.3 Заключение	29
Задача 5.....	30
5.1 Определение сплайна и его параметров	30
5.2 Построение сплайновой кривой	32
5.3 Алгоритм построения	33
5.4 Реализация	33
5.5 Тестовый пример.....	36
5.6 Заключение	37
Приложения	41
Список литературы	70

Задача 1

Набор данных: Swiss.

Объясняемая переменная: Education.

Регрессоры: Fertility, Examination.

1. Оценить среднее значение, дисперсию и СКО для указанных переменных.

Используя код из *Приложения 1*, находим среднее значение, дисперсию и СКО для *Education*, *Fertility* и *Examination*. Вычисленные значения представлены в таблице 1.

Таблица 1. Характеристики переменных *Education*, *Fertility* и *Examination*.

Переменная	Среднее значение	Дисперсия	СКО
Education	10.97872	92.45606	9.615407
Fertility	70.14255	156.0425	12.4917
Examination	16.48936	63.64662	7.977883

2. Построить зависимость вида $y = a + bx$, где y — объясняемая переменная, x — регрессор.

Построим зависимости с помощью команды `lm` пакета `lmtest` (см. *приложение 2*). Характеристики моделей зависимости *Education* от *Examination* и *Fertility* приведены в таблицах 2 и 3 соответственно.

Таблица 2. Характеристики модели Зависимости *Education* от регрессора *Examination*.

Параметр	Значение	Std. Error	t value	Pr(> t)	Уровень значимости
(Intercept)	-2.9015	2.3507	-1.234	0.223	
Examination	0.8418	0.1286	6.546	4.81e-08	***

Таблица 3. Характеристики модели зависимости *Education* от регрессора *Fertility*.

Параметр	Значение	Std. Error	t value	Pr(> t)	Уровень значимости
(Intercept)	46.81788	6.11244	7.659	1.08e-09	***
Fertility	-0.51095	0.08582	-5.954	3.66e-07	***

В результате получаем зависимости вида:

$y = 0.8418 * x - 2.9015$, для зависимости *Education* от *Examination*.

$y = (-0.51095) * x + 46.81788$, для зависимости *Education* от *Fertility*.

3. Оценить, насколько хороша модель по коэффициенту детерминации R^2 .

R^2 для модели *Education~Examination* составляет 47%. Можем сделать вывод, что модель относительно хороша: для зависимости с одной объясняющей переменной коэффициент высок, но для полного описания нужно добавлять другие параметры.

В случае *Education~Fertility* R^2 равен 42%, что немного хуже, чем у предыдущей модели. Можно сказать, что данная модель также находится в пределах нормы.

4. Оценить, есть ли взаимосвязь между объясняемой и объясняющей переменной.

Используя данные из таблицы 2, видим, что коэффициент при *Examination* подобран хорошо: уровень значимости (***), а вероятность равна $4.81e-08$. Из этого следует, что взаимосвязь между *Education* и *Examination* велика.

В случае с *Fertility* имеем похожую ситуацию: уровень значимости (***), вероятность $3.66e-07$. Следовательно взаимосвязь между *Examination* и *Fertility* тоже высока.

Заключение

В результате исследования моделей зависимости Education от Examination и Fertility, можем сказать, что построенные нами модели являются относительно хорошими, что следует из низких значений р-статистики, высокого уровня значимости регрессоров и значений $R^2(>40\%)$. Такие модели мы можем использовать для дальнейших предсказаний значений объясняемой переменной Education.

Задача 2.1

Набор данных: Swiss.

Объясняемая переменная: Examination.

Регрессоры: Agriculture, Catholic, Fertility.

1. Проверить, что в наборе данных нет линейной зависимости между регрессорами (построить зависимости между регрессорами и проверить, что R^2 в каждой из них невысокий).

Проверим линейную регрессию $Fertility \sim Agriculture, Catholic$. R^2 в данной модели составляет 21%, поэтому делаем вывод, что параметр Fertility не зависит от других регрессоров линейно и может быть использован при построении модели.

Зависимость $Agriculture \sim Fertility, Catholic$. $R^2 = 16\%$ — очень низкий, значит линейной зависимости нет. Agriculture можно использовать в линейных моделях.

В регрессии $Catholic \sim Agriculture, Fertility$ значение R^2 около 25%. Параметр Catholic можно использовать в линейной регрессии.

Таким образом, заключаем, что все регрессоры линейно независимы между собой и их можно использовать при построении нашей модели.

2. Построить линейную модель зависимой переменной от указанных регрессоров. Оценить, насколько хороша модель.

Построим модель зависимости Examination от Agriculture, Catholic, Fertility. Характеристики данной модели представлены в таблице 4.

Таблица 4. Характеристики модели зависимости параметра Examination от параметров Agriculture, Catholic, Fertility в наборе данных Swiss. График построенной модели находится в приложении 3.

Параметр	Значение	Std. Error	t value	Pr(> t)	Уровень значимости
(Intercept)	43.68234	4.10586	10.639	1.27e-13	***
Agriculture	-0.16431	0.03337	-4.923	1.30e-05	***
Catholic	-0.03953	0.01919	-2.060	0.045506	*
Fertility	-0.24582	0.06274	-3.918	0.000315	***

Коэффициент детерминации R^2 равен 66%. Р-значения низкие, все коэффициенты подобраны хорошо.

Делаем вывод, что наша модель является очень хорошей.

3. Ввести в модель логарифмы регрессоров. Сравнить модели и выбрать наилучшую.

Введём в нашу модель логарифмы регрессоров. Ниже, в таблицах 5-7 приведены характеристики построенных нами моделей.

Таблица 5. Характеристики модели зависимости параметра *Examination* от параметров *Agriculture*, *Catholic*, *Fertility*, $\log(Fertility)$ в наборе данных Swiss.

Параметр	Значение	Std. Error	t value	Pr(> t)	Уровень значимости
(Intercept)	45.38047	90.34704	0.502	0.618	
Agriculture	-0.16401	0.03736	-4.390	7.5e-05	***
Catholic	-0.03979	0.02381	-1.671	0.102	
Fertility	-0.23714	0.46575	-0.509	0.613	
I(log(Fertility))	-0.54609	29.02354	-0.019	0.985	

Таблица 6. Характеристики модели зависимости параметра *Examination* от параметров *Agriculture*, *Catholic*, *Fertility*, $\log(Catholic)$ в наборе данных Swiss.

Параметр	Значение	Std. Error	t value	Pr(> t)	Уровень значимости
(Intercept)	40.85227	6.29783	6.487	7.92e-08	***
Agriculture	-0.15229	0.03921	-3.884	0.000357	***
Catholic	-0.07818	0.06767	-1.155	0.254493	
Fertility	-0.23653	0.06510	-3.633	0.000756	***
I(log(Catholic))	1.08167	1.81489	0.596	0.554376	

Таблица 7. Характеристики модели зависимости параметра *Examination* от параметров *Agriculture*, *Catholic*, *Fertility*, $\log(Agriculture)$ в наборе данных Swiss.

Параметр	Значение	Std. Error	t value	Pr(> t)	Уровень значимости
(Intercept)	48.39635	4.93681	9.803	2.02e-12	***
Agriculture	-0.06500	0.06851	-0.949	0.34813	
Catholic	-0.05095	0.02005	-2.541	0.01483	*
Fertility	-0.19742	0.06815	-2.897	0.00597	**
I(log(Agriculture))	-3.38240	2.05005	-1.650	0.10642	

Коэффициент детерминации R^2 во всех моделях приблизительно равный(66%). Сделаем вывод, что модель *Examination* ~ *Agriculture*, *Catholic*, *Fertility*, $\log(Catholic)$ является наилучшей, так как в ней большее количество

коэффициентов подсчитано с меньшей погрешностью, по сравнению с двумя другими моделями.

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

Рассмотрим следующие модели:

- 1) *Examination* ~ *Agriculture*, *Catholic*, *Fertility*, (*Agriculture***Catholic*).
- 2) *Examination* ~ *Agriculture*, *Catholic*, *Fertility*, (*Agriculture***Fertility*).
- 3) *Examination* ~ *Agriculture*, *Catholic*, *Fertility*, (*Fertility***Catholic*).
- 4) *Examination* ~ *Agriculture*, *Catholic*, *Fertility*, (*Agriculture*²).
- 5) *Examination* ~ *Agriculture*, *Catholic*, *Fertility*, (*Catholic*²).
- 6) *Examination* ~ *Agriculture*, *Catholic*, *Fertility*, (*Fertility*²).

Модель 3 является наилучшей: R^2 самый высокий (70%), уровень значимости всех переменных выше, чем у других моделей, вероятность неправильно посчитанных коэффициентов ниже. В таблице 8 приведены характеристики модели 3. С характеристиками остальных моделей можно ознакомиться в приложении 4.

Таблица 8. Характеристики модели зависимости параметра *Examination* от параметров *Agriculture*, *Catholic*, *Fertility*, (*Fertility***Catholic*) в наборе данных Swiss.

Параметр	Значение	Std. Error	t value	Pr(> t)	Уровень значимости
(Intercept)	61.921772	7.931892	7.807	1.04e-09	***
Agriculture	-0.162664	0.031297	-5.197	5.60e-06	***
Catholic	-0.413637	0.143379	-2.885	0.00615	**
Fertility	-0.514393	0.117846	-4.365	8.12e-05	***
(Fertility*Catholic)	0.005095	0.001937	2.630	0.01188	*

Заключение

В ходе решения задачи 2 убедились в отсутствии линейной зависимости между регрессорами, построили нашу модель и поэкспериментировали с добавлением новых параметров. Лучшей оказалась модель с введёнными произведениями пар регрессоров, поскольку значение R^2 было самым высоким, относительно других моделей(70%), а также уровень значимости всех регрессоров был самым высоким((*) являлся минимальным значением). Также стоит отметить, что зависимость является отрицательной по всем параметрам.

Задача 2.2

Набор данных: Swiss.

Объясняемая переменная: Examination.

Регрессоры: Agriculture, Catholic, Fertility.

1. Оценить доверительные интервалы для всех коэффициентов в модели, $p=95\%$.

Доверительный интервал для каждого параметра указывает на диапазон значений, в котором с определенной вероятностью находится истинное значение этого параметра в генеральной совокупности.

Вычислим доверительные интервалы, используя формулу: $[\beta - t \times \sigma, \beta + t \times \sigma]$, где β — вычисленное в модели значение коэффициента, t — значение t-критерия Стьюдента, σ — стандартная ошибка коэффициента в модели. Для вычислений нам понадобится критерий Стьюдента. Воспользовавшись командой `qt`, выясняем, что $t = 2.017$.

Доверительные интервалы параметров:

Свободный: [35.39, 51.97], *Agriculture*: [-0.83, 0.56], *Catholic*: [-0.77, -0.01], *Fertility* [-0.37, -0.13].

Это означает, что с вероятностью $p(95\%)$ истинное значение параметра "Свободный" будет находиться в диапазоне от 35.39 до 51.97, *Agriculture* от -0.83 до 0.56 и т.д.

2. Сделать вывод о отвержении или невозможности отвергнуть статистическую гипотезу о том, что коэффициент равен 0 с учётом данных доверительных интервалов.

Доверительный интервал параметра *Agriculture* содержит 0, это значит, что мы не можем отвергнуть гипотезу о том, что коэффициент равен 0. Следовательно, значение коэффициента *Agriculture* не является значимым для модели в данном контексте или что нам нужны дополнительные данные или более точные методы анализа для получения более точных выводов о значимости этого коэффициента.

Доверительные интервалы остальных параметров не содержат 0. Отвергаем гипотезу.

3. Оценить доверительный интервал для прогноза.

Сделаем прогноз по следующим данным: *Catholic* = 84.32, *Agriculture*=45.1, *Fertility*=83.1. С помощью команды `predict`(см. Приложение 5) получаем

интервал $[11.46277, 15.93292]$ и возможное значение 13.69785. Сравним с реальным значением 13.23152 и делаем вывод, что наш прогноз оказался удачным.

Заключение

Доверительный интервал свободного коэффициента оказался достаточно большим, однако это означает бóльшую вероятность попадания в него. Доверительный интервал Agriculture содержит 0, следовательно, данный параметр не является значимым, или нам необходимо больше данных. Доверительный интервал для нашего прогноза оказался не сильно большим, но при этом значение по прогнозу находилось в его пределах, что является положительным результатом.

Задача 3

В рамках данной задачи необходимо проанализировать данные волны мониторинга экономического положения и здоровья населения РФ.

Из набора данных необходимо взять параметры: пол, зарплата, семейное положение, наличие высшего образования, возраст, тип населённого пункта, длительность рабочей недели.

Из параметра семейное положение, сделаем дамми-переменные:

- 1) $wed1 = 1$ в случае, если респондент женат, 0 — в противном случае;
- 2) $wed2 = 1$, если респондент разведён или вдовец;
- 3) $wed3 = 1$, если респондент никогда не состоял в браке.

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

Из параметра, отвечающего типу населённого пункта, создадим одну дамми-переменную $city_status$ со значением 1 для города или областного центра, 0 — в противоположном случае.

Введём параметр $higher_educ$, характеризующий наличие полного высшего образования.

Факторные переменные, “имеющие много значений”, такие как: зарплата($wage$), длительность рабочей недели($working_hours$) и возраст(age) преобразуем в вещественные переменные и нормализуем их: вычтем среднее значение по этой переменной, разделим её значения на стандартное отклонение.

Набор данных: `r12i_os26b.csv` — данные исследования RLMS-HSE

Объясняемая переменная: $wage$

Регрессоры: sex , age , $wed1$, $wed2$, $wed3$, $higher_educ$, $city_status$, $working_hours$.

1. Построить линейную регрессию зарплаты на все параметры, которые мы выделили из данных мониторинга. Оценить коэффициент вздутия дисперсии VIF.

Построим модель $wage \sim sex, age, wed1, wed2, wed3, higher_educ, city_status, working_hours$ с помощью команды `lm` пакета `lmtest`. Данные модели и коэффициенты вздутия дисперсии приведены ниже в таблицах 1 и 2.

Таблица 1. Характеристики модели зависимости параметра *wage* от параметров *sex*, *age*, *wed1*, *wed2*, *wed3*, *higher_educ*, *city_status*, *working_hours*.

Параметр	Значение	Std. Error	t value	Pr(> t)	Уровень значимости
(Intercept)	0.003089	0.60824	0.051	0.959502	
sex	0.071031	0.034637	2.051	0.040367	*
age	-0.071925	0.018743	-3.837	0.000127	***
wed1	-0.084457	0.056176	-1.503	0.132818	
wed2	-0.024650	0.069725	-0.354	0.723715	
wed3	0.012479	0.069819	0.179	0.858157	
higher_educ	0.011345	0.039456	0.288	0.773712	
city_status	0.020885	0.038134	0.548	0.583947	
working_hours	0.100138	0.016841	5.946	3.02e-09	***

Multiple R-squared: 0.02044. Adjusted R-squared: 0.01819

Таблица 2. Коэффициенты вздутия дисперсии модели *wage ~ age, sex, wed1, wed2, wed3, higher_educ, city_status, working_hours*.

sex	1.056137	age	1.248458
wed1	2.669306	higher_educ	1.034786
wed2	2.099898	city_status	1.020696
wed3	2.105525	working_hours	1.007910

Уберём регрессоры *wed3*, *wed2*, *higher_educ*, *city_status*, поскольку у них низкий уровень значимости и плохая р-статистика. VIF всех параметров стал лучше и не превышает 1.1, а R^2 немного увеличился. Характеристики получившейся модели можем наблюдать на рисунке 1.

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	0.01513	0.03006	0.503	0.6148	
sex	0.07287	0.03403	2.141	0.0323	*
wed1	-0.07878	0.03548	-2.220	0.0265	*
age	-0.07551	0.01719	-4.393	1.15e-05	***
working_hours	0.09975	0.01682	5.931	3.31e-09	***

Рисунок 1. Характеристики модели зависимости параметра *wage* от параметров *sex*, *wed1*, *age*, *working_hours*.

2. Поэкспериментировать с функциями вещественных параметров: использовать логарифмы, степени (от 0.1 до 2 с шагом 0.1), произведения вещественных регрессоров.

Поскольку из всех регрессоров целочисленными являются только *working_hours* и *age* — логарифмы, степени и произведения будем вводить именно для этих параметров.

Логарифмы:

Введём логарифмы $\log(\text{working_hours})$ и $\log(\text{age})$. В результате получаем очень большие коэффициенты вздутия дисперсии:

sex 1.071412e+00, *wed1* 1.088968e+00, *age* 1.055581e+00,
working_hours 3.544698e+10, $\log(\text{working_hours})$ 3.544697e+10,
 $\log(\text{age})$ 1.024272e+00.

Заметим, что *working_hours* сильно зависит от логарифма, поэтому уберём его.

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	0.23497	0.04993	4.706	2.63e-06	***
<i>sex</i>	0.07376	0.03404	2.166	0.0303	*
<i>wed1</i>	-0.08429	0.03585	-2.351	0.0188	*
<i>age</i>	-0.07524	0.01719	-4.377	1.24e-05	***
$I(\log(\text{Mod}(\text{working_hours})))$	0.15866	0.02677	5.927	3.39e-09	***
$I(\log(\text{Mod}(\text{age})))$	-0.01820	0.01705	-1.068	0.2858	

Multiple R-squared: 0.02056, Adjusted R-squared: 0.01915

Рисунок 2. Характеристики модели зависимости параметра *wage* от параметров *wed1*, *age*, $\log(\text{working_hours})$, $\log(\text{age})$.

Глядя на *рисунок 2*, видим, что параметр $\log(\text{age})$ мало важен, поэтому его можно исключить из нашей модели. Значение R^2 приблизительно такое же, как у нашей модели из пункта 1. Делаем выводы, что модель построена хорошо.

Степени:

Введём переменную *current_pow* для возведения в нужную нам степень и будем её изменять по ходу решения задачи (изначально *current_pow* = 0.1).

Построим модель с введёнными параметрами степеней.

На *рисунке 3* отчётливо видим, что коэффициенты вздутия дисперсии принимают адекватные значения, R^2 также хороший для нашей модели и значимость регрессоров тоже хороша. Поэкспериментируем и будем изменять значение переменной *current_pow* от 0.1 до 2.0 с шагом 0.1 (подробнее ознакомиться со значениями каждой регрессии можно в *приложении б*). Заметим, что при увеличении степени коэффициенты вздутия дисперсии

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-1.15939	0.29761	-3.896	9.98e-05	***
sex	0.07388	0.03405	2.170	0.0301	*
wed1	-0.08468	0.03590	-2.359	0.0184	*
age	-0.07519	0.01719	-4.374	1.26e-05	***
I(Mod(working_hours)^current_pow)	1.58006	0.26662	5.926	3.40e-09	***
I(Mod(age)^current_pow)	-0.20477	0.19074	-1.074	0.2831	

	sex	wed1	age
I(Mod(working_hours)^current_pow)	1.021481	1.091285	1.051150
	1.006115	1.024398	

Multiple R-squared: 0.02056, Adjusted R-squared: 0.01916

Рисунок 3. Характеристики модели зависимости параметра *wage* от параметров *sex*, *wed1*, *age*, $working_hours^{0.1}$, $age^{0.1}$.

растут а значения R^2 уменьшаются, делаем вывод, что чем ниже степень, тем лучше наша модель. Стоит отметить, что во всех моделях присутствовал мало важный параметр $age^{current_pow}$, который можно исключить (на итог не влияет).

Произведения:

Введём произведение регрессоров — $working_hours * age$ и построим данную модель.

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	0.015519	0.030068	0.516	0.6058	
sex	0.073553	0.034061	2.159	0.0309	*
wed1	-0.079460	0.035504	-2.238	0.0253	*
age	-0.075206	0.017199	-4.373	1.26e-05	***
working_hours	0.101020	0.016979	5.950	2.95e-09	***
I(working_hours * age)	0.009784	0.017791	0.550	0.5824	

sex	wed1	age	working_hours	I(working_hours * age)
1.022068	1.066995	1.051916	1.025183	1.022409

Multiple R-squared: 0.02032, Adjusted R-squared: 0.01892

Рисунок 4. Характеристики модели зависимости параметра *wage* от параметров *sex*, *wed1*, *age*, *working_hours*, $working_hours * age$.

На рисунке 4 представлены характеристики построенной нами модели. Видим, что R^2 упал по сравнению с первоначальной моделью, VIF параметров отличный, однако значимость произведения низкая. Делаем вывод, что построенная нами модель не является удачной.

3. Выделить наилучшие модели из построенных.

Исходя из пункта 2, видим, что лучшими моделями оказались модель с логарифмами, степенями 0.1 и 0.2 (характеристики данной модели можем увидеть ниже на *рисунке 5*), поскольку все эти модели лучше первоначальной по коэффициентам вздутия дисперсии, уровню значимости регрессоров и разбросу R^2 .

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-0.47106	0.14375	-3.277	0.00106	**
sex	0.07399	0.03405	2.173	0.02985	*
wed1	-0.08497	0.03595	-2.364	0.01816	*
age	-0.07513	0.01719	-4.370	1.28e-05	***
I(Mod(working_hours)^current_pow)	0.78046	0.13171	5.926	3.42e-09	***
I(Mod(age)^current_pow)	-0.11217	0.10513	-1.067	0.28607	

Multiple R-squared: 0.02056, Adjusted R-squared: 0.01915

	sex	wed1	age
I(Mod(working_hours)^current_pow)	1.021674	1.094188	1.051281
	1.006120	1.027122	

Рисунок 5. Характеристики модели зависимости параметра *wage* от параметров *sex*, *wed1*, *age*, *working_hours* ^ 0.2, *age* ^ 0.2.

4. Сделать вывод о том, какие индивиды получают наибольшую зарплату.

Исходя из построенных нами моделей, заметим сильную положительную зависимость от количества рабочих часов в неделю, отрицательную зависимость от возраста индивида, принадлежности индивида к мужскому полу и не состоящего в браке.

Итог: большую зарплату получают молодые неженатые мужчины, много работающие.

5. Оцените лучшие модели для подмножеств индивидов (1. Не вступавшие в брак, без высшего образования; 2. Городские жители, состоящие в браке). Сделайте вывод о том, какие индивиды получают наибольшую зарплату.

Воспользуемся лучшей нашей моделью со степенью 0.1 и рассмотрим подмножество индивидов, не вступавших в брак и без высшего образования. Отбросим ненужные нам признаки и построим модель. Исходя из характеристик данной модели, представленных на *рисунке 6*, делаем вывод, что большую зарплату из данного подмножества получают индивиды, много работающие. Теперь возьмём подмножество индивидов, живущих в городах и не состоящих в браке. Снова воспользуемся нашей лучшей моделью. Исходя из данных на *рисунке 7*, делаем вывод, что в этом подмножестве большую зарплату получают молодые индивиды, много работающие.

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-2.29307	1.28465	-1.785	0.075100	.
sex	0.01813	0.12065	0.150	0.880663	
age	-0.03307	0.10226	-0.323	0.746610	
city_status	0.18921	0.13287	1.424	0.155296	
I(Mod(working_hours)^current_pow)	3.11862	0.83181	3.749	0.000206	***
I(Mod(age)^current_pow)	-0.48169	1.10014	-0.438	0.661761	

Рисунок 6. Характеристики модели зависимости параметра wage от параметров sex, age, city_status, working_hours ^ 0.1, age ^ 0.1.

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-0.97590	0.43516	-2.243	0.025066	*
sex	0.03806	0.04830	0.788	0.430846	
age	-0.06652	0.02674	-2.487	0.012977	*
I(Mod(working_hours)^current_pow)	1.41859	0.39931	3.553	0.000393	***
I(Mod(age)^current_pow)	-0.31472	0.27797	-1.132	0.257718	

Рисунок 7. Характеристики модели зависимости параметра wage от параметров sex, age, working_hours ^ 0.1, age ^ 0.1.

Заключение

С помощью построенных нам моделей легко оценить множество индивидов, получающих бóльшую зарплату. Наша модель показывает, что это молодые(отрицательная зависимость от возраста, средний приоритет) неженатые мужчины, много работающие(положительная зависимость, высокий приоритет). Экспериментирование с добавлением степеней, логарифмов и произведений даёт нам отличную(по отношению к остальным) модель с параметром степени(0.1) и относительно неплохую с логарифмами. Такие выводы мы делаем опираясь на коэффициенты детерминации, коэффициенты вздутия дисперсии, а также уровню значимости регрессоров.

Задача 4

Набор данных: Students performance in exams.

Классификатор: SVM.

Целевой параметр: Writing Score(выше среднего значения — класс 0, ниже или совпадает — класс 1)

1. Обработать набор данных. Выделить целевой признак и удалить его из данных, на основе которых будет обучаться классификатор. Разделить набор данных на тестовую и обучающую выборку. Построить классификатор типа SVM для задачи классификации по целевому параметру. Оценить точность построенного классификатора с помощью метрик precision, recall и F1 на тестовой выборке.

Для начала выделим необходимые нам параметры из набора данных, а именно: *gender, level of education, writing score, reading score, math score*. Преобразуем их в целочисленные переменные с помощью *one-hot-encoding*:

- gender равен 0, если личность женщина, и равен 1 в случае, когда личность мужчина;
- level of education: 1 — Бакалавриат, 2 — Магистратура, 3 — Высшее среднее, 4 — Колледж, 5 — другие.

Теперь выделяем целевой признак *writing_score_above_avg*(выше среднего значения по *writing score* — 0, ниже или совпадает — класс 1).

Далее нам необходимо разделить наши данные на тестовую и обучающую выборки. Для обучающей выборки мы выделим 70% наших данных. Для решения задачи классификации создадим классификатор типа SVM, с параметром *linear*(линейное представление гиперплоскости). Затем обучаем наш классификатор, делаем прогноз и оцениваем точность с помощью метрик *accurasy, precision, recall, f1*. Результаты метрик представлены ниже в таблице 1.

Таблица 1. Результаты оценки качества классификатора SVM с помощью метрик *accurasy, precision, recall* и *f1*.

Метрика	Точность
accurasy	0.9066666666666668
precision	0.9185974945533768
recall	0.9

f1	0.9063845372228225
----	--------------------

Исходя из данных в таблице 1, делаем вывод о том, что наш классификатор успешно обучился, так как показал высокие результаты(около 90% точности) классификации на нашей тестовой выборке. Код, решающий пункт 1 предоставлен в приложении 7.

2. Построение классификатора типа Случайный Лес (Random Forest) для решения той же задачи классификации. Оценка его качества с помощью метрик precision, recall и F1 на тестовой выборке. С помощью GridSearch перебрать различные комбинации гиперпараметров: на первой итерации задать большие шаги (50 или 100) по числу деревьев n_estimators. На следующих итерациях определить лучшее количество деревьев n_estimators с точностью до 10. Выбрать лучший классификатор.

Снова разделим данные на тестовую и обучающие выборки, но поэкспериментируем с количеством обучающих данных, сделав его 80%. Далее создадим и построим классификатор типа Случайный Лес с количеством деревьев равным 100. Оценка качества классификации с помощью тех же метрик представлена в таблице 2.

Таблица 2. . Результаты оценки качества классификатора Random Forest с помощью метрик accuracy, precision, recall и f1.

Метрика	Точность
accuracy	0.8966666666666667
precision	0.9037908029843514
recall	0.8933333333333333
f1	0.896499688318275

Классификатор типа Random Forest справился с задачей тоже на достаточно высоком уровне, однако стоит подметить, что его построение заняло намного больше времени. Построение классификатора находится в приложении 8.

Теперь воспользуемся перебором по сетке параметров GridSearch для классификатора типа Random Forest. Определим некоторый набор параметров(сетку), по которому будем делать перебор:

- n_estimators(количество деревьев) [50, 100, 150, 200];
- max_depth(максимальная глубина дерева) [None, 5, 10, 15];

- `min_samples_split`(минимальное количество образцов (сэмплов), необходимое для разделения внутреннего узла дерева) [2, 5, 10];
- `min_samples_leaf`(минимальное количество образцов (сэмплов), необходимое для формирования листового узла дерева) [1, 2, 4];
- `max_features`(количество признаков, которые следует учитывать при поиске наилучшего разделения в каждом узле) [“sqrt”, “log2”].

Теперь создаём экземпляр класса `GridSearch`(см. Приложение 9) и прогоняем его по данным параметрам. На выходе получаем лучшие параметры для построения нашего классификатора типа Случайны Лес:

```
{'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 150}.
```

Теперь сделаем итерацию по количеству деревьев от 150 до 300 с шагом 50 и 10. После перебора данных параметров на выходе получаем, что лучшим количеством деревьев в обоих случаях оказалось 200(см. Приложение 10).

В конечном итоге классификатор типа `SVM` справился с задачей немного лучше, чем `Random Forest`, при этом затратив меньше времени на обучение.

Заключение

Построенные классификаторы типа SVM и Random Forest отлично справились с поставленной задачей классификации по целевому параметру `writing_score` (точность не менее 89%). Классификатор SVM справился с задачей немного лучше (такие выводы мы делаем, опираясь на результаты метрик оценки `accuracy`, `precision`, `recall` и `f1`) и быстрее (поскольку построение Random Forest занимает большее количество времени, в связи с перебором различных вариантов классификации). Для наиболее эффективного использования классификатора типа Random Forest значение гиперпараметра `n_estimators` (количество деревьев в ансамбле) должно равняться 200.

Задача 5

Тема: Эффективная программная реализация вычисления сплайнов Коханек-Бартельса.

1. Определение сплайна и его параметров.

Для начала, давайте введем определение сплайна.

Определение: Сплайн — это функция, кусочно определяемая полиномами.

Определение: Сплайн Эрмита — это сплайн, каждая часть которого задается полиномами третьей степени.

Определение: Сплайн Коханек-Бартельса — это сплайн Эрмита с тремя параметрами: натяжение, смещение и непрерывность.

Одно из основных преимуществ сплайна Коханек-Бартельса — это возможность создать гладкую прямую, которая будет проходить через опорные точки, такая кривая создается путем подбора параметров. Параметрами сплайна можно контролировать внешний вид сплайна и его поведение.

Разберем каждый параметр отдельно:

Tension(t):

Tension или же непрерывность, отвечает за форму кривой сплайна, которая может быть более или менее «напряженной», он определяет то, с каким как будет изгибаться сплайн при переходе между опорными точками, при разных значениях параметра t переходы между опорными точкам будут либо более резкими, либо более плавными. Выбор оптимального значения Tension зависит от конкретной задачи и требуемого эффекта. Например, если нужно создать более органичную форму, то значение Tension можно выбрать более низким, чтобы изгибы сплайна были более плавными и естественными. Если же нужно подчеркнуть геометрические формы, то можно выбрать более высокое значение Tension, чтобы изгибы были более резкими и угловатыми.

Bias(b):

Bias или же смещение, определяет насколько смещен относительно опорных точек. Значение Bias может быть использовано для создания различных эффектов при построении сплайнов. Например, если значение параметра Bias выбрать близким к нулю, то сплайн будет проходить близко к опорным точкам, и переходы между ними будут плавными и естественными. Если же значение параметра Bias будет выше нуля или меньше нуля, то сплайн будет

сильнее смещаться в сторону следующей или предыдущей опорной точки, что создаст более резкие переходы.

Continuity(c):

Continuity или же непрерывность, определяет то, насколько гладко и плавно сплайн переходит от одной опорной точки к другой или то, насколько гладкой будет кривая сплайна. На рисунке 1 видно то, как каждый параметр изменяет внешний вид сплайна. Кроме того, параметр Continuity может быть использован для создания сплайнов разной степени гладкости и кривизны. Например, если выбрать непрерывность первого порядка (C1), то сплайн будет иметь плавные переходы между опорными точками, а кривизна будет определяться касательными к опорным точкам. Если же выбрать более высокую непрерывность, например, непрерывность второго порядка (C2), то сплайн будет еще более гладким и кривизна будет более однородной на всей длине сплайна. Выбор оптимального значения Continuity зависит от конкретной задачи и требуемого эффекта при построении сплайна.

Немного о порядках непрерывности:

Порядки непрерывности - это способ классификации сплайнов по степени их гладкости на переходах между опорными точками. Обычно различают три порядка непрерывности: первый, второй и третий.

Непрерывность первого порядка (C1) означает, что кривая сплайна имеет плавные переходы между опорными точками, а её касательные на этих точках совпадают. Это может использоваться для создания плавных поверхностей и кривых объектов.

Непрерывность второго порядка (C2) означает, что помимо плавных переходов и совпадения касательных на опорных точках, кривая сплайна также имеет плавные переходы вторых производных. Это может использоваться для создания более сложных поверхностей и объектов, например, для моделирования автомобильных кузовов или корпусов самолетов.

Непрерывность третьего порядка (C3) означает, что помимо плавных переходов, совпадения касательных и плавных переходов вторых производных, кривая сплайна также имеет плавные переходы третьих производных. Это может использоваться для создания ещё более сложных объектов, например, для моделирования гладких поверхностей в 3D-графике или при создании анимации.

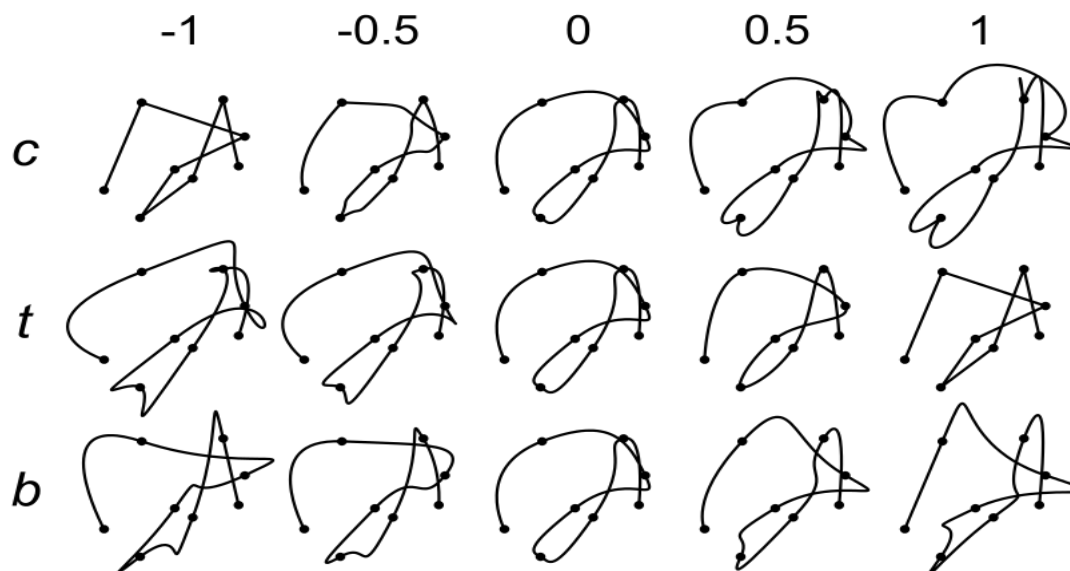


Рисунок 1^[4]. Влияние параметров на внешний вид сплайна.

2. Построение сплайновой кривой.

Для начала нам нужно будет посчитать касательные в каждой опорной точке.

Пусть задано $n + 1$ узлов, где $i \in [1, n - 1]$, тогда для интерполяции кривой Эрмита у нас есть начальная точка p_i и конечная точка p_{i+1} , где начальная касательная это d_i , а конечная касательная это d_{i+1} , тогда вычисление касательных будем осуществлять по следующим формулам:

$$d_i = \frac{(1-t)(1+b)(1+c)}{2}(p_i - p_{i-1}) + \frac{(1-t)(1-b)(1-c)}{2}(p_{i+1} - p_i)$$

$$d_{i+1} = \frac{(1-t)(1+b)(1+c)}{2}(p_{i+1} - p_i) + \frac{(1-t)(1-b)(1+c)}{2}(p_{i+2} - p_{i+1})$$

Так же для интерполяционного многочлена нам понадобятся базисные полиномы Эрмита, которые позволяют получать более гладкие и точные результаты интерполяции и описывать форму сплайна на каждом сегменте:

Таблица 1. Базисные полиномы Эрмита.

	Значение полинома
$h_{00}(t)$	$2t^3 - 3t^2 + 1$
$h_{10}(t)$	$t^3 - 2t^2 + t$
$h_{01}(t)$	$-2t^3 + 3t^2$
$h_{11}(t)$	$t^3 - t^2$

Сам же интерполяционный многочлен имеет следующий вид:

$$p(t) = h_{00}p_k + h_{10}(x_{k+1} - x_k)m_k + h_{01}p_{k+1} + h_{11}m_{k+1},$$

где p_k и p_{k+1} — начальная и конечная точка, а m_k и m_{k+1} — касательные в начальной и конечной точке.

Параметризацию аргумента сплайна будем осуществлять с помощью нормированной параметризации, то есть, когда значение аргумента сплайна переводим в произвольное значение на $[0,1]$, тогда t будет выглядеть следующим образом:

$$t = \frac{x - x_k}{x_{k+1} - x_k},$$

где x — значение аргумента сплайна, а x_k и x_{k+1} — соответствующие значения опорным точкам сплайна.

3. Алгоритм построения сплайновой кривой.

Построение сплайновой кривой будем реализовывать по следующему алгоритму:

1. Считаем значение параметра t для каждой пары точек p_i и p_{i+1} .
2. Вычисляем значения базисных полиномов Эрмита для параметров t .
3. Делаем подсчёт касательных к точкам.
4. Считаем значение функции $p(t)$.

4. Реализация.

Решать данную задачу будем на языке C++. В ходе решения использовалась библиотека `freeglut` для визуализации сплайнов.

Создадим класс *Kochanek_Bartels_Spline* со следующими полями:

- `int num_of_points;` — количество точек;
- `std::vector<std::tuple<float, float, float>> tbc;` — Вектор из наборов параметров `tension`, `bias` и `continuity`;
- `std::vector<Point> points;` — вектор точек;
- `std::vector<Point> tangent;` — вектор касательных.

В данном случае *Point* — класс, описывающий наши точки. В нём хранится размерность пространства, в которой находится точка и соответствующие ей координаты.

Добавим в наш класс базисные функции Эрмита, поскольку они необходимы нам для дальнейших вычислений:

```
float h00(const float x) const
{
    return 2 * x * x * x - 3 * x * x + 1;
}

float h10(const float x) const
{
    return x * x * x - 2 * x * x + x;
}

float h01(const float x) const
{
    return -2 * x * x * x + 3 * x * x;
}

float h11(const float x) const
{
    return x * x * x - x * x;
}
```

Листинг 1. Базисные функции Эрмита.

Добавим метод вычисления касательной типа `void`. Вместо того, чтобы возвращать, мы просто будем изменять вектор касательных и записывать в него посчитанные нами касательные. Также нам понадобится функция для вычисления сплайна.

Заключительный и главный метод — изображение сплайна. Внутри него мы считаем наши касательные, а затем проверяем то, в каком пространстве мы находимся. В случаях 2-мерного и 3-мерного пространств рисуем наши сплайны: определяем направление сплайна и, в зависимости от него задаём наши функции. Делаем подсчёт и изображаем сплайны.

Если же мы работаем в пространстве $L: \dim L > 3$, тогда из-за невозможности изобразить, просто выводим подсчитанные нами функции между точками.

Теперь наш класс полностью готов. Теперь пропишем метод, который будет изменять положение камеры:

```
void specialKeyboard(const int key, const int x, const int y)
{
    switch (key)
    {
        case GLUT_KEY_UP:
            rotate_x -= 5; break;
        case GLUT_KEY_DOWN:
            rotate_x += 5; break;
        case GLUT_KEY_LEFT:
            rotate_y += 5; break;
        case GLUT_KEY_RIGHT:
            rotate_y -= 5; break;
        case GLUT_KEY_PAGE_UP:
            zoom_x -= 0.03; break;
        case GLUT_KEY_PAGE_DOWN:
            zoom_x += 0.03; break;
    }
}
```

```

    }
    glutPostRedisplay(); // Перерисовываем
}

```

Листинг 2. Функция взаимодействия с камерой.

Также для удобства нам понадобится начертить координатные оси:

```

void draw_coordinate_Oxyz()
{
    // x
    glColor3f(1.0f, 0.0f, 0.0f); // Красный
    glBegin(GL_LINES);
    glVertex3f(-10.0f, 0.0f, 0.0f);
    glVertex3f(10.0f, 0.0f, 0.0f);
    glEnd();

    // y
    glColor3f(0.0f, 1.0f, 0.0f); // Зелёный
    glBegin(GL_LINES);
    glVertex3f(0.0f, -10.0f, 0.0f);
    glVertex3f(0.0f, 10.0f, 0.0f);
    glEnd();

    // z
    glColor3f(0.0f, 0.0f, 1.0f); // Синий
    glBegin(GL_LINES);
    glVertex3f(0.0f, 0.0f, -10.0f);
    glVertex3f(0.0f, 0.0f, 10.0f);
    glEnd();
}

};

```

Листинг 3. Функция для изображения координатных осей Oxyz.

Осталось написать функцию, отвечающую за отображение. В ней мы будем создавать сплайн исходя из заданных нами точек и параметров разбиения, устанавливать положение камеры и матрицы проекции для ориентации объектов в пространстве:

```

void display()
{
    Kochanek_Bartels_spline test_spline = Kochanek_Bartels_spline(point_spline.size(), point_spline, 0.5, -1.0, 0.0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-1, 1, -1, 1, -10, 10);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(cameraPosX, cameraPosY, cameraDistance, cameraPosX, cameraPosY, 0.0, 0.0, 1.0, 0.0);
    glScalef(zoom_x, zoom_x, 0);
    glRotatef(rotate_x, 1.0, 0.0, 0.0);
    glRotatef(rotate_y, 0.0, 1.0, 0.0);
    glEnable(GL_MAP1_VERTEX_3);
    test_spline.draw_spline();
    test_spline.draw_coordinate_Oxyz();
    glutSwapBuffers();
}

```

Листинг 4. Функция, отвечающая за отображение.

В главной функции пользователь может вводить размерность пространства, количество и координаты всех точек, количество частей разбиения сплайна и параметры *tension*, *bias*, *continuity* для каждой части разбиения. Также в главной функции происходит указание на все прописанные нами функции, необходимые для отображения.

С полным кодом можно ознакомиться в приложении 1.

5. Тестовый пример.

В качестве теста возьмём 8 точек трёхмерного пространства. Их координаты представлены в таблице 2.

Таблица 2. Координаты точек трёхмерного пространства в тестовом варианте.

Точка	x	y	z		Точка	x	y	z
<i>x1</i>	-3.5	0.0	-2.0		<i>x5</i>	3.6	0.0	2.0
<i>x2</i>	-1.0	1.5	3.5		<i>x6</i>	1.0	-1.5	1.0
<i>x3</i>	0.42	0.1	1.0		<i>x7</i>	0.12	-0.1	-1.0
<i>x4</i>	1.2	1.5	-2.5		<i>x8</i>	-1.69	-1.5	2.0

Сплайн разбивать не будем и проверим картину при базовых значениях *tension*, *bias*, *continuity* равных нулю.

Результат выполнения программы можем видеть на рисунках 2-4.

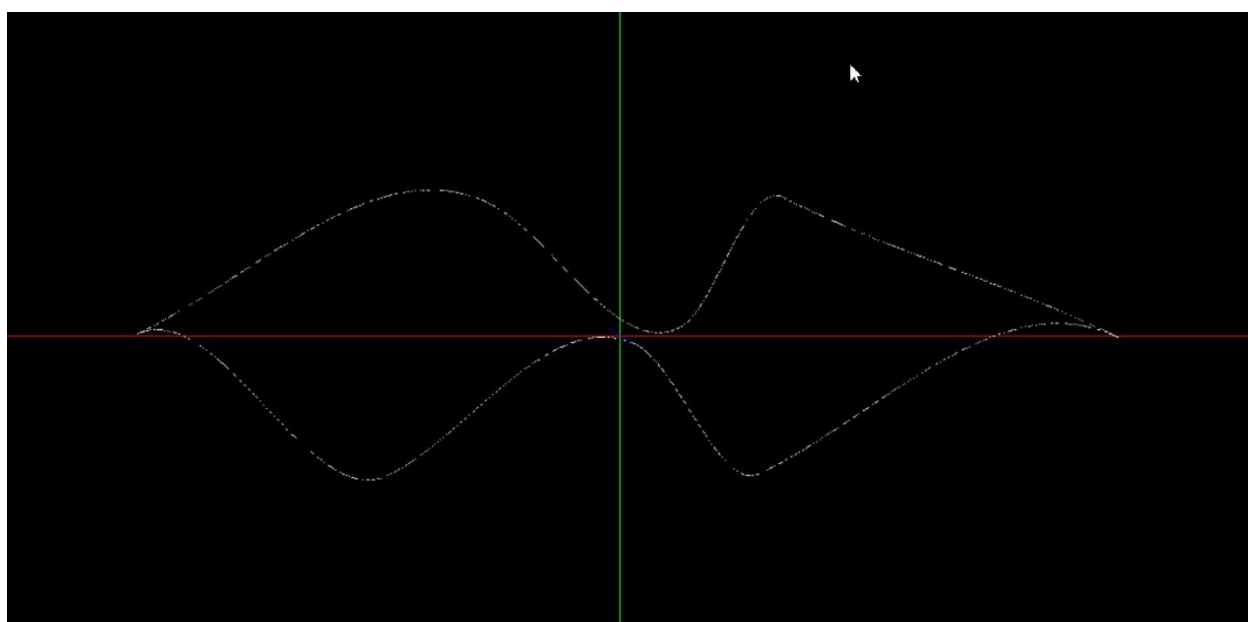


Рисунок 2. Проекция тестового сплайна на ось Oyz

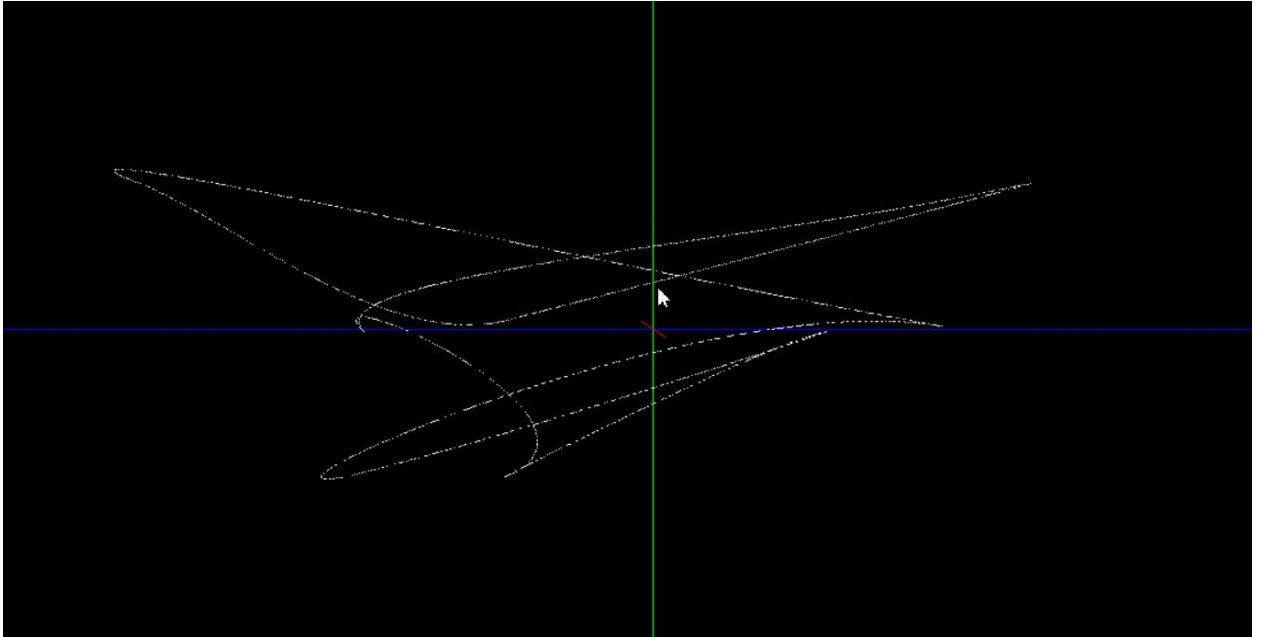


Рисунок 3. Проекция тестового сплайна на ось Oxz

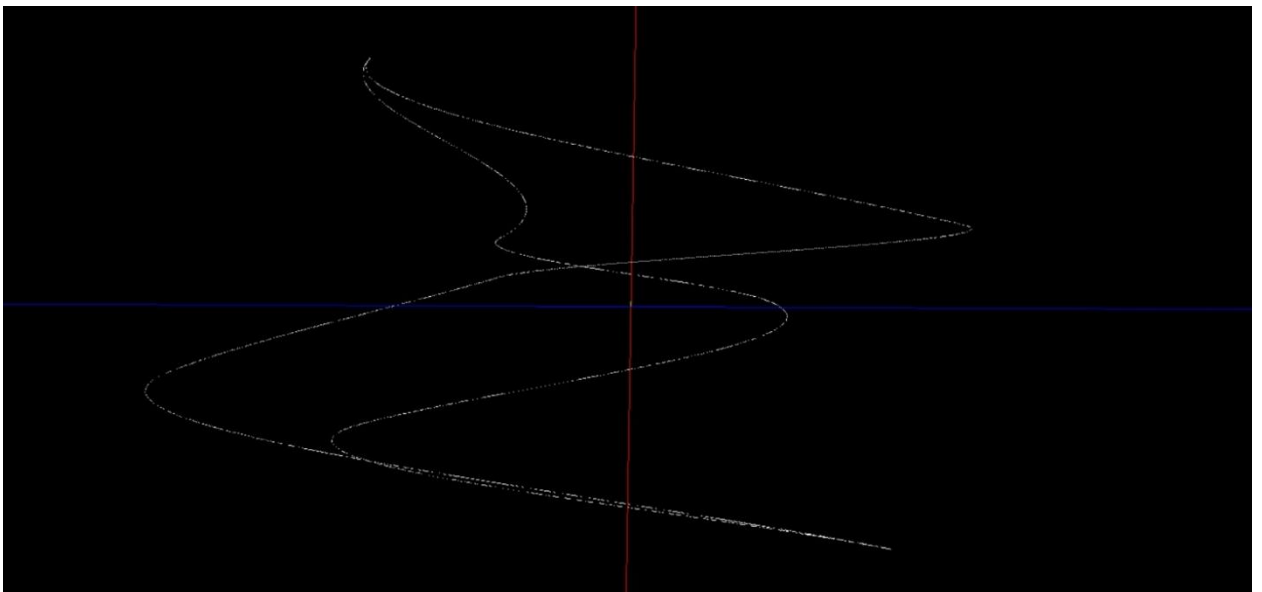


Рисунок 4. Проекция тестового сплайна на ось Oxy

Теперь посмотрим на то, как каждый параметр влияет на наш сплайн. По отдельности будем изменять их от -1.0 до 1.0 с шагом 0.5.

Результаты при изменении параметра *tension* наблюдаем на рисунках 5-8.

Итоги изменения параметра *bias* можно увидеть на рисунках 9-12.

Поведение сплайна при изменении параметра *continuity* видим на рисунках 13-16.

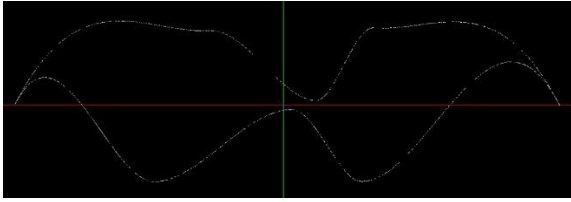


Рисунок 5. $t = -1.0$, $b = 0.0$, $c = 0.0$

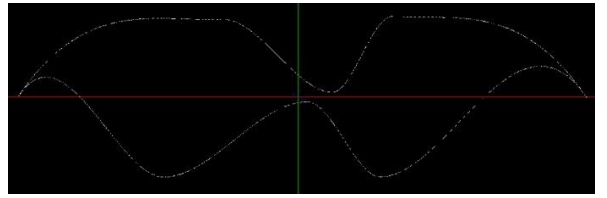


Рисунок 6. $t = -0.5$, $b = 0.0$, $c = 0.0$

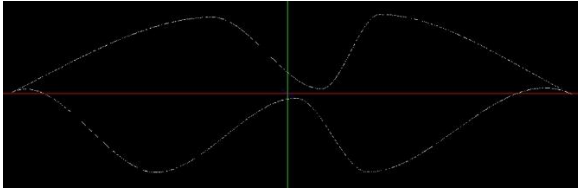


Рисунок 7. $t = 0.5$, $b = 0.0$, $c = 0.0$

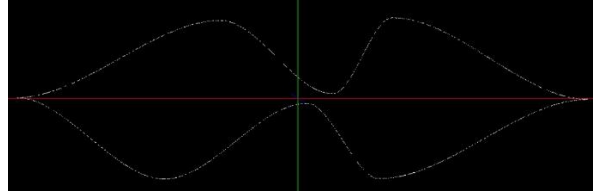


Рисунок 8. $t = 1.0$, $b = 0.0$, $c = 0.0$

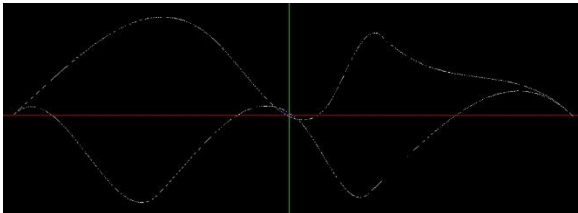


Рисунок 9. $t = 0.0$, $b = -1.0$, $c = 0.0$

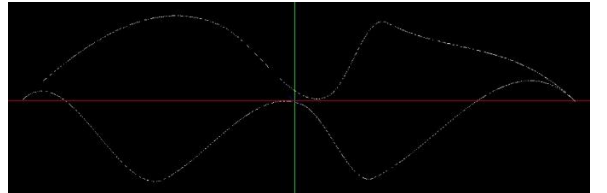


Рисунок 10. $t = 0.0$, $b = -0.5$, $c = 0.0$

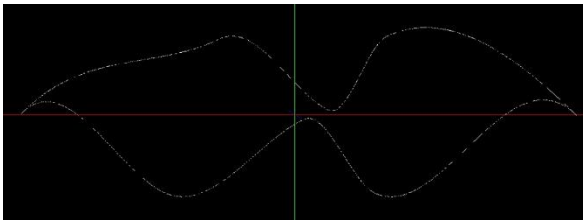


Рисунок 11. $t = 0.0$, $b = 0.5$, $c = 0.0$

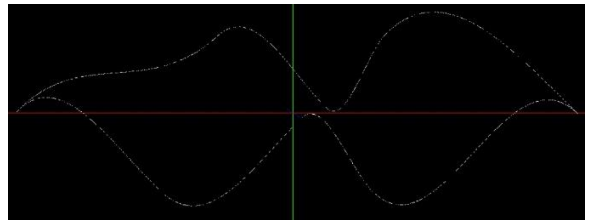


Рисунок 12. $t = 0.0$, $b = 1.0$, $c = 0.0$

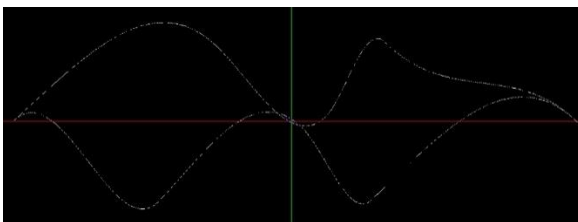


Рисунок 13. $t = 0.0$, $b = 0.0$, $c = -1.0$

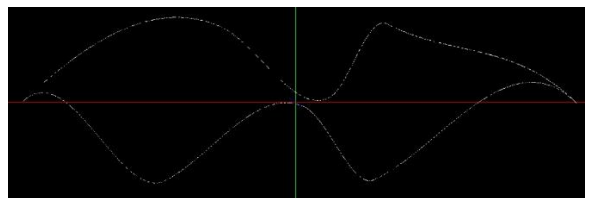


Рисунок 14. $t = 0.0$, $b = 0.0$, $c = -0.5$

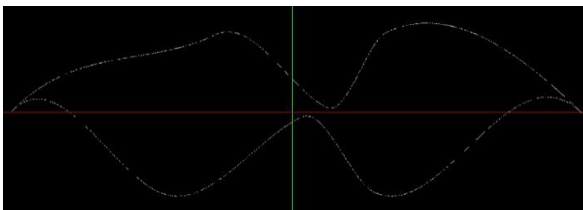


Рисунок 15. $t = 0.0$, $b = 0.0$, $c = 0.5$

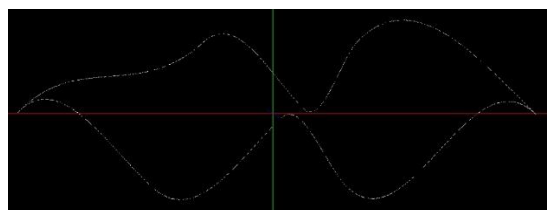


Рисунок 16. $t = 0.0$, $b = 0.0$, $c = 1.0$

Теперь поэкспериментируем с различными параметрами tension, bias и continuity. Ниже на рисунках 17-19 представлены довольно интересные изображения при заданных параметрах.

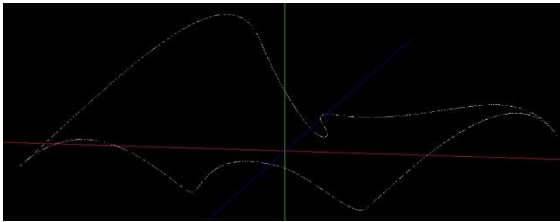


Рисунок 17. $t = 0.5$, $b = -1.0$, $c = -1.0$

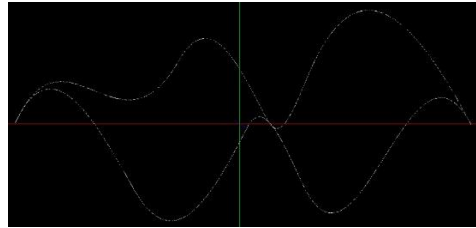


Рисунок 18. $t = 0.0$, $b = 1.0$, $c = 1.0$

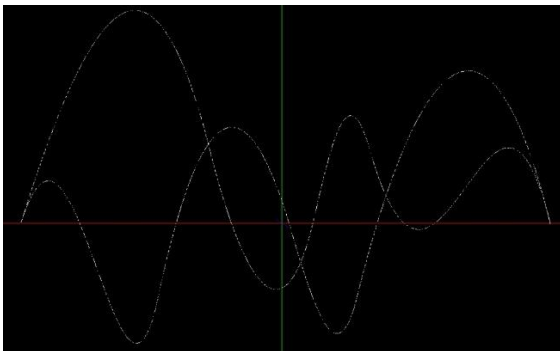


Рисунок 19. $t = -1.0$, $b = -1.0$, $c = -1.0$

Теперь попробуем разбить наш сплайн на части и посмотреть на результат. В первом случае(рисунок 20), разбиение происходит на 4 части. Во втором случае(рисунок 21) наше разбиение состоит из 8 частей, что делает каждую часть сплайна уникальной.

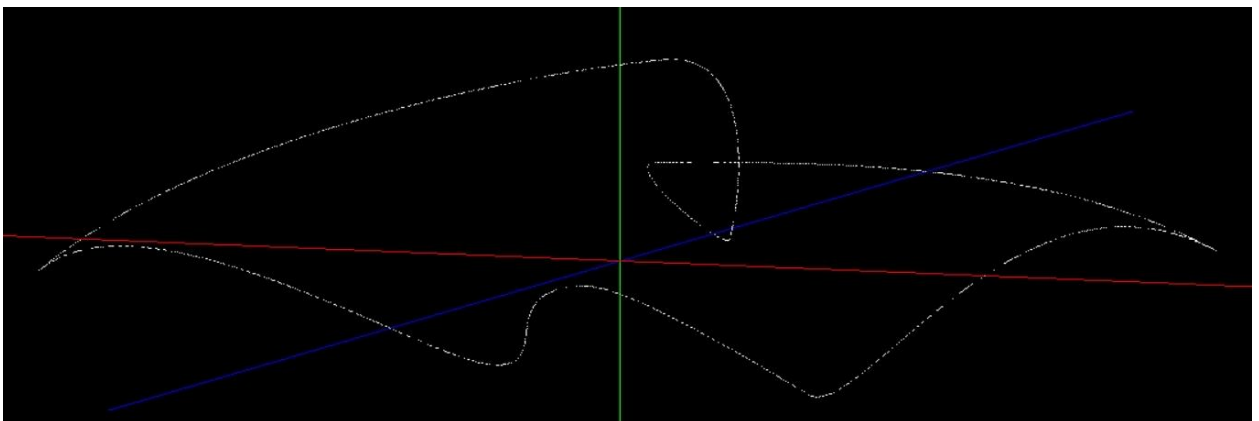


Рисунок 20. $t_1 = -1.0$, $b_1 = -1.0$, $c_1 = -1.0$; $t_2 = 0.0$, $b_2 = 0.0$, $c_2 = 0.0$; $t_3 = 0.5$, $b_3 = -0.5$, $c_3 = 0.0$; $t_4 = 1.0$, $b_4 = 0.5$, $c_4 = -1.0$

Заметим то, как наш сплайн поменял свою форму в обоих случаях. Разбиение на части помогает решать задачи с помощью сплайна более универсально, с отдельными параметрами под каждые две опорные точки.

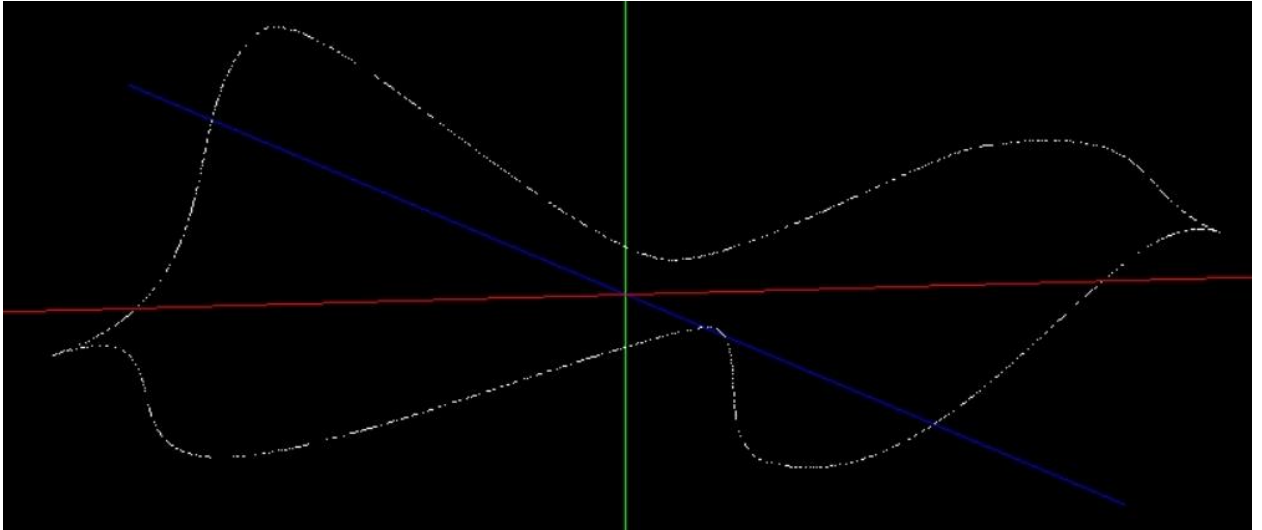


Рисунок 21. $t_1 = 0.5$, $b_1 = 0.65$, $c_1 = -0.15$; $t_2 = 0.09$, $b_2 = 0.91$, $c_2 = 0.0$; $t_3 = -1.0$, $b_3 = -0.5$, $c_3 = -0.1$; $t_4 = 0.82$, $b_4 = -0.9$, $c_4 = -0.1$; $t_5 = -0.1$, $b_5 = 0.1$, $c_5 = 0.1$; $t_6 = 0.5$, $b_6 = 0.6$, $c_6 = 0.7$; $t_7 = -0.5$, $b_7 = -0.6$, $c_7 = -0.7$; $t_8 = 0.9$, $b_8 = 1.0$, $c_8 = -0.6$

Заключение

Было предложено и реализовано решение задачи по вычислению и построению сплайнов Коханек-Бартельса. Асимптотика алгоритма выражена линейной зависимостью от количества опорных точек и размерности пространства. Погрешность отображения сплайнов зависит от выбранного разбиения сплайна и компьютерной погрешности. Мы убедились, что с помощью сплайнов Коханек-Бартельса можно строить необходимые нам кривые, изменяя параметры сплайна, благодаря этому, мы можем применять сплайны для решения универсальных задач, таких как задачи интерполяции и сглаживания.

Приложения

Приложение 1. Подсчёт Среднего значения, дисперсии и СКО для параметров Education, Fertility и Examination в наборе данных swiss.

```
library("lmtest")

data = swiss

data
summary(data)

#Считаем Среднее значение для Education
sum(data$Education)
data$Education
sum(data$Education)/47

#Дисперсия для Education
var(data$Education)

#СКО для Education
sd(data$Education)

#Считаем Среднее значение для Fertility
mean(data$Fertility)

#Дисперсия для Fertility
var(data$Fertility)

#СКО для Fertility
sd(data$Fertility)

#Считаем Среднее значение для Examination
mean(data$Examination)

#Дисперсию для Examination
var(data$Examination)

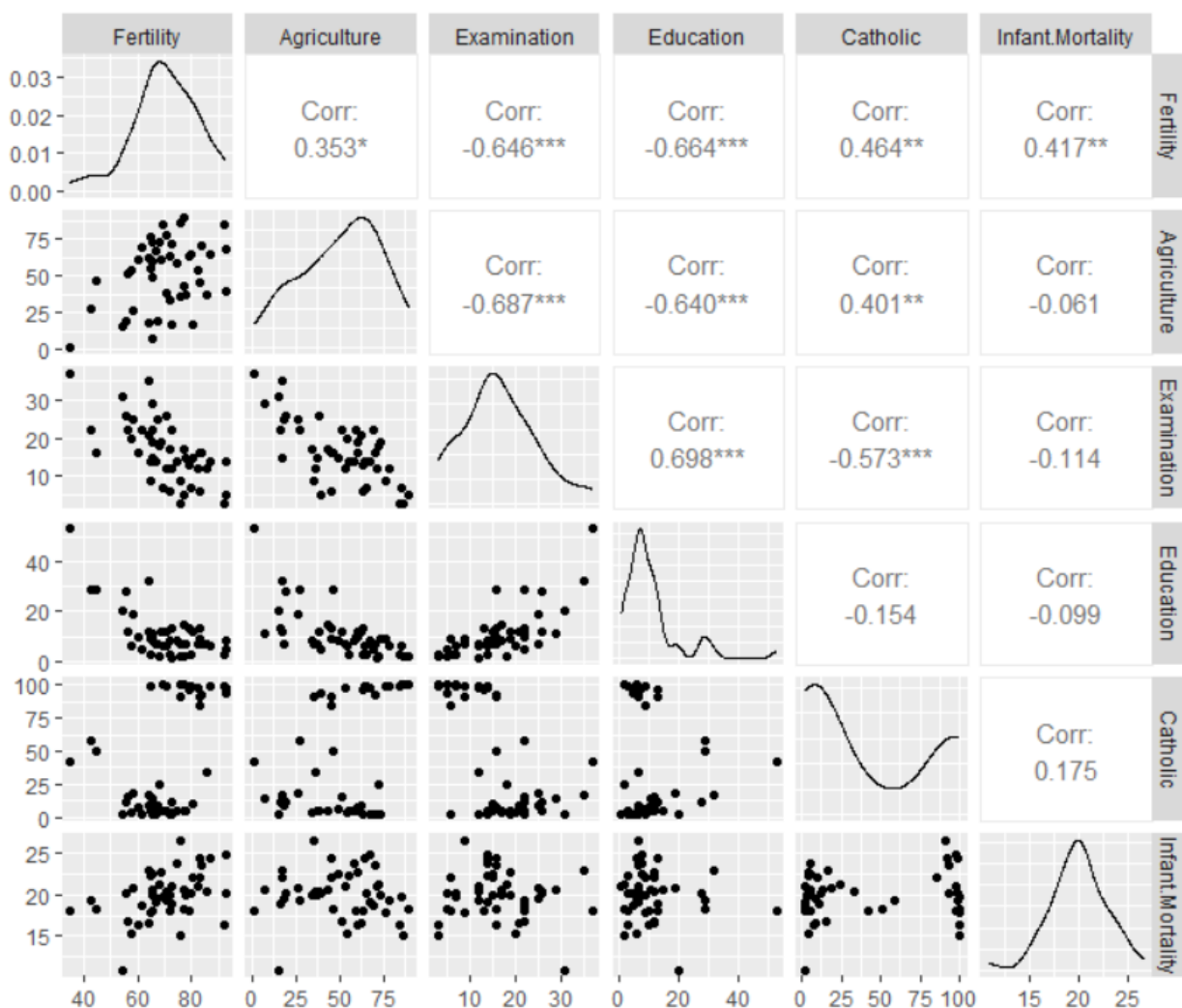
#СКО для Examination
sd(data$Examination)
```

Приложение 2. Построение линейной регрессии Education ~ Fertility в наборе данных swiss.

```
#Зависимость  $y=a+bx$ ,  $y=\text{Education}$ ,  $x=\text{Fertility}$   
model=lm(Education~Fertility,data)  
model  
summary(model)
```

```
#Зависимость  $y=a+bx$ ,  $y=\text{Education}$ ,  $x=\text{Examination}$   
model2=lm(Education~Examination,data)  
model2  
summary(model2)
```

Приложение 3. Графическое представление регрессии Examination ~ Agriculture, Catholic, Fertility в наборе данных Swiss.



Приложение 4. Характеристики моделей со всеми парами произведений регрессоров в наборе данных swiss.

*Examination ~ Agriculture, Catholic, Fertility, (Agriculture*Catholic).*

Параметр	Значение	Std. Error	t value	Pr(> t)	Уровень значимости
(Intercept)	45.6953850	4.4949641	10.166	6.85e-13	***
Agriculture	-0.2010356	0.0474032	-4.241	0.000120	***
Catholic	-0.0979916	0.0570100	-1.719	0.093006	.
Fertility	-0.2477974	0.0626272	-3.957	0.000287	***
(Agriculture*Catholic)	0.0009736	0.0008943	1.089	0.282494	

*Examination ~ Agriculture, Catholic, Fertility, (Agriculture*Fertility).*

Параметр	Значение	Std. Error	t value	Pr(> t)	Уровень значимости
(Intercept)	57.015040	7.197575	7.921	7.21e-10	***
Agriculture	-0.527662	0.167379	-3.152	0.002985	**
Catholic	-0.062020	0.021004	-2.953	0.005139	**
Fertility	-0.445613	0.108492	-4.107	0.000181	***
(Agriculture*Fertility)	0.005481	0.002478	2.212	0.032496	*

Examination ~ Agriculture, Catholic, Fertility, (Agriculture^2).

Параметр	Значение	Std. Error	t value	Pr(> t)	Уровень значимости
(Intercept)	45.219792	4.229486	10.692	1.47e-13	***
Agriculture	-0.332513	0.130428	-2.549	0.01453	*
Catholic	-0.048460	0.020165	-2.403	0.02075	*
Fertility	-0.220215	0.065075	-3.384	0.00156	**
(Agriculture^2)	0.001809	0.001357	1.333	0.18965	

Examination ~ Agriculture, Catholic, Fertility, (Catholic^2).

Параметр	Значение	Std. Error	t value	Pr(> t)	Уровень значимости
(Intercept)	45.9567120	5.7886216	7.939	6.81e-10	***
Agriculture	-0.1764261	0.0399586	-4.415	6.93e-05	***
Catholic	-0.1147372	0.1352088	-0.849	0.400919	

Fertility	-0.2632154	0.0704069	-3.738	0.000554	***
(Catholic^2)	0.0007816	0.0013908	0.562	0.577102	

Examination ~ Agriculture, Catholic, Fertility, (Fertility^2).

Параметр	Значение	Std. Error	t value	Pr(> t)	Уровень значимости
(Intercept)	44.9008383	15.9098649	2.822	0.00726	**
Agriculture	-0.1631012	0.0370429	-4.403	7.2e-05	***
Catholic	-0.0405962	0.0236036	-1.720	0.09281	.
Fertility	-0.2849617	0.4973710	-0.573	0.56974	**
(Fertility^2)	0.0002975	0.0037503	0.079	0.93714	

Приложение 5. Код на языке R, необходимый для решения задач 2 и 2.2.

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

data = swiss

data
summary(data)
ggpairs(data)

#1
#проверим на линейную независимость регрессоры

model_test1=lm(Fertility~Agriculture+Catholic,data)
summary(model_test1)

model_test2=lm(Agriculture~Fertility+Catholic,data)
summary(model_test2)

model_test3=lm(Catholic~Agriculture+Fertility,data)
summary(model_test3)

#2
#Построим нашу модель
modell=lm(Examination~Agriculture+Catholic+Fertility,data)
summary(modell)

#3
#Добавим в нашу модель логарифмы регрессоров

model_log1=lm(Examination~Agriculture+Catholic+Fertility+I(log(Fertility)),data)
summary(model_log1)

model_log2=lm(Examination~Agriculture+Catholic+Fertility+I(log(Catholic)),data)
summary(model_log2)

model_log3=lm(Examination~Agriculture+Catholic+Fertility+I(log(Agriculture)),data)
summary(model_log3)

#4
#Добавим в нашу модель всевозможные произведения, а также квадраты
регрессоров

model_cmp1=lm(Examination~Agriculture+Catholic+Fertility+I(Agriculture*Catholic),data)
summary(model_cmp1)

model_cmp2=lm(Examination~Agriculture+Catholic+Fertility+I(Agriculture*Fertility),data)
summary(model_cmp2)

model_cmp3=lm(Examination~Agriculture+Catholic+Fertility+I(Fertility*Catholic),data)
summary(model_cmp3)

model_sq1=lm(Examination~Agriculture+Catholic+Fertility+I(Agriculture^2),data)
summary(model_sq1)

model_sq2=lm(Examination~Agriculture+Catholic+Fertility+I(Catholic^2),data)
```

```

summary(model_sq2)

model_sq3=lm(Examination~Agriculture+Catholic+Fertility+I(Fertility^2),data)
summary(model_sq3)

#2.2

# 1)

#Критерий Стьюдента
t_critical = qt(0.975, df = 43)
t_critical #t=2.017

#доверительные интервалы:
#[B-t*СКО, B+t*СКО]

#свободный
#B=43.68, СКО=4.11
#[43.68-4.11*2.017, 43.68+4.11*2.017 ]
#[35.39, 51.97]

#Agriculture
#B=-0.16, СКО=0.33
#[ -0.16-0.33*2.017, -0.16+0.33*2.017]
#[ -0.83, 0.56]

#Catholic
#B=-0.39, СКО=0.19
#[ -0.39-0.19*2.017, -0.39+0.19*2.017]
#[ -0.77, -0.01]

#Fertility
#B=-0.25, СКО=0.06
#[ -0.25-0.06*2.017, -0.25+0.06*2.017]
#[ -0.37, -0.13]

# 3)
new.data = data.frame(Catholic = 84.32, Agriculture=45.1, Fertility=83.1)
predict(modell, new.data, interval = "confidence")

#43.68-0.16*45.1-0.039*84.32-0.24*83.1
# 13.23152

```

Приложение 6. Код на языке R, необходимый для решения задачи 3.

```
install.packages("devtools")
devtools::install_github("https://github.com/bdemeshev/rlms")

library("memisc")
library("GGally")
library("dplyr")
library("psych")
library("lmtest")
library("sjPlot")
library("sgof")
library("ggplot2")
library("foreign")
library("car")
library("hexbin")
library("rlms")
library("devtools")
library("rstatix")
library("sandwich")
library("haven")

data <- read.csv("r12i_os26b.csv", sep=";", dec = ".", header=TRUE)
glimpse(data)

#Выборка данных для описания соц-эконом положения граждан РФ
#hh5      Пол(1-М, 2-Ж)
#hj13.2   Среднемесячная з/п за год
#h_marst  Семейное положение
#         (1-никогда не состоял в браке,
#         2-Состоит в зарег. браке,
#         3-Живут вместе, но не зарег,
#         4-Разведен и не состоит в браке,
#         5-Вдовец(вдова),
#         6-офиц. зарег, но живут не вместе)
#h_diplom Высшее образование
#         (1-Окончил 0-6 классов,
#         2-Незаконч среднее образование(7-8кл),
#         3-2+что-то ещё,
#         4-Законч. Ср.Обр,
#         5-Законч. Ср.Спец.Обр,
#         6-Законч. Высш.Обр. и выше)
#h_age    возраст
#status   Тип населённого пункта
#         (1-Обл.Центр,
#         2-Город,
#         3-Посёлок городского типа,
#         4-Село)
#hj6.2    Длительность рабочей недели
data = select(data, hh5, h_age, h_marst, h_diplom, status, hj13.2, hj6.2)
#Убираем объекты содержащие N/A
data = na.omit(data)
#Получаем представление о наших данных
glimpse(data)

#Новая база данных для нормализованных значений
data2 = select(data)

#Сделаем Дамми-переменные по параметру Семейное Положение

#Женат?(1-Да, 0-Нет)
```

```

data2$wed1 = 0
data2$wed1[which(data$h_marst == 2)] <- 1
data2$wed1[which(data$h_marst == 6)] <- 1

#Разведён или вдовец? (1-Да, 0-Нет)
data2$wed2 = 0
data2$wed2[which(data$h_marst == 4)] <- 1
data2$wed2[which(data$h_marst == 5)] <- 1

#Никогда не состоял в браке? (1-Да)
data2$wed3 = 0
data2$wed3[which(data$h_marst == 1)] <- 1

# Проверим, что отсутствует линейная зависимость между семейными положениями
vif(lm(data$h_marst ~ data2$wed1 + data2$wed2 + data2$wed3))

#Из параметра пол делаем переменную sex (1-Мужчина, 0-Женщина)
data2["sex"] = 0
data2$sex[which(data$hh5 == 1)] <- 1

#Из параметра тип населённого пункта делаем дамми-переменную city_status
#(1-Город илил Обл.Центр, 0-В противном случае)
data2$city_status = 0
data2$city_status[which(data$status == 1)] <- 1
data2$city_status[which(data$status == 2)] <- 1

#Введём параметр higher_educ Наличие полного высшего обр (1-Да, 0-Нет)
data2$higher_educ = 0
data2$higher_educ[which(data$h_diplom == 6)] <- 1

#Возраст
age = data$h_age
data2["age"] = (age - mean(age)) / sqrt(var(age))

#Нормализованное среднее число рабочих часов в неделю
working_hours = data$hj6.2
data2$working_hours = (working_hours - mean(working_hours)) /
sqrt(var(working_hours))

#Нормализованная средняя зарплата
wage = data$hj13.2
data2$wage = (wage - mean(wage)) / sqrt(var(wage))

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

modell1_1 = lm(data = data2, wage ~ sex + age + wed1 + wed2 +
               wed3 + higher_educ + city_status + working_hours)
vif(modell1_1)
summary(modell1_1)
#Multiple R-squared - 0.02044
#Adjusted R-squared - 0.01819

```

```

#Уберём wed3, city_status с плохой p-статистикой
modell1_2 = lm(data = data2, wage ~ sex +
              wed1 + age + wed2 + higher_educ + working_hours)
vif(modell1_2)
summary(modell1_2)

#VIF всех параметров уменьшился
#Multiple R-squared - 0.02035 ~ Такой же
#Adjusted R-squared - 0.01866 Чуть лучше

#2

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

#####Логарифмы#####

model2_1 = lm(data = data2, wage ~ sex + wed1 + age + wed2 +
              higher_educ + working_hours + I(log(Mod(working_hours))) +
              I(log(Mod(age))))
vif(model2_1)
summary(model2_1)
#VIF огромный, значимость коэффициентов низкая
#Исключим working_hours

model2_2 = lm(data = data2, wage ~ sex + wed1 + age + wed2 +
              higher_educ + I(log(Mod(working_hours))) + I(log(Mod(age))))
vif(model2_2)
summary(model2_2)
#Multiple R-squared - 0.02071
#Adjusted R-squared - 0.01874 Примерно как у нашей модели 1
#Уберём Маловажный higher_educ

model2_3 = lm(data = data2, wage ~ sex + wed1 + age +
              I(log(Mod(working_hours))) + I(log(Mod(age))) )
vif(model2_3)
summary(model2_3)
#Multiple R-squared - 0.02056
#Adjusted R-squared - 0.01915
#Немного лучше R^2 , коэффициенты подсчитаны более точно, VIF не превышает
1.01

#####Степени#####

current_pow = 0.1
model2_4 = lm(data = data2, wage ~ sex + wed1 + age + wed2 +
              higher_educ + I(Mod(working_hours)^current_pow) +
              I(Mod(age)^current_pow))
vif(model2_4)
summary(model2_4)
#Уберём wed2 из-за большого VIF и higher_educ как маловажный

current_pow = 0.1
model2_5 = lm(data = data2, wage ~ sex + wed1 + age +

```



```

                                I (Mod(working_hours)^current_pow) + I (Mod(age)^current_pow))
vif(model2_5)
summary(model2_5)
#Multiple R-squared - 0.02056
#Adjusted R-squared - 0.01916
#VIF < 1.1

current_pow = 0.2
model2_6 = lm(data = data2, wage ~ sex + wed1 + age +
              I (Mod(working_hours)^current_pow) + I (Mod(age)^current_pow))
vif(model2_6)
summary(model2_6)
#Multiple R-squared - 0.02056
#Adjusted R-squared - 0.01915
#VIF < 1.1

current_pow = 0.3
model2_7 = lm(data = data2, wage ~ sex + wed1 + age +
              I (Mod(working_hours)^current_pow) + I (Mod(age)^current_pow))
vif(model2_7)
summary(model2_7)
#Multiple R-squared - 0.02055
#Adjusted R-squared - 0.01914
#VIF < 1.1

current_pow = 0.4
model2_8 = lm(data = data2, wage ~ sex + wed1 + age +
              I (Mod(working_hours)^current_pow) + I (Mod(age)^current_pow))
vif(model2_8)
summary(model2_8)
#Multiple R-squared - 0.02053
#Adjusted R-squared - 0.01912
#VIF < 1.1

current_pow = 0.5
model2_9 = lm(data = data2, wage ~ sex + wed1 + age +
              I (Mod(working_hours)^current_pow) + I (Mod(age)^current_pow))
vif(model2_9)
summary(model2_9)
#Multiple R-squared - 0.02051
#Adjusted R-squared - 0.0191
#VIF < 1.102

current_pow = 0.6
model2_10 = lm(data = data2, wage ~ sex + wed1 + age +
               I (Mod(working_hours)^current_pow) + I (Mod(age)^current_pow))
vif(model2_10)
summary(model2_10)
#Multiple R-squared - 0.02048
#Adjusted R-squared - 0.01908
#VIF < 1.105

current_pow = 0.7
model2_11 = lm(data = data2, wage ~ sex + wed1 + age +
               I (Mod(working_hours)^current_pow) + I (Mod(age)^current_pow))
vif(model2_11)
summary(model2_11)

```

```

#Multiple R-squared - 0.02045
#Adjusted R-squared - 0.01905
#VIF < 1.106

current_pow = 0.8
model2_12 = lm(data = data2, wage ~ sex + wed1 + age +
               I(Mod(working_hours)^current_pow) + I(Mod(age)^current_pow))
vif(model2_12)
summary(model2_12)
#Multiple R-squared - 0.02042
#Adjusted R-squared - 0.01902
#VIF < 1.108

current_pow = 0.9
model2_13 = lm(data = data2, wage ~ sex + wed1 + age +
               I(Mod(working_hours)^current_pow) + I(Mod(age)^current_pow))
vif(model2_13)
summary(model2_13)
#Multiple R-squared - 0.0204
#Adjusted R-squared - 0.01899
#VIF < 1.109

current_pow = 1.1
model2_14 = lm(data = data2, wage ~ sex + wed1 + age +
               I(Mod(working_hours)^current_pow) + I(Mod(age)^current_pow))
vif(model2_14)
summary(model2_14)
#Multiple R-squared - 0.02034
#Adjusted R-squared - 0.01893
#VIF < 1.111

current_pow = 1.2
model2_15 = lm(data = data2, wage ~ sex + wed1 + age +
               I(Mod(working_hours)^current_pow) + I(Mod(age)^current_pow))
vif(model2_15)
summary(model2_15)
#Multiple R-squared - 0.02032
#Adjusted R-squared - 0.01891
#VIF < 1.112

current_pow = 1.3
model2_16 = lm(data = data2, wage ~ sex + wed1 + age +
               I(Mod(working_hours)^current_pow) + I(Mod(age)^current_pow))
vif(model2_16)
summary(model2_16)
#Multiple R-squared - 0.0203
#Adjusted R-squared - 0.01889
#VIF < 1.113

current_pow = 1.4
model2_17 = lm(data = data2, wage ~ sex + wed1 + age +
               I(Mod(working_hours)^current_pow) + I(Mod(age)^current_pow))
vif(model2_17)
summary(model2_17)
#Multiple R-squared - 0.02028
#Adjusted R-squared - 0.01887
#VIF < 1.113

```

```

current_pow = 1.5
model2_18 = lm(data = data2, wage ~ sex + wed1 + age +
               I(Mod(working_hours)^current_pow) + I(Mod(age)^current_pow))
vif(model2_18)
summary(model2_18)
#Multiple R-squared - 0.02026
#Adjusted R-squared - 0.01886
#VIF < 1.114

```

```

current_pow = 1.6
model2_19 = lm(data = data2, wage ~ sex + wed1 + age +
               I(Mod(working_hours)^current_pow) + I(Mod(age)^current_pow))
vif(model2_19)
summary(model2_19)
#Multiple R-squared - 0.02025
#Adjusted R-squared - 0.01885
#VIF < 1.114

```

```

current_pow = 1.7
model2_20 = lm(data = data2, wage ~ sex + wed1 + age +
               I(Mod(working_hours)^current_pow) + I(Mod(age)^current_pow))
vif(model2_20)
summary(model2_20)
#Multiple R-squared - 0.02024
#Adjusted R-squared - 0.01884
#VIF < 1.114

```

```

current_pow = 1.8
model2_21 = lm(data = data2, wage ~ sex + wed1 + age +
               I(Mod(working_hours)^current_pow) + I(Mod(age)^current_pow))
vif(model2_21)
summary(model2_21)
#Multiple R-squared - 0.02024
#Adjusted R-squared - 0.01883
#VIF < 1.114

```

```

current_pow = 1.9
model2_22 = lm(data = data2, wage ~ sex + wed1 + age +
               I(Mod(working_hours)^current_pow) + I(Mod(age)^current_pow))
vif(model2_22)
summary(model2_22)
#Multiple R-squared - 0.02024
#Adjusted R-squared - 0.01883
#VIF < 1.114

```

```

current_pow = 2.0
model2_23 = lm(data = data2, wage ~ sex + wed1 + age +
               I(Mod(working_hours)^current_pow) + I(Mod(age)^current_pow))
vif(model2_23)
summary(model2_23)
#Multiple R-squared - 0.02024
#Adjusted R-squared - 0.01883
#VIF < 1.114

```

#Заметим, что с увеличением степени VIF немного увеличивается а R^2 падает

```
#####Произведение#####

model2_24 = lm(data = data2, wage ~ sex+
               wed1 + age + higher_educ + working_hours + I(working_hours
* age))
vif(model2_24)
summary(model2_24)
#Multiple R-squared - 0.02044
#Adjusted R-squared - 0.01847
#Уберём wed2 как незначительный + с самым большим VIF

model2_25 = lm(data = data2, wage ~ sex+
               wed1 + age + higher_educ + working_hours + I(working_hours
* age))
vif(model2_25)
summary(model2_25)
#Multiple R-squared - 0.02037
#Adjusted R-squared - 0.01868

#3
#Лучшие модели

current_pow = 0.1
model2_5 = lm(data = data2, wage ~ sex + wed1 + age +
              I(Mod(working_hours)^current_pow) + I(Mod(age)^current_pow))
vif(model2_5)
summary(model2_5)
#Multiple R-squared - 0.02056
#Adjusted R-squared - 0.01916

model2_3 = lm(data = data2, wage ~ sex + wed1 + age +
              I(log(Mod(working_hours))) + I(log(Mod(age))))
vif(model2_3)
summary(model2_3)
#Multiple R-squared - 0.02056
#Adjusted R-squared - 0.01915

current_pow = 0.2
model2_6 = lm(data = data2, wage ~ sex + wed1 + age +
              I(Mod(working_hours)^current_pow) + I(Mod(age)^current_pow))
vif(model2_6)
summary(model2_6)
#Multiple R-squared - 0.02056
#Adjusted R-squared - 0.01915
#VIF < 1.1

#Значения R^2 в этих моделях практически неотличаются, но в первой модели
#Значимость переменных выше => она лучшая

#4

#Согласно построенной нами модели наибольшую зарплату получают
#Неженатые молодые мужчины, работающие большее количество часов

#5
```

```

current_pow = 0.1

#Не вступавшие в брак
data3 = subset(data, wed3 == 1)
#Без высшего обр
data3 = subset(data3, higher_educ == 0)
model5_1 = lm(data = data3, wage ~ sex + age + city_status +
              + I(Mod(working_hours)^current_pow) +
              I(Mod(age)^current_pow))
summary(model5_1)
#Согласно модели выше, большую зарплату получают те, кто много работает

#Городские жители
data3=subset(data2, city_status == 1)
#Состоящие в браке
data3=subset(data3, wed1 == 1)
model5_2 = lm(data = data3, wage ~ sex + age +
              + I(Mod(working_hours)^current_pow) + I(Mod(age)^current_pow))
summary(model5_2)
#Согласно модели выше, большую зарплату получают молодые и много работающие

```

Приложение 7. Код на Python с чтением данных StudentsPerformance, выборкой необходимых параметров, созданием, обучением и оценкой SVM классификатора с целевым признаком writing_score_above_avg.

```
#Импорт модулей
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.svm import SVC
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.preprocessing import OneHotEncoder

# Загрузка данных
exam_df = pd.read_csv('StudentsPerformance.csv')

# Выберем необходимые нам параметры
exam_df = exam_df.loc[:, exam_df.columns.isin(['gender', 'parental level of
education', 'writing score', 'reading score', 'math score',])]

#Смотрим на наши данные
print(exam_df)

#Удаление пустых значений (если таковы имеются)
exam_df = exam_df.dropna()

# Выделение целевого признака ( writing score (выше среднего значения — класс
0, ниже или совпадает — класс 1) )
exam_df['writing_score_above_avg'] = (exam_df['writing score'] >
exam_df['writing score'].mean()).astype(int)

# Удаление целевого признака
exam_df.drop('writing score', axis=1, inplace=True)

# Создание дамми-переменных с помощью one-hot-encoding

#Пол (Мужчина - 1, женщина - 0)
exam_df['gender'] = np.where(exam_df['gender'] == 'male' , 0, 1)

#Уровень образования (Бакалавр - 1, Магистр - 2, Аспирант - 3, High school -
4, Колледж - 5, Some high school - 6)
exam_df['parental level of education'] = np.where(exam_df['parental level of
education'] == 'bachelor\'s degree' ,
                                                    1, exam_df['parental level
of education'])
exam_df['parental level of education'] = np.where(exam_df['parental level of
education'] == 'master\'s degree' ,
                                                    2, exam_df['parental level
of education'])
exam_df['parental level of education'] = np.where(exam_df['parental level of
education'] == 'associate\'s degree' ,
                                                    3, exam_df['parental level
of education'])
exam_df['parental level of education'] = np.where(exam_df['parental level of
education'] == 'high school' ,
                                                    4, exam_df['parental level
of education'])
exam_df['parental level of education'] = np.where(exam_df['parental level of
education'] == 'some college' ,
                                                    5, exam_df['parental level
of education'])
```

```

exam_df['parental level of education'] = np.where(exam_df['parental level of
education'] == 'some high school' ,
                                                    6, exam_df['parental level
of education'])

df_encoded = exam_df

# Разделение на обучающую и тестовую выборки
X_training, X_testing, y_training, y_testing =
train_test_split(df_encoded.drop('writing_score_above_avg', axis=1),
df_encoded['writing_score_above_avg'],
                                                    test_size = 0.3,
                                                    random_state = 42)

# Создание и обучение модели SVM
clf = SVC(kernel='linear', random_state=42)
clf.fit(X_training, y_training)

# Получение прогнозов на тестовой выборке
y_pred = clf.predict(X_testing)

# Оценка точности классификатора с помощью метрик precision, recall и F1
precision = precision_score(y_testing, y_pred)
recall = recall_score(y_testing, y_pred)
f1 = f1_score(y_testing, y_pred)

#Результат
print( "accuracy:"+str(np.average(cross_val_score(clf, X_testing, y_testing,
scoring= 'accuracy'))))
print( "f1:"+str(np.average(cross_val_score(clf, X_testing, y_testing,
scoring= 'f1'))))
print("precision:"+str(np.average(cross_val_score(clf, X_testing, y_testing,
scoring= 'precision'))))
print( "recall:"+str(np.average(cross_val_score(clf, X_testing, y_testing,
scoring= 'recall'))))

```


Приложение 8. Код на Python с обучением и оценкой качества классификатора типа Random Forest для набора данных StudentsPerformance.

```
#Подключаем модуль
from sklearn.ensemble import RandomForestClassifier

# разделение данных на обучающую и тестовую выборки
training_data, testing_data, training_labels, testing_labels =
train_test_split(
    df_encoded.drop("writing_score_above_avg", axis=1),
    df_encoded["writing_score_above_avg"], test_size=0.2, random_state=42)

# построение классификатора
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(training_data, training_labels)

# оценка качества классификатора на тестовой выборке
pred_labels = rf.predict(testing_data)
precision = precision_score(testing_labels, pred_labels)
recall = recall_score(testing_labels, pred_labels)
f1 = f1_score(testing_labels, pred_labels)

print( "accuracy:"+str(np.average(cross_val_score(rf, X_testing, y_testing,
scoring= 'accuracy'))))
print( "f1:"+str(np.average(cross_val_score(rf, X_testing, y_testing,
scoring= 'f1'))))
print("precision:"+str(np.average(cross_val_score(rf, X_testing, y_testing,
scoring= 'precision'))))
print( "recall:"+str(np.average(cross_val_score(rf, X_testing, y_testing,
scoring= 'recall'))))
```

Приложение 9. Код на Python с использованием поиска по сетке GridSearch для Random Forest.

```
#Делаем импорт ещё пары модулей необходимых нам
from sklearn.metrics import classification_report
from sklearn.model_selection import GridSearchCV

# определение диапазона значений для перебора
param_grid = {
    "n_estimators": [50, 100, 150, 200],
    "max_depth": [None, 5, 10, 15],
    "min_samples_split": [2, 5, 10],
    "min_samples_leaf": [1, 2, 4],
    "max_features": ["sqrt", "log2"],
}

# создание экземпляра класса GridSearchCV
grid_search_rf = GridSearchCV(RandomForestClassifier(random_state=42),
param_grid, cv=3, n_jobs=-1)

# запуск GridSearchCV
grid_search_rf.fit(training_data, training_labels)

# Лучшие гиперпараметры
print("Best parameters:", grid_search_rf.best_params_)
```

Приложение 10. Код на Python с перебором гиперпараметра количества деревьев с шагом 50 и 10.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, recall_score, f1_score

# Различные комбинации гиперпараметров для случайного дерева с шагом 50 в
параметре n_estimators
param_grid_50 = {
    'n_estimators': [i for i in range(150, 301, 50)]
}

rfc_50 = RandomForestClassifier(random_state=42)
grid_search_50 = GridSearchCV(estimator=rfc_50, param_grid=param_grid_50,
cv=5)
grid_search_50.fit(X_training, y_training)

# Вывод лучших значений для шага 50 в параметре n_estimators
print("Best n_estimators for step 50: ",
grid_search_50.best_params_['n_estimators'])

# Прогноз на основе данных тестирования с помощью случайного дерева с шагом
50 в параметре n_estimators
rfc_50_pred = grid_search_50.predict(X_testing)

# Точность, отзыв и оценку F1, используя метод случайный лес с шагом 50
rfc_50_accuracy = accuracy_score(y_testing, rfc_50_pred)
rfc_50_recall = recall_score(y_testing, rfc_50_pred, average='weighted')
rfc_50_f1_score = f1_score(y_testing, rfc_50_pred, average='weighted')

print("Accuracy for step 50: ", rfc_50_accuracy)
print("Recall for step 50: ", rfc_50_recall)
print("F1 Score for step 50: ", rfc_50_f1_score)

# Различные комбинации гиперпараметров для случайного дерева с шагом 10 в
параметре n_estimators
param_grid_10 = {
    'n_estimators': [i for i in range(150, 301, 10)]
}

rfc_10 = RandomForestClassifier(random_state=42)
grid_search_10 = GridSearchCV(estimator=rfc_10, param_grid=param_grid_10,
cv=5)
grid_search_10.fit(X_training, y_training)

# Вывод лучших значений для шага 10 в параметре n_estimators
print("Best n_estimators for step 10: ",
grid_search_10.best_params_['n_estimators'])

# Прогноз на основе данных тестирования с помощью случайного дерева с шагом
10 в параметре n_estimators
rfc_10_pred = grid_search_10.predict(X_testing)

# Точность, отзыв и оценку F1, используя метод случайный лес с шагом 10
rfc_10_accuracy = accuracy_score(y_testing, rfc_10_pred)
rfc_10_recall = recall_score(y_testing, rfc_10_pred, average='weighted')
rfc_10_f1_score = f1_score(y_testing, rfc_10_pred, average='weighted')

print("Accuracy for step 10: ", rfc_10_accuracy)
print("Recall for step 10: ", rfc_10_recall)
print("F1 Score for step 10: ", rfc_10_f1_score)
```

Приложение 11. Код на языке C++ для решения задачи 5.

```
#include <GL/glut.h>
#include <iostream>
#include <vector>
#include <tuple>

// Настройки камеры(положение)
float cameraDistance = 5.0f;
float cameraPosX = 0.0f;
float cameraPosY = 0.0f;

float rotate_x = 0.5;
float rotate_y = 0.5;
float zoom_x = 0.2;

// Точка
class Point
{
private:
    // Размерность
    const int dimension;
    // Координаты
    std::vector<float> coordinate = std::vector<float>(dimension);
public:

    // Конструкторы
    Point(const int _dimension)
        : dimension(_dimension)
    { }

    Point(const int _dimension, const std::vector<float>& _coordinate)
        : dimension(_dimension)
        , coordinate(_coordinate)
    { }

    // Оператор присваивания
    Point& operator=(const Point& a)
    {
        coordinate = a.coordinate;
        return *this;
    }

    int getDim() const
    {
        return dimension;
    }

    std::vector<float> getCoord() const
    {
        return coordinate;
    }

};

// Сплаины
class Kochanek_Bartels_spline
{
private:
    // Количество точек
    int num_of_points;
```

```

// Target biad continuity for each part
std::vector<std::tuple<float, float, float>> tbc;

// Точки
std::vector<Point> points;
// Касательные
std::vector<Point> tangent;

public:

    // Конструктор
    Kochanek_Bartels_spline(const int _num_of_points, const
std::vector<Point>& _points, const std::vector<std::tuple<float, float, float>>
_tbc)
        : num_of_points(_num_of_points)
        , points(_points)
        , tbc(_tbc)
    { }

    // Геттер точек
    std::vector<Point> getPoints() const
    {
        return points;
    }

    //  $p(x) = (h_{00}(t) * p[k]) + (h_{10}(t) * (x[k+1] - x[k]) * m[k]) + (h_{01}(t) * p[k+1]) + (h_{11}(t) * (x[k+1] - x[k]) * m[k+1])$ 

    // Базисные функции Эрмита
    float h00(const float x) const
    {
        return 2 * x * x * x - 3 * x * x + 1;
    }

    float h10(const float x) const
    {
        return x * x * x - 2 * x * x + x;
    }

    float h01(const float x) const
    {
        return -2 * x * x * x + 3 * x * x;
    }

    float h11(const float x) const
    {
        return x * x * x - x * x;
    }

    // Геттер количества точек
    int getNumOfPoints() const
    {
        return num_of_points;
    }

    // Вычисление касательной
    void calculate_tangent()
    {
        tangent.resize(num_of_points, Point(points[0].getDim()));
        //  $((1 - t) * (1 + b) * (1 + c) * (p[i] - p[i-1]) / 2) + ((1 - t) * (1 - b) * (1 - c) * (p[i+1] - p[i]) / 2)$ 
        for (int index = 0; index < num_of_points; ++index)

```

```

        {
            std::vector<float> temp_coord(points[index].getDim());
            for (int i = 0, g = 0; i < points[index].getDim(); ++i, g += ((i
% (num_of_points / tbc.size()) - 1 == 0) && i != 0))
                temp_coord[i] = ((1 - std::get<0>(tbc[g])) * (1 +
std::get<1>(tbc[g])) * (1 + std::get<2>(tbc[g])) * (points[(index +
num_of_points) % num_of_points].getCoord()[i] - points[(index - 1 +
num_of_points) % num_of_points].getCoord()[i]) / 2 + (1 -
std::get<0>(tbc[g])) * (1 - std::get<1>(tbc[g])) * (1 - std::get<2>(tbc[g]))
* (points[(index + 1 + num_of_points) % num_of_points].getCoord()[i] -
points[(index + num_of_points) % num_of_points].getCoord()[i]) / 2);
            tangent[index] = { points[index].getDim(), temp_coord };
        }
    }
    // Геттер касательных
    std::vector<Point> getTangent()
    {
        return tangent;
    }

    // Вычисление p(x)
    Point calculate_function(const float x, const int start_index, const int
end_index) const
    {
        std::vector<float> coord(points[start_index].getDim() - 1);

        // t = (x - x[k]) / (x[k+1] - x[k])
        float t = (x - points[start_index].getCoord()[0]) /
(points[end_index].getCoord()[0] - points[start_index].getCoord()[0]);

        for (int i = 1; i < points[start_index].getDim(); ++i)
            coord[i - 1] = (h00(t) * points[start_index].getCoord()[i] +
h10(t) * (points[end_index].getCoord()[0] -
points[start_index].getCoord()[0]) * tangent[start_index].getCoord()[i] +
h01(t) * points[end_index].getCoord()[i] + h11(t) *
(points[end_index].getCoord()[0] - points[start_index].getCoord()[0]) *
tangent[end_index].getCoord()[i]);

        return { points[start_index].getDim(), coord };
    }

    // Изображаем сплайны(если это возможно)
    void draw_spline()
    {
        glBegin(GL_LINES);

        calculate_tangent();

        if (points[0].getDim() == 2) // 2D
        {
            for (int i = 0; i < num_of_points; ++i)
            {
                float sign = (points[(i + 1) % num_of_points].getCoord()[0] -
points[i % num_of_points].getCoord()[0]) / abs((points[(i + 1) %
num_of_points].getCoord()[0] - points[i % num_of_points].getCoord()[0]));
                if (sign > 0)
                {
                    for (float x_pos = points[i %
num_of_points].getCoord()[0]; x_pos < points[(i + 1) %
num_of_points].getCoord()[0]; x_pos += sign / 10)
                    {
                        glVertex2f(x_pos, calculate_function(x_pos, i %
num_of_points, (i + 1) % num_of_points).getCoord()[0]);
                    }
                }
            }
        }
    }

```

```

        glColor3f(1.0, 0.0, 0.0);
    }
    else
    {
        for (float x_pos = points[i %
num_of_points].getCoord()[0]; x_pos > points[(i + 1) %
num_of_points].getCoord()[0]; x_pos += sign / 10)
        {
            glVertex2f(x_pos, calculate_function(x_pos, i %
num_of_points, (i + 1) % num_of_points).getCoord()[0]);
            glColor3f(1.0, 0.0, 0.0);
        }
    }
}
else if (points[0].getDim() == 3) // 3D
{
    for (int i = 0; i < num_of_points; i++)
    {
        float sign = (points[(i + 1) % num_of_points].getCoord()[0] -
points[i % num_of_points].getCoord()[0]) / abs((points[(i + 1) %
num_of_points].getCoord()[0] - points[i % num_of_points].getCoord()[0]));
        if (sign > 0)
        {
            for (float x_pos = points[i %
num_of_points].getCoord()[0]; x_pos < points[(i + 1) %
num_of_points].getCoord()[0]; x_pos += sign / 1000)
            {
                glVertex3f(x_pos, calculate_function(x_pos, i %
num_of_points, (i + 1) % num_of_points).getCoord()[0],
calculate_function(x_pos, i % num_of_points, (i + 1) %
num_of_points).getCoord()[1]);
                glColor3f(1.0, 1.0, 1.0);
            }
        }
        else
        {
            for (float x_pos = points[i %
num_of_points].getCoord()[0]; x_pos > points[(i + 1) %
num_of_points].getCoord()[0]; x_pos += sign / 1000)
            {
                glVertex3f(x_pos, calculate_function(x_pos, i %
num_of_points, (i + 1) % num_of_points).getCoord()[0],
calculate_function(x_pos, i % num_of_points, (i + 1) %
num_of_points).getCoord()[1]);
                glColor3f(1.0, 1.0, 1.0);
            }
        }
    }
}
else //nD
// Не можем рисовать, выводим функции
{
    std::cout << "Cannot draw > 3 dimension graph" << "\n";
    for (int j = 0; j < points[0].getDim() - 1; j++)
    {
        std::cout << "functions of " << j + 2 << " dimension" <<
"respect to x are: \n";
        for (int i = 0; i < num_of_points; i++)
        {

```



```

        std::cout << "function between point x = " << points[i %
num_of_points].getCoord()[0] << " and x = " << points[(i + 1) %
num_of_points].getCoord()[0] << "\n";
        std::cout << "t = " << "(x - " << points[i %
num_of_points].getCoord()[0] << ") / " << "( " << points[(i + 1) %
num_of_points].getCoord()[0] - points[i % num_of_points].getCoord()[0] <<
")\n";

        std::cout << "( 2t^3 - 3t^2 + 1) * " << points[i %
num_of_points].getCoord()[j + 1] << " + (t^3 - 2t^2 + t) * " << (points[(i +
1) % num_of_points].getCoord()[0] - points[(i) %
num_of_points].getCoord()[0]) * tangent[i % num_of_points].getCoord()[j + 1]
<< " + (-2t^3 + 3t^2) * " << points[(i + 1) % num_of_points].getCoord()[j +
1] << " + (t^3 - t^2) * " << (points[(i + 1) % num_of_points].getCoord()[0] -
points[(i) % num_of_points].getCoord()[0]) * tangent[(i + 1) %
num_of_points].getCoord()[j + 1] << "\n";
    }
}

}

glEnd();

}

// Рисуем оси
void draw_coordinate_Oxyz()
{
    // x
    glColor3f(1.0f, 0.0f, 0.0f); // Красный
    glBegin(GL_LINES);
    glVertex3f(-10.0f, 0.0f, 0.0f);
    glVertex3f(10.0f, 0.0f, 0.0f);
    glEnd();

    // y
    glColor3f(0.0f, 1.0f, 0.0f); // Зелёный
    glBegin(GL_LINES);
    glVertex3f(0.0f, -10.0f, 0.0f);
    glVertex3f(0.0f, 10.0f, 0.0f);
    glEnd();

    // z
    glColor3f(0.0f, 0.0f, 1.0f); // Синий
    glBegin(GL_LINES);
    glVertex3f(0.0f, 0.0f, -10.0f);
    glVertex3f(0.0f, 0.0f, 10.0f);
    glEnd();
}

};

// Клавиши взаимодействия с камерой
void specialKeyboard(const int key, const int x, const int y)
{
    switch (key)
    {
    case GLUT_KEY_UP:
        rotate_x -= 5; break;
    case GLUT_KEY_DOWN:
        rotate_x += 5; break;
    case GLUT_KEY_LEFT:
        rotate_y += 5; break;
    case GLUT_KEY_RIGHT:
        rotate_y -= 5; break;
    case GLUT_KEY_PAGE_UP:

```

```

        zoom_x -= 0.03; break;
    case GLUT_KEY_PAGE_DOWN:
        zoom_x += 0.03; break;
    }
    glutPostRedisplay(); // Перерисовываем
}

// Тестовый пример
const std::vector<Point> points_3d
{
    {3, {-3.5, 0, -2.0}},
    {3, {-1, 1.5, 3.5}},
    {3, {0.420, 0.1, 1.0} },
    {3, {1.2, 1.5, -2.5}},
    {3, {3.6, 0, 2.0}},
    {3, {1, -1.5, 1.0}},
    {3, {0.12, -0.1, -1.0}},
    {3, {-1.69, -1.5, 2.0}}
};

// Тестовый пример
const std::vector<Point> points_2d
{
    {2, {-3.5, -2.0}},
    {2, {-1, 3.5}},
    {2, {0.420, 1.0} },
    {2, {1.2, -2.5}},
    {2, {3.6, 2.0}},
    {2, {1, 1.0}},
    {2, {0.12, -1.0}},
    {2, {-1.69, 2.0}}
};

const std::vector<std::tuple<float, float, float>> test_tbc{ {0.0, 0.0, 0.0}
};

```

// Вектор точек (поскольку из main невозможно передать в display в качестве аргумента ничего, то существует 2 решения проблемы
// 1. Использование глобальной переменной 2. Использование статической переменной

// Т.к. статическая переменная должна всё равно получить какие-то значения, то создания глобальной переменной не избежать
// Использование глобальной переменной в рамках нашей небольшой программы допустимо (хотя в больших проектах стоит искать другой путь,
// например, помещение переменной в пространство имён)

```

std::vector<Point> point_spline;
std::vector<std::tuple<float, float, float>> g_tbc;

```

```

void display()
{
    Kochanek_Bartels_spline test_spline =
    Kochanek_Bartels_spline(point_spline.size(), point_spline, g_tbc);
    // Можно поэкспериментировать со значениями tension, bias, continuity (или
    при необходимости сделать ввод в main)

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); // Очистка буфера
    цвета и глубины
    glMatrixMode(GL_PROJECTION); // Устанавливаем матрицу проекции
    glLoadIdentity();
    glOrtho(-1, 1, -1, 1, -10, 10); // Adjust the orthographic projection
    glMatrixMode(GL_MODELVIEW);
}

```

```

glLoadIdentity();
gluLookAt(cameraPosX, cameraPosY, cameraDistance, // Camera position
cameraPosX, cameraPosY, 0.0, // Target position (look at)
0.0, 1.0, 0.0); // Adjust the camera position

glScalef(zoom_x, zoom_x, 0);
glRotatef(rotate_x, 1.0, 0.0, 0.0);
glRotatef(rotate_y, 0.0, 1.0, 0.0);

glEnable(GL_MAP1_VERTEX_3);

test_spline.draw_spline();
test_spline.draw_coordinate_Oxyz();

glutSwapBuffers();
}

int main(int argc, char** argv)
{
    char key{};
    std::cout << "If you want to check default splines enter '1' (any symbol
if you don't want):"; // Чтобы посмотреть результат тестового примера ввести
1
    std::cin >> key;
    if (key == '1')
    {
        point_spline = points_3d;
        g_tbc = test_tbc;
    }
    else
    { // Иначе вводим размерность пространства, количество частей ,параметры
tension, bias, continuity , количество точек и их координаты
        int n, number, parts;
        std::cout << "Enter the dimension(dim > 1):";
        std::cin >> n;
        std::cout << "Enter the number of points(number > 1):";
        std::cin >> number;
        point_spline.resize(number, Point(n)); // resize the vector

        std::cout << "Enter a count of parts you want to divide spline into
(>1 and <= " << number << "):";
        std::cin >> parts;
        g_tbc.resize(parts);
        for (int i = 0; i < parts; ++i)
        {
            float x, y, z;
            std::cout << "\n" << i << " part:\n";
            std::cout << "Enter the tension (value is between -1 and 1): ";
            std::cin >> x;
            std::cout << "Enter the bias (value is between -1 and 1): ";
            std::cin >> y;
            std::cout << "Enter the continuity (value is between -1 and 1): ";
            std::cin >> z;
            g_tbc[i] = { x, y, z };
        }

        std::cout << "Enter the coordinates of points:\n";
        for (int i = 0; i < number; ++i)
        {
            std::vector<float> coords(n);
            for (int g = 0; g < n; ++g)
                std::cin >> coords[g];
        }
    }
}

```

```

        point_spline[i] = { n, coords };
    }

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DEPTH | GLUT_DOUBLE);
    glutInitWindowSize(1280, 960);
    glutCreateWindow("GLUT");

    glutSpecialFunc(specialKeyboard); // Обработка нажатий с помощью
функции specialKeyboard
    glutDisplayFunc(display); // Указываем на то, что за отображение
отвечает функция display
    glEnable(GL_DEPTH_TEST);
    glutMainLoop();

    return 0;
}

glutDisplayFunc(display); // Указываем на то, что за отображение отвечает
функция display
    glEnable(GL_DEPTH_TEST);
    glutMainLoop();

    return 0;
}

```

Список литературы

1. Christoph Hanck, Martin Arnold, Alexander Gerber, and Martin Schmelzer. Introduction to Econometrics with R — режим доступа: <https://www.econometrics-with-r.org>
2. А.Б. Шипунов, Е.М. Балдин, П.А. Волкова и др.: Наглядная статистика. Используем R! (ISBN: 978-5-97060-094-8)
3. «Крупномасштабное машинное обучение вместе с Python» Бастиан Шарден, Лука Массарон, Альберто Боскетти. (ISBN: 978-5-97060-618-6)
4. Kochanek-Bartels spline wiki —Режим доступа: https://en.wikipedia.org/wiki/Kochanek–Bartels_spline, свободный
5. Properties of Kochanek-Bartels spline — Режим доступа: <https://splines.readthedocs.io/en/latest/euclidean/kochanek-bartels.html>
6. Kochanek, D. H. U., & Bartels, R. H. (1984). Interpolating splines with local tension, continuity, and bias control. ACM SIGGRAPH Computer Graphics, 18(3), 33–41 — Режим доступа: <https://sci-hub.ru/10.1145/964965.808575>