Оглавление

1. Список Задач.............................................................................................[3](#СписокЗадач)
2. Решения…………………………………………….................................[5](#Решения)
   1. Задача №1…………………………………………........................[5](#Задача1)
   2. Задача №2……………………………............................................[8](#Задача2)
   3. Задача №3………………………………………..........................[14](#Задача3)
   4. Задача №4……………………………………..............................[21](#Задача4)
   5. Задача №5………………………………………..........................[25](#Задача5)
3. Приложения…………………….............................................................[34](#Приложение)
4. Список Литературы................................................................................[56](#СписокЛитературы)

**Список Задач**

1. Задача №1
   1. Оцените среднее значение, дисперсию и СКО переменных, указанных во втором и третьем столбце.
   2. Постройте зависимости вида y = a + bx, где y – объясняемая переменная, x – регрессор(для каждого варианта по две зависимости).
   3. Оцените, насколько “хороша” модель по коэффициенту детерминации R^2?
   4. Оцените, есть ли взаимосвязь между объясняемой переменной и объясняющей переменной (по значению p-статистики, «количеству звездочек» у регрессора в модели).
2. Задача №2
   1. Проверьте, что в наборе данных нет линейной зависимости (построить зависимости между переменными, указанными в варианте, и проверить, что R2 в каждой из них невысокий). В случае, если R2 большой, один из таких столбцов можно исключить из рассмотрения.
   2. Постройте линейную модель зависимой переменной от указанных в варианте регрессоров по методу наименьших квадратов (команда lm пакета lmtest в языке R). Оценить, насколько хороша модель, согласно: 1) R^2, 2) p-значениям каждого коэффициента.
   3. Введите в модель логарифмы регрессоров (если возможно). Сравнить модели и выбрать наилучшую.
   4. Введите в модель всевозможные произведения пар регрессоров, в том числе квадраты регрессоров. Найдите одну или несколько наилучших моделей по доле объяснённого разброса в данных R^2.
3. Задача №3
   1. Постройте линейную регрессию зарплаты на все параметры, которые Вы выделили из данных мониторинга. Не забудьте оценить коэффициент вздутия дисперсии VIF.
   2. Поэкспериментируйте с функциями вещественных параметров: используйте логарифмы, степени (хотя бы от 0.1 до 2 с шагом 0.1), произведения вещественных регрессоров.
   3. Выделите наилучшие модели из построенных: по значимости параметров, включённых в зависимости, и по объяснённому с помощью построенных зависимостей разбросу adjusted R^2- *R^2adj*.
   4. Сделайте вывод о том, какие индивиды получают наибольшую зарплату.
   5. Оцените лучшие модели для подмножества индивидов, указанных в варианте. Сделайте вывод о том, какие индивиды получают наибольшую зарплату.
4. Задача №4
   1. Обработайте набор данных набор данных, указанный во втором столбце таблицы 4.1, подготовив его к решению задачи классификации. Выделите целевой признак, указанный в последнем столбце таблицы, и удалите его из данных, на основе которых будет обучаться классификатор. Разделите набор данных на тестовую и обучающую выборку. Постройте классификатор типа, указанного в третьем столбце, для задачи классификации по параметру, указанному в последнем столбце. Оцените точность построенного классификатора с помощью метрик precision, recall и F1 на тестовой выборке.
   2. Постройте классификатор типа Случайный Лес (Random Forest) для решения той же задачи классификации. Оцените его качество с помощью метрик precision, recall и F1 на тестовой выборке. С помощью GridSearch переберите различные комбинации гиперпараметров: на первой итерации задайте большие шаги (50 или 100) по числу деревьев n\_estimators. На следующих итерациях определите лучшее количество деревьев n\_estimators с точностью до 10. Какой из классификаторов оказывается лучше?
5. Задача №5
   1. Определение сплайна и его параметров
   2. Построение сплайновой кривой
   3. Алгоритм построения сплайновой кривой

**Решения**

Задача №1

Набор Данных: Swiss

Объясняемая переменная: Infant.Mortality

Регрессоры: Agriculture, Examination

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

Используя код из Приложения 1, находим среднее значение, дисперсию и СКО для Infant.Mortality, Agriculture, Examination

Таблица №1. Характеристики переменных Infant.Mortality, Agriculture, Examination

|  |  |  |  |
| --- | --- | --- | --- |
| Переменная | Среднее значение | Дисперсия | СКО |
| Infant.Mortality | 19.94255 | 8.483802 | 2.912697 |
| Agriculture | 50.65957 | 515.7994 | 22.71122 |
| Examination | 16.48936 | 63.64662 | 7.977883 |

1. Построить зависимость вида y = a + bx, где y – объясняемая переменная, x – регрессор

Затем мы построили модели линейной регрессии, чтобы оценить взаимосвязь между "Infant.Mortality" и каждый из регрессоров. Коэффициенты регрессии представлены в таблице 2 и таблице 3.

Таблица №2. Характеристики модели Зависимости Infant.Mortality от регрессора Agriculture

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Параметр | Значение | Std. Error | t value | Pr(>|t|) | Уровень значимости |
| (Intercept) | 20.337955 | 1.057543 | 19.231 | <2e-16 | \*\*\* |
| Agriculture | -0.007805 | 0.019083 | -0.409 | 0.684 |  |

Таблица №3. Характеристики модели зависимости Infant.Mortality от регрессора Examination

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Параметр | Значение | Std. Error | t value | Pr(>|t|) | Уровень значимости |
| (Intercept) | 20.62899 | 0.98846 | 20.87 | <2e-16 | \*\*\* |
| Examination | -0.04163 | 0.05407 | -0.77 | 0.445 |  |

В результате получаем зависимости вида:

y = (-0.007805) \* x + 20.337955, для зависимости Infant.Mortality от Agriculture

y = (-0.0463) \* x + 20.62899, для зависимости Infant.Mortality от Examination

1. Оценить, насколько хороша модель по коэффициенту детерминации R2.

Модель "Infant.Mortality ~ Agriculture" имеет R2, равный 0,0037, что указывает на то, что только 0,37% вариабельности у "Infant.Mortality" может быть объяснена "Agriculture". Модель "Infant.Mortality ~ Examination" имеет несколько более высокий R2, равный 0,013, что указывает на то, что 1,3% вариабельности может быть объяснено "Examination". Обе модели демонстрируют ограниченную способность объяснять изменчивость "Infant.Mortality".

1. Оценить, есть ли взаимосвязь между Объясняемой и объясняющей переменной.

Значения p для коэффициентов "Agriculture" и "Examination" превышают общепринятый порог статистической значимости в 0,05. Это означает, что нет достаточных доказательств для вывода о статистически значимой взаимосвязи между любым из регрессоров и "Infant.Mortality".

С другой стороны, для переменной Examination значение t равно -0,77, а связанное с ним значение p равно 0,445. Аналогичным образом, коэффициент не является статистически значимым, что позволяет предположить отсутствие значимой взаимосвязи между Examination и переменной результата.

Заключение

Модели зависимости Infant.Mortality от Agriculture и Examination имеют относительно низкие значения R-квадрата, что указывает на ограниченную объяснительную силу. В то время как коэффициенты для Agriculture и коэффициента Examination статистически значимы, другие модели объясняют менее 2% вариабельности в Infant.Mortality. Для повышения прогностической способности этих моделей необходимы дальнейшее совершенствование и учет дополнительных факторов.

Задача 2

Набор данных: Attitude

Объясняемая переменная: Rating

Регрессоры: Privileges, Learning, Raises

1. Проверить, что в наборе данных нет линейной зависимости между регрессорами (построить зависимости между регрессорами и проверить, что R2 в каждой из них невысокий).

Таблица №4

|  |  |  |
| --- | --- | --- |
| Privileges | Learning | Raises |
| 1.373324 | 1.865749 | 1.761178 |

Сначала мы проверили линейную независимость между регрессорами, используя Variance Inflation Factor (VIF). VIF для переменных "Privileges", "Learning" и "Raises" равны 1,373324, 1,865749 и 1,761178 (в таблице №4) соответственно. Поскольку все эти значения значительно ниже порогового значения 5, можно с уверенностью заключить, что линейная независимость не является проблемой в этом наборе данных.

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

Мы построили линейную регрессионную модель с Rating в качестве зависимой переменной и Privileges, Learning и Raises заработной платы в качестве независимых переменных.

Таблица №5: Характеристики модели зависимости параметра Rating от параметров Privileges, Learning, Raises в наборе данных Attitude. График построенной модели находится в приложении :::

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Параметр | Значение | Std.Error | t value | Pr(>|t|) |
| (Intercept) | 14.1672 | 11.5195 | 1.230 | 0.2298 |
| Privileges | 0.1046 | 0.1682 | 0.622 | 0.5396 |
| Learning | 0.3936 | 0.2044 | 1.926 | 0.0651 |
| Raises | 0.3516 | 0.2242 | 1.568 | 0.1289 |

R2 этой модели составляет 45%, что позволяет предположить, что модель объясняет 45% различий в рейтинге. Однако ни одно из значений p для регрессоров не является статистически значимым на стандартном уровне.

1. Ввести в модель логарифмы регрессов. Сравнить модели и выбрать наилучшую.

Введём в нашу модель логарифмы регрессоров. Ниже, в таблицах 6-8 приведены характеристики построенных нами моделей.

Таблица №6: Характеристики модели зависимости параметра Rating от параметров Privileges, Learning, Raises, log(Privileges)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Параметр | Значение | Std.Error | t value | Pr(>|t|) |
| (Intercept) | -56.7404 | 164.1343 | -0.346 | 0.732 |
| Privileges | -0.3394 | 1.0393 | -0.327 | 0.747 |
| Learning | 0.3607 | 0.2211 | 1.631 | 0.115 |
| Raises | 0.3516 | 0.2278 | 1.544 | 0.135 |
| log(Privileges) | 24.4116 | 56.3634 | 0.433 | 0.669 |

Таблица №7: Характеристики модели зависимости параметра Rating от параметров Privileges, Learning, Raises, log(Learning)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Параметр | Значение | Std.Error | t value | Pr(>|t|) |
| (Intercept) | 252.3294 | 232.1077 | 1.087 | 0.2874 |
| Privileges | 0.1077 | 0.1681 | 0.641 | 0.5275 |
| Learning | 1.8464 | 1.4287 | 1.292 | 0.2081 |
| Raises | 0.3972 | 0.2283 | 1.740 | 0.0942 |
| log(Learning) | -80.5935 | 78.4481 | -1.027 | 0.3141 |

Таблица №8: Характеристики модели зависимости параметра Rating от параметров Privileges, Learning, Raises, log(Raises)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Параметр | Значение | Std.Error | t value | Pr(>|t|) |
| (Intercept) | -326.23303 | 335.80662 | -0.971 | 0.3406 |
| Privileges | 0.09615 | 0.16833 | 0.571 | 0.5730 |
| Learning | 0.38870 | 0.20434 | 1.902 | 0.0687 |
| Raises | -1.32165 | 1.66480 | -0.794 | 0.4347 |
| log(Raises) | 108.09934 | 106.57789 | 1.014 | 0.3202 |

Сделаем вывод, что модель Rating ~ Privileges, Learning, Raises, log(Learning) является наилучшей, так как в ней большее количество коэффициентов подсчитано с меньшей погрешностью, по сравнению с двумя другими моделями.

1. Ввести в модель всевозможные произведения пар регрессоров, в том числе квадраты регрессоров. Найти одну или несколько наилучших моделей по доле объяснённого разброса в данных R2

Рассмотрим следующие модели:

1. Rating ~ Privileges, Learning, Raises, (Privileges \* Learning)
2. Rating ~ Privileges, Learning, Raises, (Privileges \* Raises)
3. Rating ~ Privileges, Learning, Raises, (Raises \* Learning)
4. Rating ~ Privileges, Learning, Raises, (Privileges ^ 2)
5. Rating ~ Privileges, Learning, Raises, (Learning ^ 2)
6. Rating ~ Privileges, Learning, Raises, (Raises ^ 2)

Модель №6 является наилучшей: R2 самый высокий(48.6%), уровень значимости всех переменных выше, чем у других моделей, вероятность неправильно посчитанных коэффициентов ниже. В таблице 9 приведены характеристики модели 3. С характеристиками остальных моделей можно ознакомиться в Приложении 7.

Таблица №9: Характеристики модели зависимости параметра Rating от параметров Privileges, Learning, Raises, (Raises ^ 2)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Параметр | Значение | Std.Error | t value | Pr(>|t|) |
| (Intercept) | -51.24402 | 57.01256 | -0.899 | 0.3773 |
| Privileges | 0.10063 | 0.16706 | 0.602 | 0.5524 |
| Learning | 0.38455 | 0.20310 | 1.893 | 0.0699 |
| Raises | 2.40658 | 1.76881 | 1.361 | 0.1858 |
| I(Raises ^ 2) | -0.01557 | 0.01330 | -1.171 | 0.2526 |

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 R2 было самым высоким, относительно других моделей(48.6%). Также стоит отметить, что зависимость является отрицательной по всем параметрам.

Задача 2.2

Набор данных: Attitude

Объясняемая переменная: Rating

Регрессоры: Privileges, Learning, Raises

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:  
**,** 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 = 2.055.

Свободный: [-9.49, 37.81], Privileges: [-0.23, 0.42], Learning: [-0.02, 0.801], Raises: [-0.10, 0.80]

Это означает, что с вероятностью p(95%) истинное значение параметра "Свободный" будет находиться в диапазоне от -9.49до 37.81, Privileges от 0.23 до 0.42 и т.д.

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

Доверительный интервал параметра Свободный, Privileges, Learning, Raises содержит 0, это значит, что мы не можем отвергнуть гипотезу о том, что коэффициент равен 0.

1. Оценить доверительный интервал для прогноза.

Сделаем прогноз по следующим данным: Privileges = 80, Learning = 60, Raises = 40. С помощью команды predict(см Приложении 7) получаем интервал [45.90852, 72.83382] и возможное значение 59.37117.

Заключение

Доверительный интервал свободного коэффициента оказался достаточно большим, однако это означает бо́льшую вероятность попадания в него. Доверительный интервал Свободный, Privileges, Learning, Raises 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: r21i\_os24a.sav – 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 ~ 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 параметровsex, age, wed1, wed2, wed3, higher\_educ, city\_status, working\_hours.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Параметр | Значение | Std.Error | t value | Pr(>|t|) |
| (Intercept) | 6.886e+06 | 1.742e+07 | 0.395 | 0.6949 |
| sex | 3.884e+06 | 1.240e+07 | 0.313 | 0.7558 |
| age | 1.601e-01 | 1.134e-01 | 1.412 | 0.1666 |
| wed1 | 3.474e+06 | 1.552e+07 | 0.224 | 0.8242 |
| wed2 | -1.643e+07 | 2.804e+07 | -0.586 | 0.5616 |
| wed3 | 1.035e+07 | 2.303e+07 | 0.449 | 0.6559 |
| higher\_education | 3.342e+06 | 1.237e+07 | 0.270 | 0.7886 |
| city\_status | -2.415e+07 | 1.258e+07 | -1.921 | 0.0627 |
| working\_hours | -9.984e+02 | 2.289e-01 | -0.436 | 0.6654 |

Multiple R-squared: 0.1379, Adjusted R-square: -0.05368

Таблица №2: Коэффициенты вздутия дисперсии модели wage ~ age, sex, wed1, wed2, wed3, higher\_educ, city\_status, working\_hours

|  |  |  |  |
| --- | --- | --- | --- |
| sex | age | wed1 | wed2 |
| 1.30556 | 1.161522 | 1.935293 | 1.768322 |
| wed3 | higher\_education | city\_status | working\_hours |
| 1.552833 | 1.268070 | 1.117966 | 1.179166 |

Убираем регрессоры sex, wed1, wed2, wed3, higher\_educ, working\_hours из-за плохой p-статистики. VIF всех параметров улучшился и не превышает 1.1, а R2 немного увеличился. Характеристики полученной модели можно наблюдать на таблице 3.

Таблица №3: Характеристики модели зависимости параметра wage от параметров age, city\_status.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Параметр | Значение | Std.Error | t value | Pr(>|t|) |
| (Intercept) | 1.245e+07 | 7.266e+06 | 1.713 | 0.0941 |
| age | 1.355e-01 | 1.022e-01 | 1.326 | 0.1921 |
| city\_status | -2.146e+07 | 1.156e+07 | -1.857 | 0.0704 |

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

Логарифмы:

Введём логарифмы log(working\_hours) и log(age). В результате получаем большие коэффициенты вздутия дисперсии:

Таблица №4: Коэффициенты вздутия дисперсии модели wage ~sec, wed1, wed2, age, working\_hours, log(working\_hours), log(age)

|  |  |  |
| --- | --- | --- |
| sex | Wed1 | Wed2 |
| 1.476054 | 1.346406 | 1.445865 |
| age | working\_hours | log(working\_hours) |
| 1675.357905 | 455.803594 | 462.588667 |
| log(age) |
| 1672.146402 |

Заметим, что working\_hours сильно зависит от логарифма, поэтому уберём его.

Таблица №5: Характеристики модели зависимости параметра wage от параметров sex, wed1, wed2, age, log(working\_hours), log(age)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Параметр | Значение | Std.Error | t value | Pr(>|t|) |
| (Intercept) | 5.776e+07 | 8.982e+07 | 0.643 | 0.524 |
| Sex | 3.367e+06 | 1.222e+07 | 0.275 | 0.784 |
| Wed1 | -6.657e+05 | 1.289e+07 | -0.052 | 0.959 |
| Wed2 | -1.093e+07 | 2.542e+07 | -0.430 | 0.670 |
| Age | 2.234e+00 | 4.121e+00 | 0.542 | 0.591 |
| I(log(working\_hours) | -8.724e+05 | 1.574e+06 | -0.554 | 0.583 |
| I(log(age)) | -1.406e+07 | 2.723e+07 | -0.516 | 0.609 |

Multiple R-Squared: 0.0475, Adjusted R-Squared: -0.1029

Степени:

Построим модель с введёнными параметрами степеней. На таблице №6 отчётливо видим, что коэффициенты вздутия дисперсии принимают адекватные значения, R2 также плохой для нашей модели и значимость регрессоров тоже плохо. Поэкспериментируем и будем изменять значение переменной current\_pow от 0.1 до 2.0 с шагом 0.1(подробнее ознакомиться со значениями каждой регресии можно в Приложении 8). Заметим, что при увеличении степени коэффициенты вздутия дисперсии.

|  |  |  |
| --- | --- | --- |
| sex | Age\_power\_0.1 | Wed1 |
| 1.248309 | 1.085183 | 1.755333 |
| Wed2 | Wed3 | Higheduc\_power\_0.1 |
| 1.563387 | 1.524007 | 1.254498 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Параметр | Значение | Std.Error | t value | Pr(>|t|) |
| (Intercept) | -91559535 | 246486463 | -0.371 | 0.712 |
| sex | 3301888 | 2283360 | 0.266 | 0.792 |
| age\_power\_0.1 | 1972480 | 15141151 | 0.864 | 0.393 |
| wed1 | 1727131 | 15141151 | 0.114 | 0.910 |
| wed2 | -12911085 | 27004320 | -0.478 | 0.635 |
| wed3 | 14461956 | 23369904 | 0.619 | 0.540 |
| highereduc\_power\_0.1 | 70638745 | 187627154 | 0.376 | 0.709 |

Таблица №6 характеристики модели зависимости параметра wage от параметров sex, age\_power\_0.1, wed1, wed2, wed3,highereduc\_power\_0.1

Произведения:

Введём произведение регрессеров – working\_hours \* age и построим данную модель.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Параметр | Значение | Std.Error | t value | Pr(>|t|) |
| (Intercept) | 5.624e+06 | 1.367e+07 | 0.412 | 0.683 |
| sex | 4.086e+06 | 1.151e+07 | 0.355 | 0.724 |
| wed1 | 8.544e+05 | 1.246e+07 | 0.069 | 0.946 |
| age | 1.167e-01 | 1.155e-01 | 1.011 | 0.318 |
| working\_hours | -9.695e-02 | 3.839e-01 | -0.253 | 0.802 |
| working\_hours\*age | -1.005e-09 | 4.661e-09 | -0.216 | 0.830 |

Multiple R-squared: 0.03866, Adjusted R-squared: -0.08459

|  |  |  |
| --- | --- | --- |
| sex | wed1 | age |
| 1.092829 | 1.210800 | 1.170938 |
| working\_hours | working\_hours \* age |  |
| 3.221106 | 3.240285 |  |

Таблица №7: Характеристики модели зависимости параметра wage от параметров sex, wed1, age, working\_hours, working\_hours \* age

в таблице 4 приведены характеристики построенной нами модели. Мы видим, что R2 близок к 0, VIF из параметров отличный. Мы приходим к выводу, что построенная нами модель не является успешной.

1. 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 таблице 8).

|  |  |  |
| --- | --- | --- |
| sex | age\_power\_0.2 | wed1 |
| 1.248775 | 1.085860 | 1.761695 |
| wed2 | wed3 | highereduc\_power\_0.2 |
| 1.567784 | 1.519911 | 1.255323 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Параметр | Значение | Std.Error | t value | Pr(>|t|) |
| (Intercept) | -42177225 | 124699179 | -0.338 | 0.737 |
| Sex | 3301123 | 12416945 | 0.266 | 0.792 |
| age\_power\_0.2 | 257624 | 296555 | 0.869 | 0.390 |
| wed1 | 1761908 | 15168239 | 0.116 | 0.908 |
| wed2 | -12881114 | 27041697 | -0.476 | 0.637 |
| wed3 | 14390453 | 23337980 | 0.617 | 0.541 |
| highereduc\_power\_0.2 | 26422991 | 71855070 | 0.368 | 0.715 |

Multiple R-squared: 0.04522, Adjusted R-squared: -0.1055

Таблица №8: Характеристики модели зависимости параметра wage от параметров sex, age\_power\_0.2, wed1,wed2, wed3, highereduc\_power\_0.2

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

В зависимости от "lm\_model2", который мы создали, включая "женат", "возраст" и "образование", мы можем видеть, что положительная независимость зависит от образования и брака, отрицательная - от возраста и рабочего времени. Из этого мы видим, что молодые женатые мужчины с высшим образованием получают большую зарплату

1. 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.

Давайте воспользуемся нашей лучшей моделью со степенью 0,1 и рассмотрим подмножество людей, которые не состояли в браке и не имеют высшего образования. Давайте отбросим знаки, которые нам не нужны, и построим модель. Основываясь на характеристиках этой модели, представленных в таблице 9, мы приходим к выводу, что лица, имеющие высшее образование, получают большую зарплату в этой подгруппе. Теперь давайте возьмем подгруппу людей, живущих в городах и не состоящих в браке. Давайте снова воспользуемся нашей лучшей моделью. Основываясь на данных, приведенных в таблице 10, мы приходим к выводу, что в этой подгруппе молодые люди с высшим образованием получают большую зарплату.

Таблица №9: Характеристики модели зависимости параметра wage от параметров sex, age, city\_status, age\_power\_0.1, highereduc\_power\_0.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Параметр | Значение | Std.Error | t value | Pr(>|t|) |
| (Intercept) | 3.559e+08 | 3.740e+08 | 0.951 | 0.3472 |
| Sex | 7.882e+06 | 1.074e+07 | 0.734 | 0.4675 |
| Age | 1.174e+01 | 9.560e+00 | 1.228 | 0.2268 |
| city\_status | -2.700e+07 | 1.245e+07 | -2.168 | 0.0363 |
| age\_power\_0.1 | -2.356e+08 | 1.942e+08 | -1.213 | 0.2325 |
| highereduc\_power\_0.1 | -1.488e+07 | 1.604e+08 | -0.093 | 0.9266 |

Таблица №9: Характеристики модели зависимости параметра wage от параметров sex, age, age\_power\_0.1, highereduc\_power\_0.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Параметр | Значение | Std.Error | t value | Pr(>|t|) |
| (Intercept) | 1.449e+08 | 3.775e+08 | 0.384 | 0.703 |
| Sex | 4.765e+06 | 1.113e+07 | 0.428 | 0.671 |
| Age | 4.974e+00 | 9.445e+00 | 0.527 | 0.601 |
| age\_power\_0.1 | -9.909e+07 | 1.921e+08 | -0.516 | -0.516 |
| highereduc\_power\_0.1 | -1.111e+06 | 1.676e+08 | -0.007 | 0.995 |

Заключение

С помощью построенных нам моделей легко оценить множество индивидов, получающих бо́льшую 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: Credit Customers

Классификатор: Logistic Regression

Целевой параметр: Total Transaction Amount(ниже или совпадает с 5000 – класс 0, выше – 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: Customer age, Dependent count, Months on book, total relationship count, months inactive 12 mon, contacts count 12 mon, credit limit, total revolving bal, avg open to but, total amt chng q4 q1 etc. Преобразуем их в целочисленные переменные с помощью one-hot-encoding:

* если total\_transaction\_amt меньше 5000- класс 0,если больше - класс 1

Модель логистической регрессии достигла accuracy {accuracy\_score}, precision {precision\_score}, recall {recall\_score} и оценки F1 {f1\_score} по тестовым данным. Эти показатели дают представление о различных аспектах производительности модели, таких как общая accuracy, precision (способность правильно прогнозировать положительные примеры), recall(способность фиксировать все положительные примеры) и оценка F1 (среднее гармоническое значение точности и отзыва).

Таблица №1: Результаты оценки качества классификатора Logistic Regression с помощью метрик accuracy, precision, recall, f1 score

|  |  |
| --- | --- |
| Метрика | Точность |
| Accuracy | 0.8999670944389602 |
| Precision | 0.8021505376344086 |
| Recall | 0.6376068376068376 |
| F1 score | 0.7104761904761903 |

1. 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 переберать 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, f1 score, recall

|  |  |
| --- | --- |
| Accuracy | 0.9230068065551029 |
| Precision | 0.8998929676964886 |
| F1 | 0.7665570178370613 |
| Recall | 0.6678260869565217 |

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

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. Приложении 9) и прогоняем его по данным параметрам. На выходе получаем лучшие параметры для построения нашего классификатора типа Случайны Лес:

Best Parameters: “‘max\_depth’: None, ‘max\_features’: ‘sqrt’, ‘min\_samples\_leaf’: 2, ‘min\_samples\_split’: 5, ‘n\_estimators’: 200”

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.

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

Заключение

В данной работе были построены классификаторы Logistic Regression и Random Forest для решения задачи классификации по целевому параметру Total\_Transaction\_Amt, где прогнозируемая категория задавалась условием: 0, если значение Total\_Transaction\_Amt меньше или равно 5000, и 1 в противном случае. Оба классификатора справились с задачей на высоком уровне с точностью не менее 90%. Однако классификатор Logistic Regression показал немного лучшие результаты (такие выводы сделаны на основании оценок метрик accuracy, precision, recall и F1), и время его обучения было меньше по сравнению с Random Forest. С другой стороны, классификатор Random Forest показал хорошие результаты, но требовал больше времени на обучение и оптимизацию гиперпараметров, при этом лучшее значение для n\_estimators (количество деревьев в ансамбле) оказалось равным 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.

*Bias(b):*

Bias или же смещение, определяет насколько смещен относительно опорных точек, например, если или сплайн будет смещен сильнее в сторону следующей или предыдущей опорной точки, тем самым переход между опорными точками так же будет более резким.

*Continuity(c):*

Continuity или же непрерывность, определяет то, насколько гладко и плавно сплайн переходи 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.

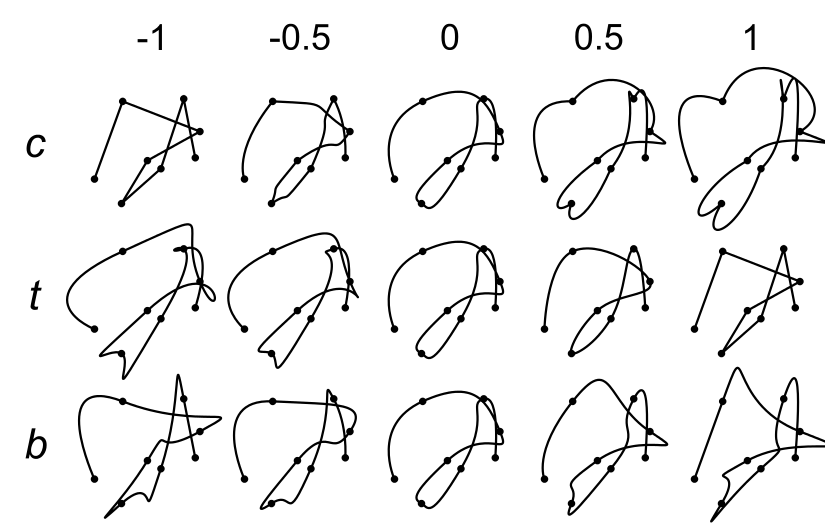


Рисунок 1. Влияние параметров на внешний вид сплайна.

## 2. Построение сплайновой кривой:

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

Пусть задано + 1 узлов, где , тогда для интерполяции кривой Эрмита у нас есть начальная точка и конечная точка , где начальная касательная это , а конечная касательная это , тогда вычисление касательных будем осуществлять по следующим формулам:

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 |
|  |  |
|  |  |
|  |  |
|  |  |

Сам же интерполяционный многочлен имеет следующий вид:

,

где и ― начальная и конечная точка, а и ― 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 , тогда t будет выглядеть следующим образом:

,

где x – значение аргумента сплайна, а и ― 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 и .
2. Вычисляем значения базисных полиномов Эрмита для параметров t.
3. Делаем подсчёт касательных к точкам.
4. Считаем значение функции .

## 4. Реализация

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

Создадим класс *Kochanek\_Bartels\_Spline* со следующими полями:

* int num\_of\_points; — количество точек;
* float tension; — натяжение сплайна;
* float bias; — смещение сплайна;
* float continuity; — непрерывность сплайна;
* 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, 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. Также в главной функции происходит указание на все прописанные нами функции, необходимые для отображения.

С полный 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 |
| ***x1*** | -3.5 | 0.0 | -2.0 |  | ***x5*** | 3.6 | 0.0 | 2.0 |
| ***x2*** | -1.0 | 1.5 | 3.5 |  | ***x6*** | 1.0 | -1.5 | 1.0 |
| ***x3*** | 0.42 | 0.1 | 1.0 |  | ***x7*** | 0.12 | -0.1 | -1.0 |
| ***x4*** | 1.2 | 1.5 | -2.5 |  | ***x8*** | -1.69 | -1.5 | 2.0 |

Результат выполнения программы можем видеть на рисунках 2-4.

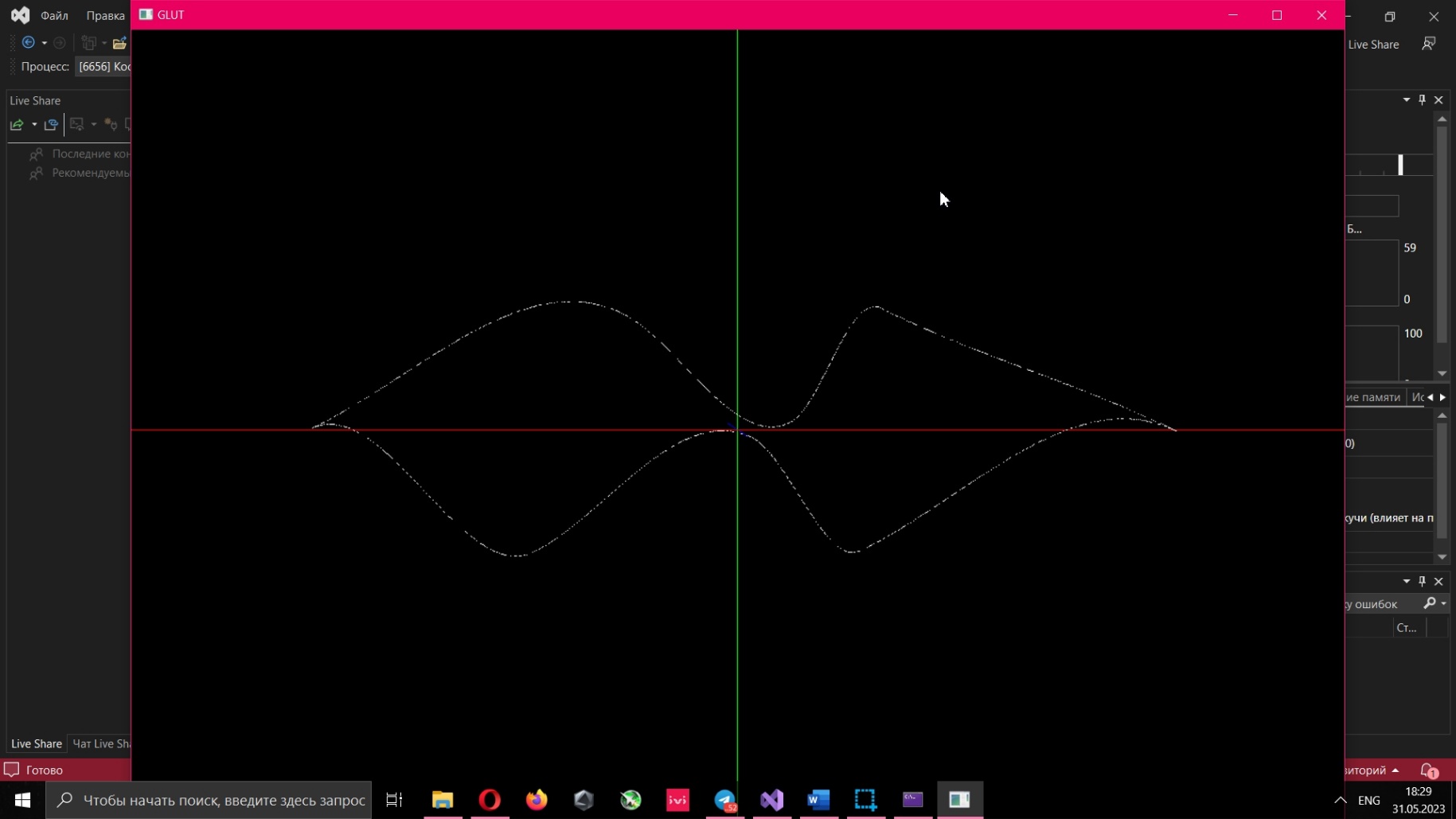


Рисунок 2. Проекция тестового сплайна на ось Oyz

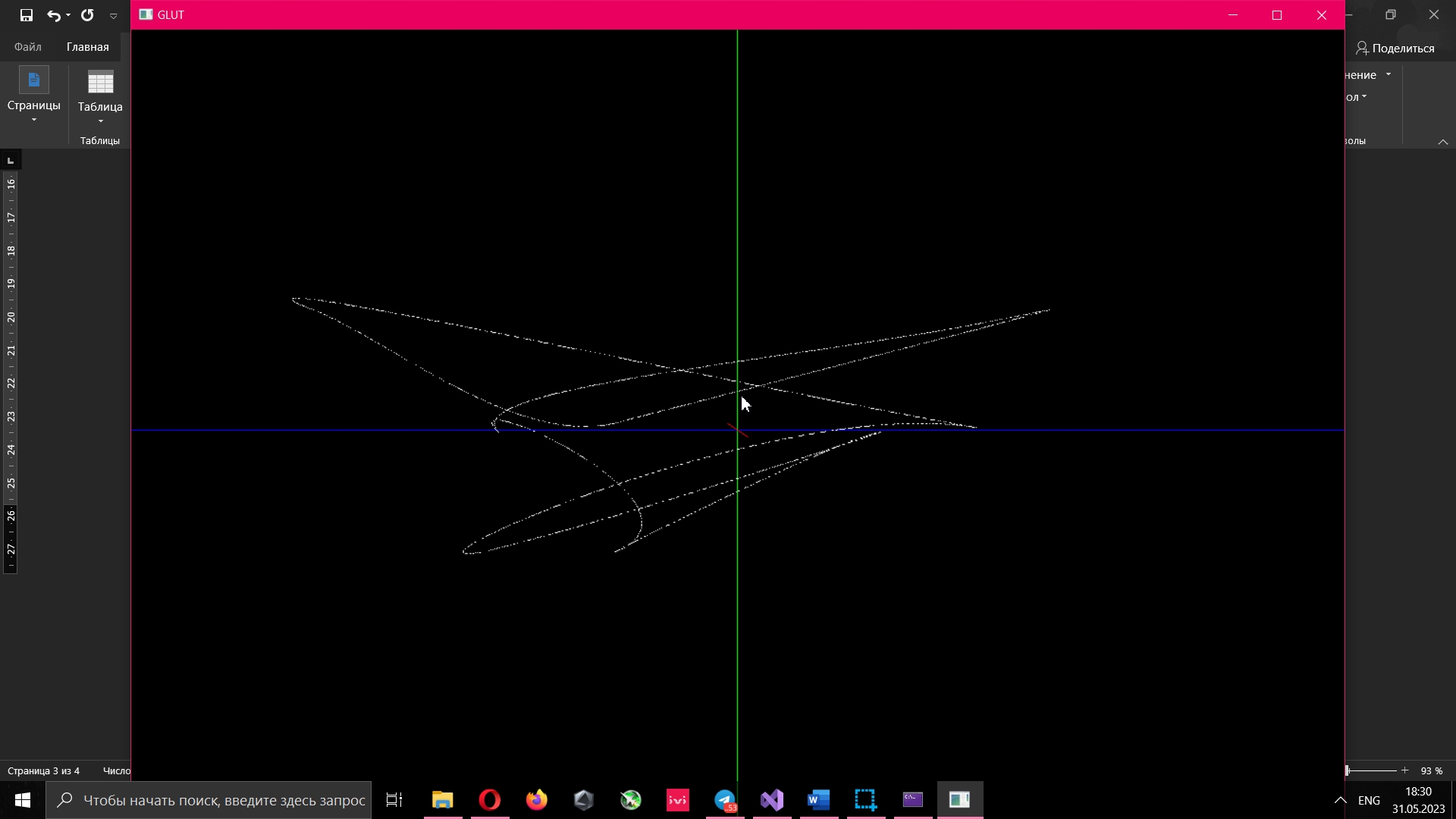


Рисунок 3. Проекция тестового сплайна на ось Oxz

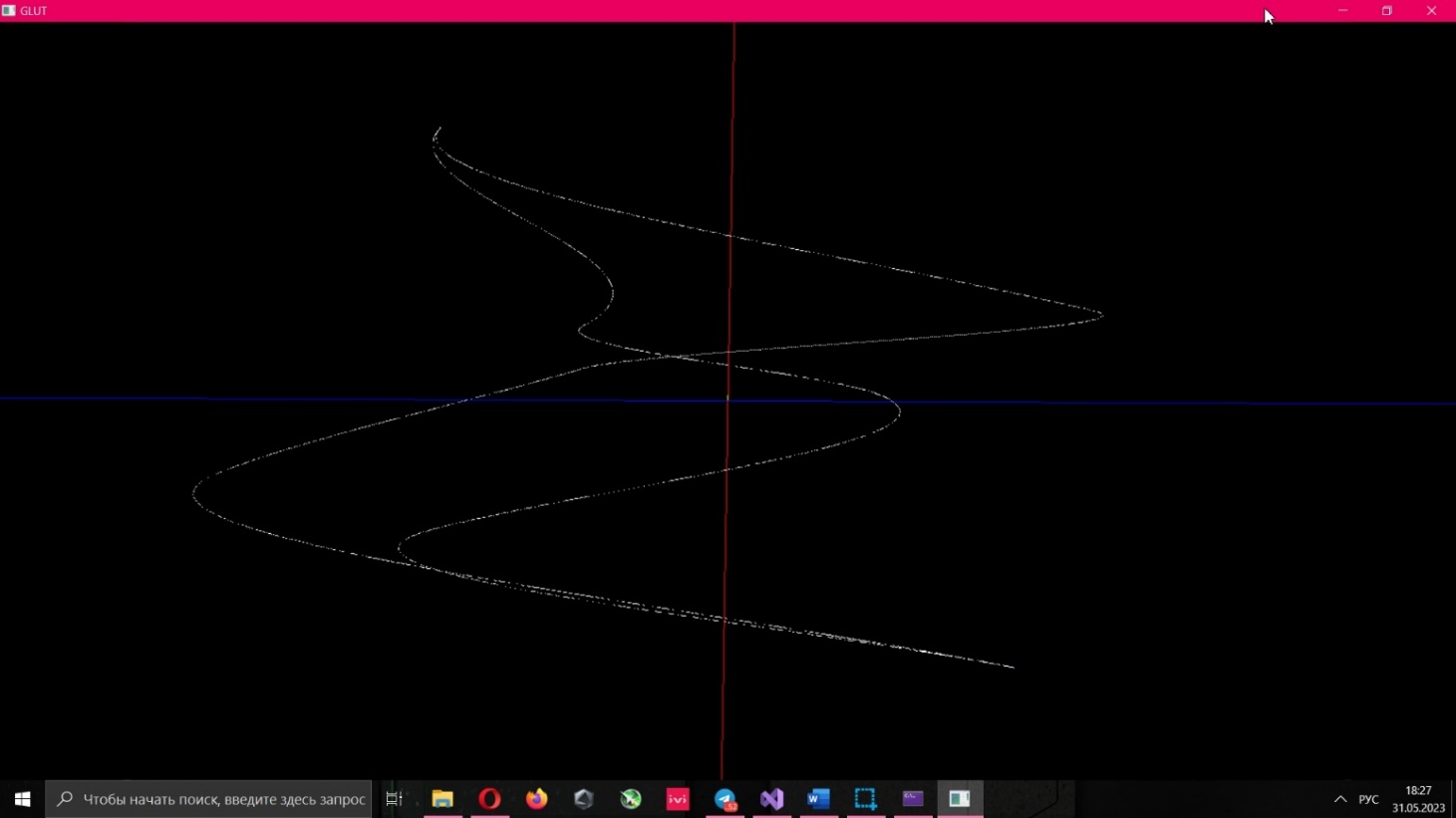


Рисунок 4. Проекция тестового сплайна на ось Oxy

# Заключение

Было предложено и реализовано решение задачи по вычислению и построению сплайнов Коханек-Бартельса. Данный алгоритм работает за время O(n\*m), где n — количество точек, m — размерность пространства, что является отличным(эффективным) временным показателем. Погрешность отображения сплайнов зависит от выбранного разбиения сплайна, погрешность при подсчётах полностью отсутствует.

**Приложение**

Приложение №1: Подсчёт Среднего значения, дисперсии и СКО для параметров Infant.Mortality, Agriculture, Examination в наборе данных Swiss

library("lmtest")

library("GGally")

data = swiss

help(swiss)

data

summary(data)

ggpairs(data)

#Считаем Среднее значение, Дисперсия и СКО для Infant.Mortality, Agriculture и Examination

mean(swiss$Infant.Mortality)

var(swiss$Infant.Mortality)

sd(swiss$Infant.Mortality)

mean(swiss$Agriculture)

var(swiss$Agriculture)

sd(swiss$Agriculture)

mean(swiss$Examination)

var(swiss$Examination)

sd(swiss$Examination)

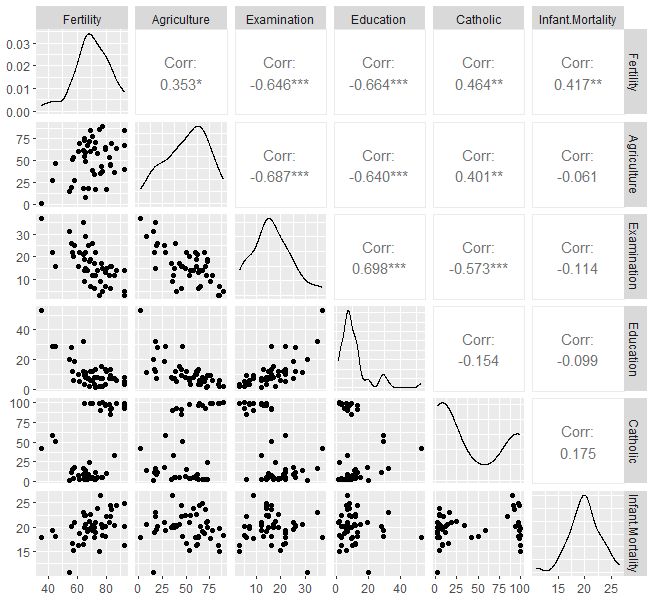
Приложение №2: Построение линейной регрессии Infant.Mortality ~ Agriculture и Infant.Mortality ~ Examination в наборе данных swiss.

model1 <- lm(Infant.Mortality~Agriculture, data = swiss)

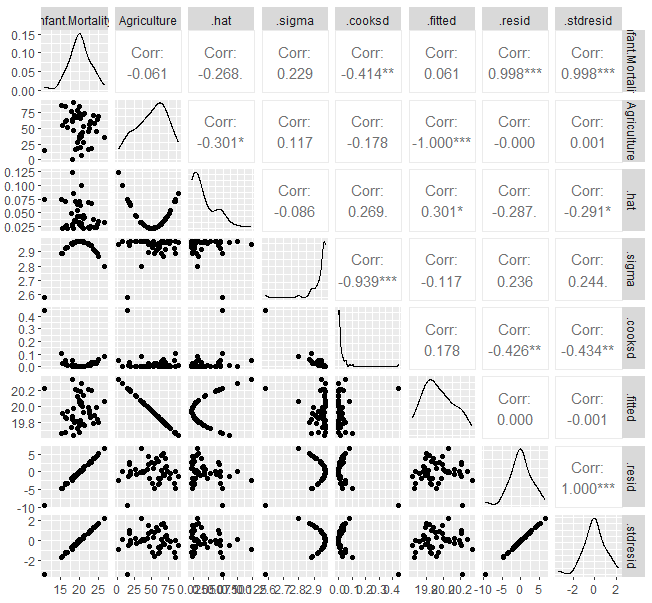
summary(model1)

model2 <- lm(Infant.Mortality~Examination, data = swiss)

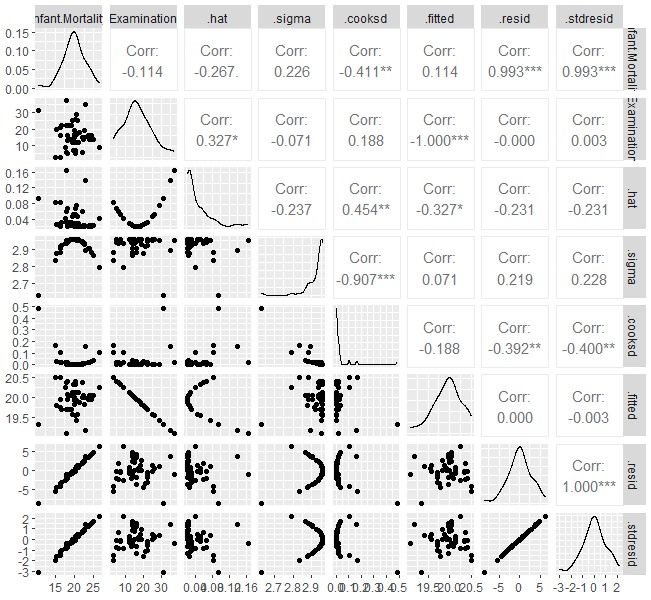
summary(model2)

Приложение №3: Графическое представление регрессии набора данных swiss.

Приложение №4: Графическое представление регрессии Infant.Mortality ~ Agriculture в наборе данных swiss.



Приложение №5: Графическое представление регрессии Infant.Mortality ~ Examination в наборе данных swiss



Приложение №6: Характеристики моделей с Infant.Mortality ~ Agriculture и Infant.Mortality ~ Examination в наборе данных swiss.

Рисунок №1: Характеристики с Infant.Mortality ~ Agriculture

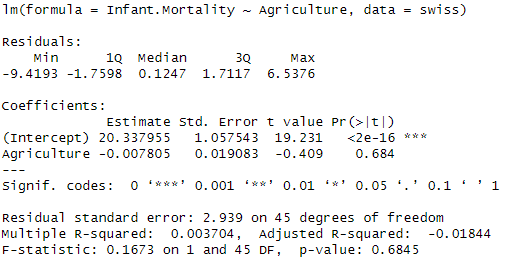
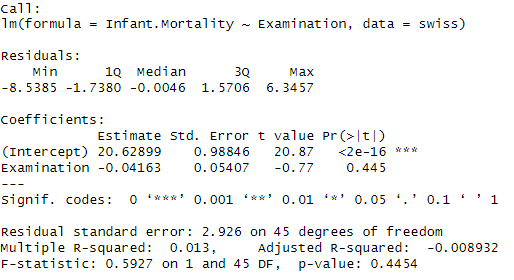


Рисунок №2: Характеристики с Infant.Mortality ~ Examination



Приложение №7: Код для решения задач 2 и 2.2

library("lmtest")

library("GGAlly")

library("car")

data = attitude

data

summary(data)

ggpairs(data)

model = lm(rating ~ privileges + learning + raises, data = data)

vif(model)

summary(model1)

model\_log1 = lm(rating ~ privileges + learning + raises + I(log(privileges)), data = data)

summary(model\_log1)

model\_log2 = lm(rating ~ privileges + learning + raises + I(log(learning)), data = data)

summary(model\_log2)

model\_log3 = lm(rating ~ privileges + learning + raises + I(log(raises)), data = data)

summary(model\_log3)

cmpmodel1 = lm(rating ~ privileges + learning + raises + I(privileges \* learning), data = data)

summary(cmpmodel1)

cmpmodel2 = lm(rating ~ privileges + learning + raises + I(privileges \* raises), data = data)

summary(cmpmodel2)

cmpmodel3 = lm(rating ~ privileges + learning + raises + I(raises \* learning), data = data)

summary(cmpmodel3)

sqmodel1 = lm(rating ~ privileges + learning + raises + I(privileges^2), data = data)

summary(sqmodel1)

sqmodel2 = lm(rating ~ privileges + learning + raises + I(learning^2), data = data)

summary(sqmodel2)

sqmodel3 = lm(rating ~ privileges + learning + raises + I(raises^2), data = data)

summary(sqmodel3)

#2.2

t\_critical = qt(0.975, df = 26)

t\_critical #t = 2.055

model1 <- lm(rating ~ privileges+raises+learning, data = data)

confint(model1, level = 0.95)

summary(model1)

new.data = data.frame(privileges=80, raises=60, learning=40)

predict(model1, new.data, interval="confidence", level=0.95)

Приложение №8: код для решения задачи 3

install.packages("devtools")

devtools::install\_github("https://github.com/bdemeshev/rlms")

library("lmtest")

library("rlms")

library("dplyr")

library("GGally")

library("car")

library("sandwich")

library("foreign")

library("haven")

library("devtools")

df <- read\_sav("r21i\_os24a.sav")

data = select(df, qj13.2, qh5, q\_educ, status, qj6.2, q\_marst, qm148\_a)

names(df)

data = na.omit(data)#remove missing values

#qh5-gender, q\_educ education, status city status, qj13.2 salary, qj6.2 work time in a week,q\_marst marital status, qm148\_a age

data$wed1 <- as.numeric(data$q\_marst == 2)

data$wed2 <- as.numeric(data$q\_marst %in% c(4, 5))

data$wed3 <- as.numeric(data$q\_marst == 1)

data$city\_status <- as.numeric(data$status == 2)

data$status[which(data$status!=2)] <- 0

data$status[which(data$status==2)] <- 1

data$sex <- as.numeric(data$qh5 == 1)

data$qh5[which(data$qh5!=1)] <- 0

data$qh5[which(data$qh5==1)] <- 1

data$higher\_education <- as.numeric(data$q\_educ >= 21)

data$wage = data$qj13.2

data$age = data$qm148\_a

data$working\_hours = data$qj6.2

#1 Build linear regression and adding VIF

lm\_model <- lm(data$wage ~ wed1 + q\_educ + age, data = data)

vif(lm\_model)

summary(lm\_model)

lm\_model2 <- lm(data$wage ~ sex + age + wed1 + wed2 + wed3 + higher\_education + city\_status + working\_hours, data =data)

vif(lm\_model2)

summary(lm\_model2)

lm\_model3 <- lm(data$wage ~ age + city\_status, data = data)

vif(lm\_model3)

summary(lm\_model3)

#2 Experiment with the functions of real parameters: use logarithms, powers

logmodel1 <- lm(wage ~ sex + wed1 + wed2 + age + working\_hours + I(log(working\_hours)) + I(log(age)), data = data)

vif(logmodel1)

summary(logmodel1)

logmodel2 <- lm(wage ~ sex + wed1 + wed2 + age + I(log(working\_hours)) + I(log(age)), data = data)

vif(logmodel2)

summary(logmodel2)

data$age\_power\_0.1 <- data$age^0.1

data$higherduc\_power\_0.1 <- data$q\_educ^0.1

data$age\_power\_0.2 <- data$age^0.2

data$higherduc\_power\_0.2 <- data$q\_educ^0.2

data$age\_power\_0.3 <- data$age^0.3

data$higherduc\_power\_0.3 <- data$q\_educ^0.3

data$age\_power\_0.4 <- data$age^0.4

data$higherduc\_power\_0.4 <- data$q\_educ^0.4

data$age\_power\_0.5 <- data$age^0.5

data$higherduc\_power\_0.5 <- data$q\_educ^0.5

data$age\_power\_0.6 <- data$age^0.6

data$higherduc\_power\_0.6 <- data$q\_educ^0.6

data$age\_power\_0.7 <- data$age^0.7

data$higherduc\_power\_0.7 <- data$q\_educ^0.7

data$age\_power\_0.8 <- data$age^0.8

data$higherduc\_power\_0.8 <- data$q\_educ^0.8

data$age\_power\_0.9 <- data$age^0.9

data$higherduc\_power\_0.9 <- data$q\_educ^0.9

data$age\_power\_1.0 <- data$age^1.0

data$higherduc\_power\_1.0 <- data$q\_educ^1.0

data$age\_power\_1.1 <- data$age^1.1

data$higherduc\_power\_1.1 <- data$q\_educ^1.1

data$age\_power\_1.2 <- data$age^1.2

data$higherduc\_power\_1.2 <- data$q\_educ^1.2

data$age\_power\_1.3 <- data$age^1.3

data$higherduc\_power\_1.3 <- data$q\_educ^1.3

data$age\_power\_1.4 <- data$age^1.4

data$higherduc\_power\_1.4 <- data$q\_educ^1.4

data$age\_power\_1.5 <- data$age^1.5

data$higherduc\_power\_1.5 <- data$q\_educ^1.5

data$age\_power\_1.6 <- data$age^1.6

data$higherduc\_power\_1.6 <- data$q\_educ^1.6

data$age\_power\_1.7 <- data$age^1.7

data$higherduc\_power\_1.7 <- data$q\_educ^1.7

data$age\_power\_1.8 <- data$age^1.8

data$higherduc\_power\_1.8 <- data$q\_educ^1.8

data$age\_power\_1.9 <- data$age^1.9

data$higherduc\_power\_1.9 <- data$q\_educ^1.9

data$age\_power\_2.0 <- data$age^2.0

data$higherduc\_power\_2.0 <- data$q\_educ^2.0

lm\_model2\_0.1 <- lm(qj13.2 ~ sex + age\_power\_0.1 + wed1 + wed2 + wed3 + higherduc\_power\_0.1, data = data)

vif(lm\_model2\_0.1)

summary(lm\_model2\_0.1)

#Multiple R-squared: 0.04518

#Adjusted R-squared: -0.1056

lm\_model2\_0.2 <- lm(qj13.2 ~ sex + age\_power\_0.2 + wed1 + wed2 + wed3 + higherduc\_power\_0.2, data = data)

vif(lm\_model2\_0.2)

summary(lm\_model2\_0.2)

#Multiple R-squared: 0.0447

#Adjusted R-squared: -0.1061

lm\_model2\_0.3 <- lm(qj13.2 ~ sex + age\_power\_0.3 + wed1 + wed2 + wed3 + higherduc\_power\_0.3, data = data)

vif(lm\_model2\_0.3)

summary(lm\_model2\_0.3)

#Multiple R-squared: 0.04522

#Adjusted R-squared: -0.1055

lm\_model2\_0.4 <- lm(qj13.2 ~ sex + age\_power\_0.4 + wed1 + wed2 + wed3 + higherduc\_power\_0.4, data = data)

vif(lm\_model2\_0.4)

summary(lm\_model2\_0.4)

#Multiple R-squared: 0.04506

#Adjusted R-squared: -0.1057

lm\_model2\_0.5 <- lm(qj13.2 ~ sex + age\_power\_0.5 + wed1 + wed2 + wed3 + higherduc\_power\_0.5, data = data)

vif(lm\_model2\_0.5)

summary(lm\_model2\_0.5)

#Multiple R-squared: 0.04495

#Adjusted R-squared: -0.1058

lm\_model2\_0.6 <- lm(qj13.2 ~ sex + age\_power\_0.6 + wed1 + wed2 + wed3 + higherduc\_power\_0.6, data = data)

vif(lm\_model2\_0.6)

summary(lm\_model2\_0.6)

#Multiple R-squared: 0.04483

#Adjusted R-squared: -0.106

lm\_model2\_0.7 <- lm(qj13.2 ~ sex + age\_power\_0.7 + wed1 + wed2 + wed3 + higherduc\_power\_0.7, data = data)

vif(lm\_model2\_0.7)

summary(lm\_model2\_0.7)

#Multiple R-squared: 0.04471

#Adjusted R-squared: -0.1061

lm\_model2\_0.8 <- lm(qj13.2 ~ sex + age\_power\_0.8 + wed1 + wed2 + wed3 + higherduc\_power\_0.8, data = data)

vif(lm\_model2\_0.8)

summary(lm\_model2\_0.8)

#Multiple R-squared: 0.0446

#Adjusted R-squared: -0.1063

lm\_model2\_0.9 <- lm(qj13.2 ~ sex + age\_power\_0.9 + wed1 + wed2 + wed3 + higherduc\_power\_0.9, data = data)

vif(lm\_model2\_0.9)

summary(lm\_model2\_0.9)

#Multiple R-squared: 0.04449

#Adjusted R-squared: -0.1071

lm\_model2\_1.0 <- lm(qj13.2 ~ sex + age\_power\_1.0 + wed1 + wed2 + wed3 + higherduc\_power\_1.0, data = data)

vif(lm\_model2\_1.0)

summary(lm\_model2\_1.0)

#Multiple R-squared: 0.04372

#Adjusted R-squared: -0.1064

lm\_model2\_1.1 <- lm(qj13.2 ~ sex + age\_power\_1.1 + wed1 + wed2 + wed3 + higherduc\_power\_1.1, data = data)

vif(lm\_model2\_1.1)

summary(lm\_model2\_1.1)

#Multiple R-squared: 0.04427

#Adjusted R-squared: -0.1066

lm\_model2\_1.2 <- lm(qj13.2 ~ sex + age\_power\_1.2 + wed1 + wed2 + wed3 + higherduc\_power\_1.2, data = data)

vif(lm\_model2\_1.2)

summary(lm\_model2\_1.2)

#Multiple R-squared: 0.04417

#Adjusted R-squared: -0.1067

lm\_model2\_1.3 <- lm(qj13.2 ~ sex + age\_power\_1.3 + wed1 + wed2 + wed3 + higherduc\_power\_1.3, data = data)

vif(lm\_model2\_1.3)

summary(lm\_model2\_1.3)

#Multiple R-squared: 0.04407

#Adjusted R-squared: -0.1069

lm\_model2\_1.4 <- lm(qj13.2 ~ sex + age\_power\_1.4 + wed1 + wed2 + wed3 + higherduc\_power\_1.4, data = data)

vif(lm\_model2\_1.4)

summary(lm\_model2\_1.4)

#Multiple R-squared: 0.04397

#Adjusted R-squared: -0.107

lm\_model2\_1.5 <- lm(qj13.2 ~ sex + age\_power\_1.5 + wed1 + wed2 + wed3 + higherduc\_power\_1.5, data = data)

vif(lm\_model2\_1.5)

summary(lm\_model2\_1.5)

#Multiple R-squared: 0.04388

#Adjusted R-squared: -0.1071

lm\_model2\_1.6 <- lm(qj13.2 ~ sex + age\_power\_1.6 + wed1 + wed2 + wed3 + higherduc\_power\_1.6, data = data)

vif(lm\_model2\_1.6)

summary(lm\_model2\_1.6)

#Multiple R-squared: 0.04379

#Adjusted R-squared: -0.1072

lm\_model2\_1.7 <- lm(qj13.2 ~ sex + age\_power\_1.7 + wed1 + wed2 + wed3 + higherduc\_power\_1.7, data = data)

vif(lm\_model2\_1.7)

summary(lm\_model2\_1.7)

#Multiple R-squared: 0.0437

#Adjusted R-squared: -0.1073

lm\_model2\_1.8 <- lm(qj13.2 ~ sex + age\_power\_1.8 + wed1 + wed2 + wed3 + higherduc\_power\_1.8, data = data)

vif(lm\_model2\_1.8)

summary(lm\_model2\_1.8)

#Multiple R-squared: 0.04361

#Adjusted R-squared: -0.1074

lm\_model2\_1.9 <- lm(qj13.2 ~ sex + age\_power\_1.9 + wed1 + wed2 + wed3 + higherduc\_power\_1.9, data = data)

vif(lm\_model2\_1.9)

summary(lm\_model2\_1.9)

#Multiple R-squared: 0.04353

#Adjusted R-squared: -0.1075

lm\_model2\_2.0 <- lm(qj13.2 ~ sex + age\_power\_2.0 + wed1 + wed2 + wed3 + higherduc\_power\_2.0, data = data)

vif(lm\_model2\_2.0)

summary(lm\_model2\_2.0)

#Multiple R-squared: 0.04345

#Adjusted R-squared: -0.1076

data$workhours\_times\_age <- data$working\_hours \* data$age

timesmodel <- lm(wage ~ sex + wed1 + age + working\_hours +I(working\_hours \* age), data = data)

vif(timesmodel)

summary(timesmodel)

model1\_0.1 <- lm(wage ~ sex + age + city\_status + age\_power\_0.1 + higherduc\_power\_0.1, data = data)

summary(model1\_0.1)

model2\_0.1 <- lm(wage ~ sex + age + age\_power\_0.1 + higherduc\_power\_0.1, data = data)

summary(model2\_0.1)

#3 Choose the best model from the last models

summary(lm\_model2\_0.1)#this model is better

#4 Make a conclusion about which individuals receive the highest salary.

coef(lm\_model2)#If the coef of the lm\_model2 is positive, depending on education and age the salary increases.

#5 Subset the data for urban, unmarried/divorced women with higher education

subset\_data <- filter(data, city\_status == 1 & wed1 == 0 & wed2 == 1 & higher\_education == 1)

lm\_subset\_data <- lm(qj13.2 ~ city\_status + q\_educ + q\_marst, data=subset\_data)

coef(lm\_subset\_data)

summary(lm\_subset\_data)

Приложение №9: код для решения задачи 4:

import pandas as pd

import numpy as np

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import GridSearchCV

from sklearn.model\_selection import cross\_val\_score

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import classification\_report

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score

df = pd.read\_csv("./BankChurners.csv")

df.head()

df = df.drop(["CLIENTNUM", "Attrition\_Flag", "Gender", "Education\_Level",

             "Marital\_Status", "Income\_Category", "Card\_Category"], axis=1)

df.info()

df.describe()

X = df.drop(['Total\_Trans\_Amt'], axis=1)

y = df['Total\_Trans\_Amt'].apply(lambda v: 0 if v <= 5000 else 1)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3)

model = LogisticRegression(max\_iter = 500)

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred)

recall = recall\_score(y\_test, y\_pred)

f1 = f1\_score(y\_test, y\_pred)

print(f"""

    Accuracy: {accuracy}

    Precision: {precision}

    Recall: {recall}

    F1 score: {f1}

""")

Приложение №10: оценка качества классификатора типа RandomForest для набора данных BankChurners(GridSearch для RandomForest)

rf = RandomForestClassifier(random\_state = 42)

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"],

}

grid\_search\_rf = GridSearchCV(rf,param\_grid)

grid\_search\_rf.fit(X\_train, y\_train)

prediction = grid\_search\_rf.predict(X\_test)

print(classification\_report(y\_test, prediction))

print( "accuracy:"+str(np.average(cross\_val\_score(grid\_search\_rf, X\_test, y\_test, scoring=  'accuracy'))))

print(       "f1:"+str(np.average(cross\_val\_score(grid\_search\_rf, X\_test, y\_test, scoring=  'f1'))))

print("precision:"+str(np.average(cross\_val\_score(grid\_search\_rf, X\_test, y\_test, scoring=  'precision'))))

print(   "recall:"+str(np.average(cross\_val\_score(grid\_search\_rf, X\_test, y\_test, scoring=  'recall'))))

print("Best Parameters:", grid\_search\_rf.best\_params\_)

Приложение №11: Количества деревьев с шагом 50 и 10

param\_grid\_50 = {

    'n\_estimators' : [i for i in range(50, 301, 50)]

}

rfc\_50 = RandomForestClassifier(random\_state = 42)

grid\_search\_50 =  GridSearchCV(estimator = rfc\_50, param\_grid = param\_grid\_50, cv =3)

grid\_search\_50.fit(X\_train, y\_train)

print("Best n\_estimators for step 50: ", grid\_search\_50.best\_params\_['n\_estimators'])

rfc\_50\_pred = grid\_search\_50.predict(X\_test)

rfc\_50\_accuracy = accuracy\_score(y\_test, rfc\_50\_pred)

rfc\_50\_recall = recall\_score(y\_test, rfc\_50\_pred, average = 'weighted')

rfc\_50\_f1 = f1\_score(y\_test, rfc\_50\_pred, average = 'weighted')

print("Accuracy for step 50: ", rfc\_50\_accuracy)

print("Recall for step 50: ", rfc\_50\_recall)

print("F1 for step 50: ", rfc\_50\_f1)

param\_grid\_10 = {

    'n\_estimators' : [i for i in range(50, 301, 10)]

}

rfc\_10 = RandomForestClassifier(random\_state = 42)

grid\_search\_10 =  GridSearchCV(estimator = rfc\_10, param\_grid = param\_grid\_10, cv =3)

grid\_search\_10.fit(X\_train, y\_train)

print("Best n\_estimators for step 10: ", grid\_search\_10.best\_params\_['n\_estimators'])

rfc\_10\_pred = grid\_search\_10.predict(X\_test)

rfc\_10\_accuracy = accuracy\_score(y\_test, rfc\_10\_pred)

rfc\_10\_recall = recall\_score(y\_test, rfc\_10\_pred, average = 'weighted')

rfc\_10\_f1 = f1\_score(y\_test, rfc\_10\_pred, average = 'weighted')

print("Accuracy for step 10: ", rfc\_10\_accuracy)

print("Recall for step 10: ", rfc\_10\_recall)

print("F1 for step 10: ", rfc\_10\_f1)

Приложение №12. Код на языке С++ для решения задачи 5.

#include <GL/glut.h>

#include <iostream>

#include <vector>

// 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;

}

};

// Слпайны

**class** Kochanek\_Bartels\_spline

{

**private**:

// Количество точек

**int** num\_of\_points;

// Натяжение

**float** tension;

// Смещение

**float** bias;

// Непрерывность

**float** continuty;

// 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** **float** \_tension, **const** **float** \_bias, **const** **float** \_continuty)

: num\_of\_points(\_num\_of\_points)

, points(\_points)

, tension(\_tension)

, bias(\_bias)

, continuty(\_continuty)

{ }

// Геттер точек

std::vector<Point> getPoints() **const**

{

**return** points;

}

// p(x) = (h00(t) \* p[k]) + (h10(t) \* (x[k+1] - x[k]) \* m[k]) + (h01(t) \* p[k+1]) + (h11(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()

{

// ((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;

**for** (**int** i = 0; i < points[index].getDim(); ++i)

temp\_coord.push\_back((1 - tension) \* (1 + bias) \* (1 + continuty) \* (points[(index + num\_of\_points) % num\_of\_points].getCoord()[i] - points[(index - 1 + num\_of\_points) % num\_of\_points].getCoord()[i]) / 2 + (1 - tension) \* (1 - bias) \* (1 - continuty) \* (points[(index + 1 + num\_of\_points) % num\_of\_points].getCoord()[i] - points[(index + num\_of\_points) % num\_of\_points].getCoord()[i]) / 2);

Point temp\_point(points[index].getDim(), temp\_coord);

tangent.push\_back(temp\_point);

}

}

// 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}}

};

// 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;

**void** display()

{

Kochanek\_Bartels\_spline test\_spline = Kochanek\_Bartels\_spline(point\_spline.size(), point\_spline, 0.5, -1.0, 0.0);

// Можно поэкспериментировть со значениями tension, bias, continuty(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;

**else**

{ // Иначе вводим размерность пространства, количество точек и их координаты

**int** n, number;

std::cout << "Enter the dimension(dim > 1):";

std::cin >> n;

std::cout << "Enter the number of points(number > 1):";

std::cin >> number;

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.push\_back({ 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 отображание 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](https://www.econometrics-with-r.org/)
2. [Rob J Hyndman and George Athanasopoulos. Forecasting: Principles and Practice](https://otexts.com/fpp2/)
3. [Data Analysis with Python](https://www.freecodecamp.org/learn/data-analysis-with-python/)
4. [Kochanek-Bartels Spline](https://en.wikipedia.org/wiki/Kochanek%E2%80%93Bartels_spline)
5. [Properties of Kochanek-Bartels Spline](https://splines.readthedocs.io/en/latest/euclidean/kochanek-bartels.html)