

Государственное бюджетное общеобразовательное учреждение города Москвы
"Школа № 1532"

Сравнение непараметрических методов моделирования с параметрическими:

Исследование эффективности непараметрических методов (например регрессия, основанная на оценке Надарая-Ватсона) с параметрическими методами (например аппроксимация с подгонкой по функции) для различных выборок.

10 класс, ГБОУ Школа №1532,

Гришин Илья Андреевич

Руководитель: учитель информатики, ГБОУ Школа №1532,

Сергиенко Антон Борисович

Москва, 2024

Оглавление

Введение	0
1. Актуальность работы	0
2. Цели и задачи проекта	0
3. Методика выполнения работы	0
3.1. Подготовка экспериментальных данных	0
3.2. Параметрические методы.....	0
3.3. Непараметрические методы	0
3.4. Сравнение и итоги.....	0
4. Результаты.....	0
5. Выводы	0
6. Список используемой литературы	0
7. Приложения	0

Введение

В работе рассмотрены методы статистического моделирования, на различных выборках. Для поставленных задач применяются регрессионная модель, основанная на непараметрической оценке Надарая-Ватсона, а также аппроксимация с подгонкой по функции.

Современные методы моделирования данных предлагают широкий выбор инструментов для анализа и прогнозирования поведения значений. Среди них мы рассмотрим параметрические и непараметрические методы, каждый из которых имеет свои особенности и актуален в различных ситуациях.

Мы проведём сравнительный анализ эффективности данных методов на различных выборках данных. В ходе сравнительного анализа мы постараемся обеспечить понимание того, какие методы моделирования следует применять при различных условиях.

Актуальность работы

Область анализа данных и статистики стремительно развивается, и поэтому исследования по сравнению методов моделирования остаются актуальными. В ходе данного проекта, где внимание уделяется сравнению статистических методов моделирования, существуют моменты, которые делают данное актуальным:

1. Гибкость и адаптивность методов: выявить закономерность и зависимость методов от структуры данных, сложные данные с высокой погрешностью могут поддаваться более успешному моделированию непараметрическими методами, в то время как параметрические методы могут быть эффективнее в случаях, когда структура данных более предсказуема.
2. Разнообразие областей применения: методы моделирования широко используются в различных областях, таких как экономика, медицина, биология и социальные науки. Сравнение методов на разнообразных выборках позволит сделать исследование более универсальным для различных областей.

3. Оптимизация ресурсов: эффективное использование ресурсов, таких как вычислительная мощность, время и данные, является важным вопросом в современных исследованиях. Понимание, какие методы более эффективны для различных ситуаций, может сэкономить ресурсы и повысить эффективность аналитических процессов.

При наличии всех этих факторов, данное исследование представляет собой актуальный вклад в развивающуюся область анализа данных и статистики.

Цель и задачи проекта

Провести сравнительный анализ между непараметрическими и параметрическими методами моделирования с целью выявления их преимуществ, недостатков и областей применения. Исследование направлено на определение эффективности каждого метода в различных контекстах и создание основы для рекомендаций по выбору подходящего метода в зависимости от конкретных задач и данных.

Были поставлены следующие задачи работы:

1. Создание экспериментальных данных.
2. Реализовать параметрический метод моделирования на языке Python.
3. Реализовать непараметрические методы моделирования на языке Python.
4. Рассмотреть каждый метод для определённой выборки и провести анализ эффективности.
5. Подвести итоги проделанной работы.

Методика выполнения исследования

Первый этап – Подготовка экспериментальных данных

Создадим имитации выборок с различным процентом шума как для 3D, так и для 2D моделирования и запишем их в отдельный файл.

Мы начнем с генерации некоторых случайных точек 2D-данных. Каждый пример будем записывать в отдельный .txt файл с соответствующим названием.

Шум будем рассчитывать по следующей формуле:

$$Y_i = y_i + \text{ran} \left(-\frac{(Y_{\max} - Y_{\min}) * \frac{P}{100}}{2}, \frac{(Y_{\max} - Y_{\min}) * \frac{P}{1000}}{2} \right),$$

где P – процент шума, а Y_i – новое значение с шумом для y_i -го.

Реализуем программу:

```
1 # импортируем необходимые модули
2 import numpy as np
3 import random
4
5 # сгенерируем 100 случайных точек где Y - синус X
6 X = np.linspace(-10, 10, 100)
7 Y = np.sin(X)
8
9 # пройдемся циклом по коэффициентам процентов
10 for proc in range(0, 160, 10):
11     # процент шума = proc
12     # добавим шум в данные по формуле
13     Y2 = [Yi + random.uniform(-(((max(Y) - min(Y)) * proc / 100) / 2),
14                                     (((max(Y) - min(Y)) * proc / 100) / 2)) for Yi in Y]
15     data = np.array([X, Y2]).T
16     # добавим данные в файл, в название укажем процент шума
17     with open(f'dataXY_{round(proc)}.txt', 'w') as f:
18         [print(i, j, file=f) for i, j, in data]
```

Рисунок 1 – Скрипт для генерации данных в 2D пространстве

Теперь аналогично сгенерируем точки в 3D пространстве:

```
1 # импортируем необходимые модули
2 import numpy as np
3 import random
4
5 # генерация 100 случайных точек
6 X = np.linspace(-10, 10, 100)
7 Y = [random.uniform(-10, 10) for _ in range(100)]
8 Z = np.sin(X * Y)
9
10 # пройдемся циклом по коэффициентам процентов
11 for proc in range(0, 160, 10):
12     # процент шума = proc
13     # добавим шум в данные
14     Z1 = [Zi + random.uniform(-(((max(Y) - min(Y)) * proc / 100) / 2),
15                                     (((max(Y) - min(Y)) * proc / 100) / 2)) for Zi in Z]
16     data = np.array([X, Y, Z1]).T
17     # добавим данные в файл, в название укажем процент шума
18     with open(f'dataXYZ_{round(proc)}.txt', 'w') as f:
19         [print(i, j, z, file=f) for i, j, z in data]
```

Рисунок 2 – Скрипт для генерации данных в 3D пространстве

У нас получилось множество файлов с различным уровнем шума, для дальнейшего исследования.

Теперь реализуем функции для удобного получения данных из файла.

```
6 usages
1 ~ def get_data2D(name_file):
2 ~     with open(name_file) as f:
3 ~         a = f.readlines()
4 ~         x, y = [], []
5 ~         for i in a:
6 ~             a, b = i.split()
7 ~             x.append(a)
8 ~             y.append(b)
9 ~         return x, y
10
11
12 ~ # get_data2D(name_file)[0] - X
13 ~ # get_data2D(name_file)[1] - Y
14
15
8 usages
16 ~ def get_data3D(name_file):
17 ~     with open(name_file) as f:
18 ~         a = f.readlines()
19 ~         x, y, z = [], [], []
20 ~         for i in a:
21 ~             a, b, c = i.split()
22 ~             x.append(float(a))
23 ~             y.append(float(b))
24 ~             z.append(float(c))
25
26 ~         return x, y, z
27
28 ~ # get_data3D(name_file)[0] - X
29 ~ # get_data3D(name_file)[1] - Y
30 ~ # get_data3D(name_file)[2] - Z
```

Рисунок 3 – Программа для получения данных

Здесь реализовано две функции для получения 2D и 3D данных. На вход принимается название файла и возвращается массивы с данными для каждой оси, которые можно получить, вызвав функцию и указав индекс необходимого массива, данные для X находятся под индексом 0, для Y по индексом 1.

Второй этап – Параметрические методы

Параметрические методы моделирования — это методы, основанные на использовании параметров и переменных, которые могут быть настроены или изменены в зависимости от моделей или представлений объектов. Эти методы менее устойчивы к шуму в отличие от непараметрических, но показывают более высокую точность при правильном выборе параметров и данных.

Нам необходимо создать функции, которые будет удобно применить к каждому из примеров.

Реализуем аппроксимацию для 2D пространства:

```
1 # импортируем необходимые модули
2 import matplotlib.pyplot as plt
3 import numpy as np
4 from scipy.optimize import curve_fit
5
6
7 # создадим функцию для аппроксимации на вход
8 # которой подаются два массива x и y
9 2 usages
10 def approx_2D(x, y):
11     # Определим математическую функцию, которая
12     # будет использоваться для подгонки кривой.
13     # В этом примере мы будем использовать
14     # синусоидальную функцию.
15     def func(t, A, w, p, c):
16         return A * np.sin(w * t + p) + c
17
18     x = np.array(list(map(float, x)))
19     y = np.array(list(map(float, y)))
20     popt, _ = curve_fit(func, x, y)
21     x = np.linspace(x.min(), x.max(), 100)
22     plt.plot(x, func(x, *popt), color='green',
23             label="Синусоидальная функция")
24     plt.legend(loc='best')
25     plt.scatter(x, y, color='red', s=15)
26     plt.xlabel('x')
27     plt.ylabel('y')
28     plt.show()
```

Рисунок 4 – Функция для аппроксимации в 2D пространстве.

Мы реализовали функцию для аппроксимации в 2D ‘*approx_2D*’.

На вход принимается два массива x и y. Выводится график с данными точками и аппроксимируемой кривой. Кривая подгоняется с помощью библиотеки *scipy* функции *curve_fit*. Функция для подгонки, заданная нами синусоидальная.

Теперь приступим к реализации аппроксимации для 3D пространства:

Создадим, аналогичную прошлой, функцию с некоторыми нюансами

```

1 # импортируем необходимые модули
2 import numpy as np
3 from scipy.optimize import curve_fit
4 import matplotlib.pyplot as plt
5
6 # создадим функцию для аппроксимации на вход которой
7 # подаются три массива X и Y и Z
8 2 usages
9 def approx_3D(x, y, z):
10     x = np.array(list(map(float, x)))
11     y = np.array(list(map(float, y)))
12     z = np.array(list(map(float, z)))
13     # Определение математической функции для
14     # аппроксимации кривой
15     def func(xy, a):
16         x, y = xy
17         return a*np.sin(x)
18     # Выполнить подгонку кривой
19     popt, pcov = curve_fit(func, (x, y), z)
20     # Функция для реализации 3D-графика точек данных
21     # и подобранной кривой
22     fig = plt.figure()
23     ax = fig.add_subplot(111, projection='3d')
24     ax.scatter(x, y, z, color='blue')
25     x_range = np.linspace(-10, 10, 50)
26     y_range = np.linspace(-10, 10, 50)
27     X, Y = np.meshgrid(x_range, y_range)
28     Z = func((X, Y), *popt)
29     ax.plot_surface(X, Y, Z, color='red', alpha=0.5)
30     ax.set_xlabel('X')
31     ax.set_ylabel('Y')
32     ax.set_zlabel('Z')
33     plt.show()

```

Рисунок 5 – Функция для аппроксимации в 3D пространстве.

На вход принимается уже три массива x , y и z . Выводится также график с данными точками и аппроксимирующим рельефом. Подгоняется с помощью библиотеки *scipy* функции *curve_fit*. Функция для подгонки, заданная нами также синусоидальная.

Третий этап – Непараметрические методы

Непараметрические методы моделирования — это статистические методы, которые не требуют предположения о распределении данных или наличия определенных параметров. В отличие от параметрических они основываются на анализе самих данных и являются более гибкими и универсальными, но зачастую

показывают меньшую точность. Эти методы используются для изучения связей между переменными, обнаружения шаблонов и трендов в данных, предсказания будущих значений и других аналитических задач.

В данном случае детально рассмотрим непараметрическую оценку регрессии Надарая-Ватсона.

Формула для непараметрической оценки регрессии Надарая-Ватсона

$$y_{dop}(x_{dop}) = \frac{\sum_{i=1}^n y_{pi} \cdot \Phi\left(\frac{x_{dop} - x_{pi}}{c}\right)}{\sum_{i=1}^n \Phi\left(\frac{x_{dop} - x_{pi}}{c}\right)},$$

где в данном случае, c — коэффициент размытия, а $\Phi(p)$ — колоколообразная функция, равная следующему значению:

$$\Phi(p) = \begin{cases} 0.335 - 0.067 \cdot p^2, & \text{if } p^2 \leq 5 \\ 0, & \text{else} \end{cases}$$

Напишем функцию по формуле Надарая-Ватсона:

```
1  # колоколообразная функция
1  usage
2  def phi(p):
3      if p ** 2 <= 5:
4          return 0.335 - 0.067 * p ** 2
5      else:
6          return 0
7
8  # функция для вычисления значения по формуле Надарая-Ватсона
2  usages
9  def f_nadaray_watson(x_dop, x_pi_list, y_pi_list, c):
10     numerator = 0
11     denominator = 0
12
13     for i in range(len(x_pi_list)): # вычисление весов точек
14         phi_value = phi((x_dop - x_pi_list[i]) / c)
15         numerator += y_pi_list[i] * phi_value
16         denominator += phi_value
17
18     return numerator / denominator # оценка значения в заданной точке
```

Реализуем непараметрический метод моделирования для 2D пространства:

```
1 # импортируем необходимые модули
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from Nad_Wat import f_nadaray_watson
5
6
7 # создадим функцию, которая будет реализовывать непараметрическую регрессию,
8 # на вход принимаются два массива X и Y
9 2 usages
10 def nep_regression_2D(x, y):
11     x = np.array(list(map(float, x)))
12     y = np.array(list(map(float, y)))
13
14     c = 0.5
15     # Вычисляем оценки значений в заданных точках
16     query_y = [f_nadaray_watson(q, x, y, c) for q in x]
17
18     # График
19     plt.scatter(x, y, label='Исходные данные')
20     plt.plot(x, query_y, label='Оценка Надарая-Ватсона', color='blue')
21     plt.legend()
22     plt.show()
```

Рисунок 6 – Функция для непараметрической регрессии в 2D

Реализуем непараметрические методы моделирования для 3D пространства:

```
1 # импортируем необходимые модули
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5
6 # Функция для вычисления оценки Надарая-Ватсона
7 usage
8 def nadaraya_watson(X, Z, x_query, h):
9     weights = np.exp(-np.sum((X - x_query) ** 2, axis=1) / (2 * h ** 2))
10    weighted_sum = np.sum(weights * Z)
11    sum_of_weights = np.sum(weights)
12    return weighted_sum / sum_of_weights
13
14 # функция для построения непараметрической регрессии
15 2 usages
16 def nep_regression_3D(x, y, z):
17     X = np.array([x, y, z]).T
18     # Задание сетки для построения непараметрической регрессии по координате Z
19     grid_size = 0.1
20     x_range = np.arange(-10, 10, grid_size)
21     y_range = np.arange(-10, 10, grid_size)
22     X_grid, Y_grid = np.meshgrid(x_range, y_range)
23     Z_grid = np.zeros_like(X_grid)
24
25     # Вычисление значений по координате с помощью оценки регрессии Надарая-Ватсон
26     for i in range(X_grid.shape[0]):
27         for j in range(X_grid.shape[1]):
28             x_query = [X_grid[i, j], Y_grid[i, j], 0] # Поиск координаты по Z
29             Z_grid[i, j] = nadaraya_watson(X, z, x_query, h=0.1) # Вычисление
30                                                                    # значения на поверхности
31
32     # Построение графика в 3D
33     fig = plt.figure()
34     ax = fig.add_subplot(111, projection='3d')
35     ax.scatter(x, y, z, c='blue', label='Исходные данные')
36     ax.plot_surface(X_grid, Y_grid, Z_grid, cmap='gnuplot2_r', alpha=0.5,
37                    label='Регрессия Надарая-Ватсон')
38     ax.set_xlabel('X')
39     ax.set_ylabel('Y')
40     ax.set_zlabel('Z')
41     ax.legend()
```

Рисунок 7 – Функция для непараметрического моделирования в 3D

Описание реализации:

Мы создаём функцию по формуле Надарая-Ватсона, далее проходимся по каждой точке и с помощью функции вычисляем значения, по которым далее строим график.

Четвёртый этап – Сравнение методов

Применим методы к каждому из примеров и выявим их особенности и область применения

Начнём с 2D методов. Напишем программы для запуска каждого метода.

```
1 from approx_2D import approx_2D
2 from get_points import get_data2D
3
4 # get_data2D(name_file)[0] - X
5 # get_data2D(name_file)[1] - Y
6
7 name_file = 'random_dataXY.txt'
8 # name_file = 'dataXY.txt'
9 # name_file = 'dataXY_with_hindrance.txt'
10 x = get_data2D(name_file)[0]
11 y = get_data2D(name_file)[1]
12 approx_2D(x, y)
```

Рисунок 8 – Программа для запуска аппроксимации функции в 2D

```
1 from nep_2D import nep_regression_2D
2 from get_points import get_data2D
3
4 # get_data2D(name_file)[0] - X
5 # get_data2D(name_file)[1] - Y
6
7 name_file = 'random_dataXY.txt'
8 # name_file = 'dataXY.txt'
9 # name_file = 'dataXY_with_hindrance.txt'
10 x = get_data2D(name_file)[0]
11 y = get_data2D(name_file)[1]
12 nep_regression_2D(x, y)
```

Рисунок 9 – Программа для запуска непараметрической регрессии функции в 2D

Рассмотрим результаты:

Для каких-то выводов нам надо провести анализ над нашими моделями.

Один из способов оценить точность регрессионной модели – вычислить среднеквадратическую ошибку, которая является показателем, показывающим нам среднее расстояние между предсказанными значениями из модели и полученными ранее. Формула для нахождения среднеквадратической ошибки (RMSE) следующая:

$$E = \sqrt{\frac{\sum_{i=1}^n (y^*(x_i) - y_i)^2}{n}},$$

где $y^*(x_{pi})$ - прогнозируемое значение для i -го наблюдения в наборе данных, y_{pi} - наблюдаемое значение для i -го наблюдения в наборе данных, n - размер выборки

Напишем функцию для вычисления RMSE в 2D:

```
1 import math
2
3
4 1 usage
5 def rmse(predictions, targets):
6     squared_errors = [(p - t) ** 2 for p, t in zip(predictions, targets)]
7     mean_squared_error = sum(squared_errors) / len(predictions)
8     rmse = math.sqrt(mean_squared_error)
9
10    return rmse
```

В данном случае predictions — массив предсказанных значений, targets — список или массив известных значений.

Создадим таблицу для удобства, с процентом шума выборки, и показателем RMSE для каждого метода:

Таблица 1 – Таблица результатов для 2D

Файлы	Процент шума, %	RMSE для параметрических методов	RMSE для непараметрических методов
dataXY_0	0		
dataXY_10	10		
dataXY_20	20		
dataXY_30	30		
dataXY_40	40		
dataXY_50	50		
dataXY_60	60		
dataXY_70	70		
dataXY_80	80		

dataXY_90	90		
dataXY_100	100		
dataXY_110	110		
dataXY_120	120		
dataXY_130	130		
dataXY_140	140		
dataXY_150	150		

Также для более наглядного представления рассмотрим некоторые графики с 0, 50 и 120 процентами шума:

Можем сказать...

Теперь рассмотрим 3D методы:

```

1 from approx_3D import approx_3D
2 from get_points import get_data3D
3
4 # get_data3D(name_file)[0] - X
5 # get_data3D(name_file)[1] - Y
6 # get_data3D(name_file)[2] - Z
7
8 name_file = 'random_dataXYZ.txt'
9 # name_file = 'dataXYZ.txt'
10 # name_file = 'dataXYZ_with_hindrance.txt'
11 x = get_data3D(name_file)[0]
12 y = get_data3D(name_file)[1]
13 z = get_data3D(name_file)[2]
14 approx_3D(x, y, z)

```

Рисунок 10 – Программа для запуска аппроксимации функции в 3D

```

1 from nep_3D import nep_regression_3D
2 from get_points import get_data3D
3
4 # get_data3D(name_file)[0] - X
5 # get_data3D(name_file)[1] - Y
6 # get_data3D(name_file)[2] - Z
7
8 name_file = 'random_dataXYZ.txt'
9 # name_file = 'dataXYZ.txt'
10 # name_file = 'dataXYZ_with_hindrance.txt'
11 x = get_data3D(name_file)[0]
12 y = get_data3D(name_file)[1]
13 z = get_data3D(name_file)[2]
14 nep_regression_3D(x, y, z)

```

Рисунок 11 – Программа для запуска непараметрической регрессии функции в 3D

Рассмотрим результаты:

Также создадим таблицу для 3D данных

Таблица 1 – Таблица результатов для 2D

Файлы	Процент шума, %	RMSE для параметрических методов	RMSE для непараметрических методов
dataXYZ_0	0		
dataXYZ_10	10		
dataXYZ_20	20		
dataXYZ_30	30		
dataXYZ_40	40		
dataXYZ_50	50		
dataXYZ_60	60		
dataXYZ_70	70		
dataXYZ_80	80		
dataXYZ_90	90		
dataXYZ_100	100		
dataXYZ_110	110		
dataXYZ_120	120		
dataXYZ_130	130		
dataXYZ_140	140		
dataXYZ_150	150		

Также для более наглядного представления рассмотрим некоторые графики с 0, 50 и 120 процентами шума

Можем сказать...

Результаты и обсуждение

Отличия параметрического и непараметрического моделирования:

1. Предположения о распределении данных:

- Параметрическое моделирование: Основано на предположении о конкретной функциональной форме или распределении данных, например, нормальном или экспоненциальном.
- Непараметрическое моделирование: не требует априорных предположений о распределении данных, что делает его более гибким и универсальным.

2. Число параметров модели:

- Параметрическое моделирование: имеет фиксированное число параметров, которые нужно оценить, основываясь на данных.
- Непараметрическое моделирование: Число параметров модели зависит от размера выборки, что позволяет модели гибко адаптироваться к разнообразным формам данных.

3. Устойчивость к выбросам и аномалиям:

- Параметрическое моделирование: может быть чувствительным к выбросам в данных, особенно если выбранная функциональная форма недостаточно гибка.
- Непараметрическое моделирование: более устойчиво к выбросам, так как не предполагает конкретной формы данных и может лучше адаптироваться к аномальным наблюдениям.

4. Интерпретируемость:

- Параметрическое моделирование: часто более легко интерпретируемо, так как параметры модели имеют конкретные смысловые интерпретации.
- Непараметрическое моделирование: может быть менее интерпретируемым из-за отсутствия явных параметров, хотя некоторые методы, такие как ядерная регрессия, могут предоставлять некоторую интерпретируемость.

5. Сложность модели:

- Параметрическое моделирование: часто более простое в понимании и реализации, так как требует определения конкретной функциональной формы.
- Непараметрическое моделирование: может быть более сложным и требовать более высокого уровня алгоритмического понимания для его применения.

Выбор между параметрическим и непараметрическим моделированием зависит от конкретного контекста задачи, характера данных и требований к модели.

Выводы

В данном проекте было проведено исследование эффективности непараметрических методов моделирования, таких как регрессия, основанная на оценке Надарая-Ватсона, с параметрическими методами, например аппроксимация с подгонкой по функции.

Цель исследования заключалась в сравнении этих методов на различных выборках. Для этого были выбраны несколько наборов данных с разной структурой и характером. Затем были применены непараметрические и параметрические методы к каждой выборке, и произведено сравнение результатов.

Результаты исследования показали, что эффективность непараметрических методов может значительно различаться в зависимости от выборки. В некоторых случаях непараметрические методы показали более точные и надежные результаты, особенно если выборка имела сложную структуру или сильные выбросы. Однако в других случаях параметрические методы показали более стабильные и устойчивые результаты.

Таким образом, выбор между непараметрическими и параметрическими методами моделирования должен основываться на характеристиках конкретной выборки и целях исследования. Непараметрические методы могут быть предпочтительными в случаях, когда данные имеют сложную структуру или

несимметричное распределение, в то время как параметрические методы могут быть более подходящими для простых и симметричных выборок.

Однако необходимо отметить, что эффективность методов может зависеть не только от выборки, но и от других факторов, таких как объем выборки, точность измерений и выбор функции подгонки. Поэтому для получения более точных результатов рекомендуется провести дополнительные исследования и сравнения на большем объеме данных.

Список используемой литературы

1. Бронштейн, И. Н. Справочник по математике для инженеров и учащихся втузов [Текст] / И. Н. Бронштейн, К.А. Семендяев. – М.: Наука. Главная редакция физико-математической литературы, 1981. – 720с.
2. Бесстремьянная, Г. Е. Применение ядерных и параметрических регрессий для оценки влияния страховых медицинских организаций на качество региональных систем здравоохранения [Текст] / Г. Е. Бесстремьянная, 2015. - 18 с.
3. Математический энциклопедический словарь [Текст] / Гл. ред. Ю. В. Прохоров. - М.: Советская энциклопедия, 1988. - 847 с.
4. Хиценко, В. Е. Непараметрическая статистика в задачах защиты информации. Конспект лекций [Текст] / В. Е. Хиценко, 2012. - 196 с.

переделать генерацию файлов

сделать так чтобы в непараметрике коэффициент C зависил от ошибки и был минимальным

переделать все текста

переделать аппроксимацию

создать табличку

поменять графики

начать делать приложение на pyqt

Для непараметрической оценки регрессии также вводится критерий ошибки E , который является среднеквадратичной ошибкой, которая представляет собой показатель, указывающий нам среднее расстояние между прогнозируемыми значениями из модели и фактическими значениями в наборе данных. Формула для нахождения среднеквадратичной ошибки, часто обозначаемая аббревиатурой RMSE, выглядит следующим образом:

$$E = \sqrt{\frac{\sum_{i=1}^n (y^*(x_{pi}) - y_{pi})^2}{n}}$$