**Государственное бюджетное общеобразовательное учреждение города Москвы «Школа № 1532»**

Сравнение непараметрических методов моделирования с параметрическим

or

Исследование параметрических методов моделирования с непараметрическими и исследование их эффективности

10 класс, ГБОУ Школа №1532,

Гришин Илья Андреевич

Руководитель: учитель информатики, ГБОУ Школа №1532,

Сергиенко Антон Борисович

Москва, 2024

**Оглавление**

[Введение 0](#_Toc33143531)

1. [Актуальность работы 0](#_Toc33143534)

2. [Цели и задачи проекта 0](#_Toc33143532)

3. [Методика выполнения работы 0](#_Toc33143533)

3.1. [Подготовка эксперементальных данных 0](#_Toc33143533)

3.2. [Парамметрические методы 0](#_Toc33143533)

3.3. [Непараметрические методы 0](#_Toc33143533)

3.4. [Сравнение и итоги 0](#_Toc33143533)

4. [Результаты 0](#_Toc33143534)

5. [Выводы 0](#_Toc33143535)

6. [Список используемой литературы 0](#_Toc33143536)

7. [Приложения 0](#_Toc33143533)

**Введение**

В работе рассмотрены методы статистического моделирования, на различных выборках. Для поставленных задач применяются регрессионная модель, основанная на непараметрической оценке Надарая-Ватсона [4], 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, существует 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 наличие 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 двумерного, так и для трёхмерного моделирования и запишем их в отдельный файл. Мы начнем с генерации некоторых случайных точек двумерных данных. Каждый пример будем записывать в отдельный .txt файл с соответствующим названием.

Шум будем рассчитывать по следующей формуле:

,

где P – процент шума, a – новое значение с шумом для y i-го.

Реализуем программу:

Листинг 1 – Скрипт для генерации данных в трёхмерном пространстве.

*# импортируем необходимые модули***import** numpy **as** np  
**import** random  
  
*# сгенерируем 100 случайных точек где Y - синус X*X = np.linspace(-10, 10, 100)  
Y = np.sin(X)  
  
*# пройдёмся циклом по коэффициентам процентов***for** proc **in** range(0, 160, 10):  
 *# процент шума = proc  
 # добавим шум в данные по формуле* Y2 = [Yi + random.uniform(-(((max(Y) - min(Y)) \* proc / 100) / 2),  
 (((max(Y) - min(Y)) \* proc / 100) / 2)) **for** Yi **in** Y]  
 data = np.array([X, Y2]).T  
 *# добавим данные в файл, в название укажем процент шума* **with** open(**f'dataXY\_{**round(proc)**}.txt'**, **'w'**) **as** f:  
 [print(i, j, file=f) **for** i, j, **in** data]

Листинг 1 – Скрипт для генерации данных в 2D пространстве

Теперь аналогично сгенерируем точки в 3D пространстве:

*# импортируем необходимые модули***import** numpy **as** np  
**import** random  
**import** math  
  
*# генерация 100 случайных точек*X = [random.uniform(-10, 10) **for** \_ **in** range(100)]  
Y = [random.uniform(-10, 10) **for** \_ **in** range(100)]  
Z = [(math.sin(X[i]) \* math.sin(Y[i])) **for** i **in** range(100)]  
  
*# пройдёмся циклом по коэффициентам процентов***for** proc **in** range(0, 160, 10):  
 *# процент шума = proc  
 # добавим шум в данные* Z1 = [Zi + random.uniform(-(((max(Y) - min(Y)) \* proc / 100) / 2),  
 (((max(Y) - min(Y)) \* proc / 100) / 2)) **for** Zi **in** Z]  
 data = np.array([X, Y, Z1]).T  
 *# добавим данные в файл, в название укажем процент шума* **with** open(**f'dataXYZ\_{**round(proc)**}.txt'**, **'w'**) **as** f:  
 [print(i, j, z, file=f) **for** i, j, z **in** data]

Листинг 2 – Скрипт для генерации данных в трёхмерном пространстве

У нас получилось множество файлов с различным уровнем шума, для дальнейшего исследования. Теперь реализуем функции для удобного получения данных из файла.

**def** get\_data2D(name\_file):  
 **with** open(name\_file) **as** f:  
 a = f.readlines()  
 x, y = [], []  
 **for** i **in** a:  
 a, b = i.split()  
 x.append(a)  
 y.append(b)  
 **return** x, y  
*# get\_data2D(name\_file)[0] - X  
# get\_data2D(name\_file)[1] – Y*

**def** get\_data3D(name\_file):  
 **with** open(name\_file) **as** f:  
 a = f.readlines()  
 x, y, z = [], [], []  
 **for** i **in** a:  
 print(i)  
 a, b, c = i.split()  
 x.append(float(a))  
 y.append(float(b))  
 z.append(float(c))  
  
 **return** x, y, z  
*# get\_data3D(name\_file)[0] - X  
# get\_data3D(name\_file)[1] - Y  
# get\_data3D(name\_file)[2] - Z*

Листинг 3 – Программа для получения данных

Пример того, что мы получим вызван функцию ”get\_data3D” с уровнем шума – 30%

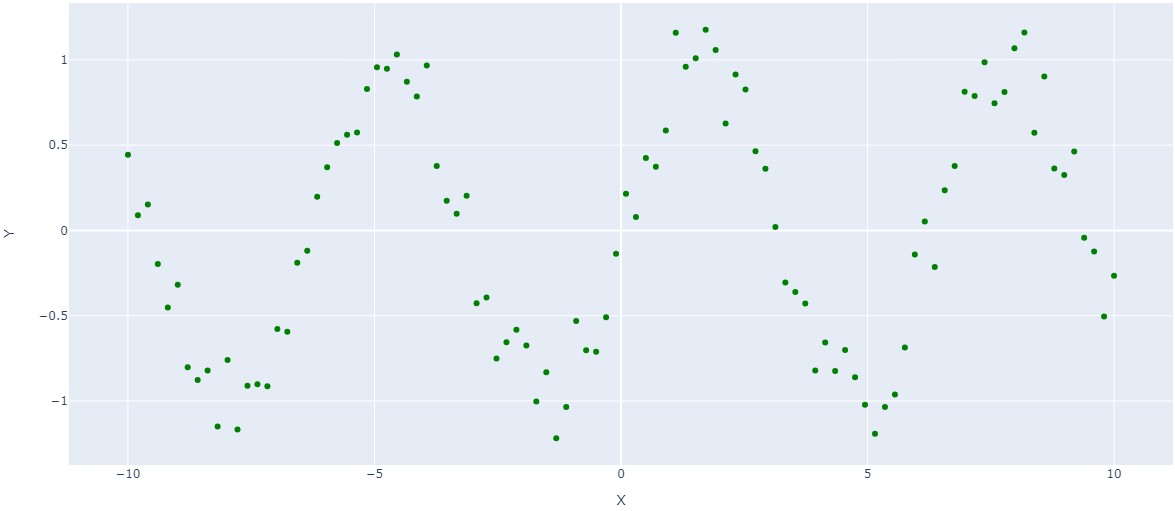


Рисунок 1 – пример выборки с уровнем шума – 30%

Здесь реализовано две функции для получения двумерных и трёхмерных 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 двумерного пространства*:*

*# импортируем необходимые модули***import** numpy **as** np  
**from** scipy.optimize **import** curve\_fit  
**import** plotly.graph\_objs **as** go  
**from** RMSE **import** rmse  
  
  
*# создадим функцию для аппроксимации на вход  
# которой подаются два массива x и y***def** approx\_2D(x, y):  
 *# Определим математическую функцию, которая  
 # будет использоваться для подгонки кривой.  
 # В этом примере мы будем использовать  
 # синусоидальную функцию.* **def** func(t, A, w, p, c):  
 **return** A \* np.sin(w \* t + p) + c  
  
 x = np.array(list(map(float, x)))  
 y = np.array(list(map(float, y)))  
 popt, \_ = curve\_fit(func, x, y)  
 x = np.linspace(x.min(), x.max(), 100)  
 fig = go.Figure()  
 fig.add\_trace(go.Scatter(x=x, y=y, mode=**'markers'**, name=**'$$Исходные данные$$'**,  
 marker=dict(color=**'green'**)))  
 fig.add\_trace(go.Scatter(x=x, y=func(x, \*popt), name=**'$$Синусоидальная функция$$'**,  
 marker=dict(color=**'blue'**)))  
 fig.add\_trace(go.Scatter(x=x, y=func(x, \*popt), mode=**'markers'**, name=**'$$Точки аппроксимации$$'**,  
 marker=dict(color=**'red'**, symbol=**"star-triangle-up"**)))  
 fig.update\_layout(xaxis\_title=**"X"**, yaxis\_title=**"Y"**)  
 fig.show()

Листинг 4 – Функция для аппроксимации в двумерном 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, генерирующая тестовую выборку заданная нами, представляет собой синусоидальную функцию.

Теперь приступим к реализации аппроксимации для трёхмерного пространства:

Создадим, аналогичную прошлой, функцию с некоторыми нюансами

*# импортируем необходимые модули***import** numpy **as** np  
**from** scipy.optimize **import** curve\_fit  
**import** plotly.graph\_objs **as** go  
  
  
*# создадим функцию для аппроксимации на вход которой  
# подаются три массива X и Y и Z***def** approx\_3D(x, y, z):  
 x = np.array(list(map(float, x)))  
 y = np.array(list(map(float, y)))  
 z = np.array(list(map(**lambda** x: x/10, z)))  
 *# Определение математической функции для  
 # аппроксимации кривой* **def** func(xy, a):  
 x, y = xy  
 **return** a \* np.sin(x)  
 *# Выполнить подгонку кривой* popt, pcov = curve\_fit(func, (x, y), z)  
 *# Функция для реализации 3D-графика точек данных  
 # и подобранной кривой* fig = go.Figure()  
  
 fig.add\_trace(go.Scatter3d(x=x, y=y, z=z, mode=**'markers'**, marker=dict(color=**'blue'**, size=2)))  
 x\_range = np.linspace(-10, 10, 100)  
 y\_range = np.linspace(-10, 10, 100)  
 X, Y = np.meshgrid(x\_range, y\_range)  
 Z = func((X, Y), \*popt)  
 fig.add\_trace(go.Surface(x=X, y=Y, z=Z, colorscale=**'Reds'**, opacity=0.5))  
 fig.update\_layout(scene=dict(  
 xaxis\_title=**'X'**,  
 yaxis\_title=**'Y'**,  
 zaxis\_title=**'Z'** ))  
 fig.show()

Листинг 5 – Функция для аппроксимации в трёхмерном 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

,

где в данном случае, — коэффициент размытия, а — колоколообразная ядерная функция, равная следующему значению:

Напишем функцию по формуле Надарая-Ватсона:

*# колоколообразная ядерная функция***def** phi(p):  
 **if** p \*\* 2 <= 5:  
 **return** 0.335 - 0.067 \* p \*\* 2  
 **else**:  
 **return** 0  
  
*# функция для вычисления значения по формуле Надарая-Ватсона***def** f\_nadaray\_watson(x\_dop, x\_pi\_list, y\_pi\_list, c):  
 numerator = 0  
 denominator = 0  
  
 **for** i **in** range(len(x\_pi\_list)): *# вычисление весов точек* phi\_value = phi((x\_dop - x\_pi\_list[i]) / c)  
 numerator += y\_pi\_list[i] \* phi\_value  
 denominator += phi\_value  
  
 **return** numerator / denominator *# оценка значения в заданной точке*

Листинг 6 – Функция для непараметрической регрессии в 2D

Реализуем непараметрический метод моделирования для 2D пространства:

!!! реализовать нахождение оптимального коэффициента С

*# импортируем необходимые модули***import** numpy **as** np  
**from** Nad\_Wat **import** f\_nadaray\_watson  
**import** plotly.graph\_objs **as** go  
**from** RMSE **import** rmse  
  
  
*# импортируем необходимые модули***import** numpy **as** np  
**from** Nad\_Wat **import** f\_nadaray\_watson  
**import** plotly.graph\_objs **as** go  
**from** RMSE **import** rmse  
  
  
*# создадим функцию, которая будет реализовывать непараметрическую регрессию,  
# на вход принимаются два массива X и Y***def** nep\_regression\_2D(x, y):  
 x = np.array(list(map(float, x)))  
 y = np.array(list(map(float, y)))  
  
 c = 0.5  
 *# Вычисляем оценки значений в заданных точках* query\_y = [f\_nadaray\_watson(q, x, y, c) **for** q **in** x]  
  
 *# График* fig = go.Figure()  
 fig.add\_trace(go.Scatter(x=x, y=y, mode=**'markers'**, name=**'$$Исходные данные$$'**,  
 marker=dict(color=**'green'**)))  
 fig.add\_trace(go.Scatter(x=x, y=query\_y, name=**'$$Оценка Надарая-Ватсона$$'**,  
 marker=dict(color=**'blue'**)))  
 fig.add\_trace(go.Scatter(x=x, y=query\_y, mode=**'markers'**, name=**'$$Точки оценки Надарая-Ватсона$$'**,  
 marker=dict(color=**'red'**, symbol=**"star-triangle-up"**)))  
 fig.update\_layout(xaxis\_title=**"X"**, yaxis\_title=**"Y"**)  
 fig.show()

Листинг 6 – Функция для непараметрической регрессии в двумерном пространстве.

Реализуем непараметрические методы моделирования для трёхмерного пространства:

*# импортируем необходимые модули***import** numpy **as** np  
**import** plotly.graph\_objs **as** go  
  
  
*# Функция для вычисления оценки Надарая-Ватсона***def** nadaraya\_watson(X, Z, x\_query, h):  
 weights = np.exp(-np.sum((X - x\_query) \*\* 2, axis=1) / (2 \* h \*\* 2))  
 weighted\_sum = np.sum(weights \* Z)  
 sum\_of\_weights = np.sum(weights)  
 **return** weighted\_sum / sum\_of\_weights  
  
  
*# функция для построения непараметрической регрессии***def** nep\_regression\_3D(x, y, z):  
 X = np.array([x, y, z]).T  
 *# Задание сетки для построения непараметрической регрессии по координате Z* grid\_size = 0.1  
 x\_range = np.arange(-10, 10, grid\_size)  
 y\_range = np.arange(-10, 10, grid\_size)  
 X\_grid, Y\_grid = np.meshgrid(x\_range, y\_range)  
 Z\_grid = np.zeros\_like(X\_grid)  
  
 *# Вычисление значений по координате с помощью оценки регрессии Надарая-Ватсон* **for** i **in** range(X\_grid.shape[0]):  
 **for** j **in** range(X\_grid.shape[1]):  
 x\_query = [X\_grid[i, j], Y\_grid[i, j], 0] *# Поиск координаты по Z* Z\_grid[i, j] = nadaraya\_watson(X, np.array(z), np.array(x\_query),  
 h=0.5) *# Вычисление значения на поверхности  
 # Построение графика в 3D* fig = go.Figure()  
 fig.add\_trace(go.Scatter3d(x=x, y=y, z=z, mode=**'markers'**, marker=dict(color=**'blue'**, size=2), name=**'Исходные данные'**))  
 fig.add\_trace(  
 go.Surface(x=X\_grid, y=Y\_grid, z=Z\_grid, colorscale=**'blackbody'**, opacity=0.3, name=**'Регрессия Надарая-Ватсон'**))  
 fig.update\_layout(scene=dict(xaxis\_title=**'X'**, yaxis\_title=**'Y'**, zaxis\_title=**'Z'**),  
 legend=dict(x=0, y=1, traceorder=**'normal'**))  
 fig.show()

Листинг 7 – Функция для непараметрического моделирования в трёхмерном пространстве.

В данном коде 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.

**from** approx\_2D **import** approx\_2D  
**from** get\_points **import** get\_data2D  
  
*# get\_data2D(name\_file)[0] - X  
# get\_data2D(name\_file)[1] - Y*proc = 120  
name\_file = **f'dataXY\_{**proc**}.txt'**x = get\_data2D(name\_file)[0]  
y = get\_data2D(name\_file)[1]  
approx\_2D(x, y)

Листинг 8 – Программа для запуска аппроксимации функции в двумерном пространстве

**from** nep\_2D **import** nep\_regression\_2D  
**from** get\_points **import** get\_data2D  
  
*# get\_data2D(name\_file)[0] - X  
# get\_data2D(name\_file)[1] - Y*proc = 0  
name\_file = **f'dataXY\_{**proc**}.txt'**x = get\_data2D(name\_file)[0]  
y = get\_data2D(name\_file)[1]  
nep\_regression\_2D(x, y)

Листинг 9 – Программа для запуска непараметрической регрессии функции в двумерном пространстве

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:

,

где - прогнозируемое значение для i -го наблюдения в наборе данных, - наблюдаемое значение для i -го наблюдения в наборе данных, n - размер выборки

Напишем функцию для вычисления RMSE в двумерном пространстве:

**import** math  
  
  
**def** rmse(targets, predictions):  
 squared\_errors = [(p - t) \*\* 2 **for** p, t **in** zip(predictions, targets)]  
 mean\_squared\_error = sum(squared\_errors) / len(predictions)  
 rmse = math.sqrt(mean\_squared\_error)  
  
 **return** round(rmse, 5)

Листинг 10 – Программа для запуска подсчёта 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 | Тут будут все значения ошибок |  |
| 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:

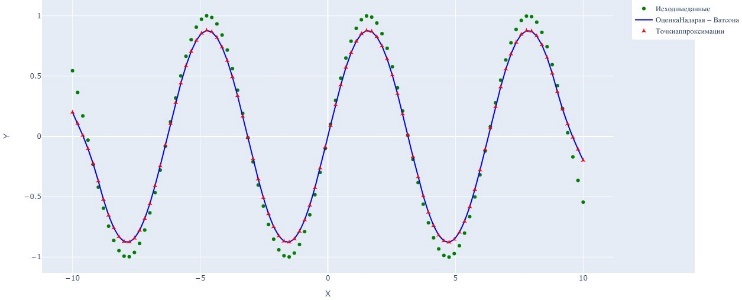
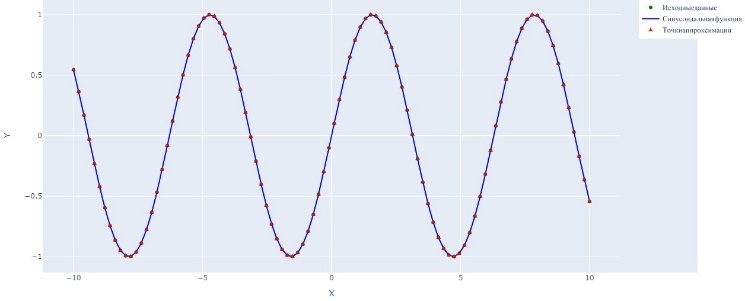


Рисунок 2 – Аппроксимация двумерной выборки с 0% шума

Рисунок 3 – Непараметрическая двумерная регрессия с 0% шума

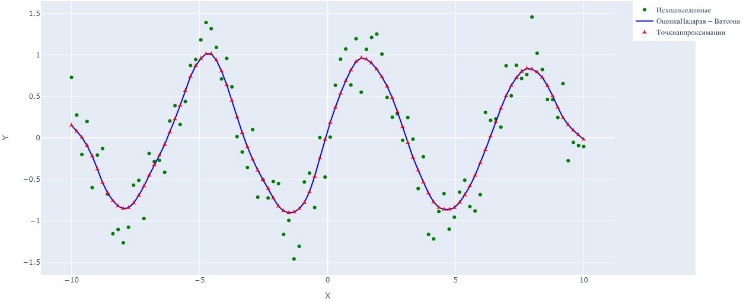
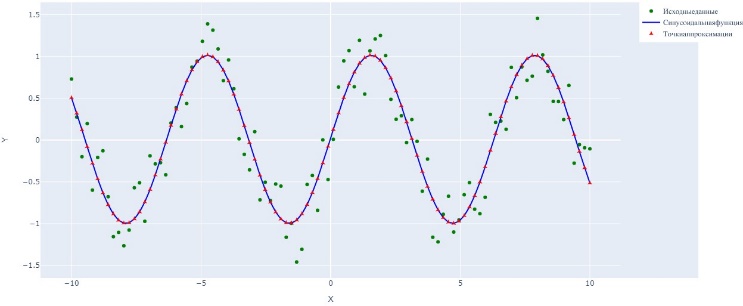


Рисунок 4 – Аппроксимация двумерной выборки с 50% шума

Рисунок 5 - Непараметрическая двумерная регрессия с 50% шума

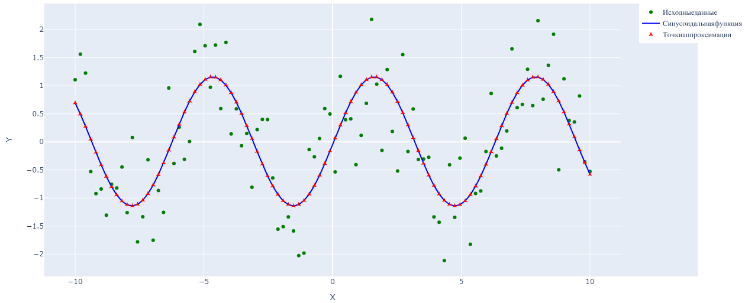
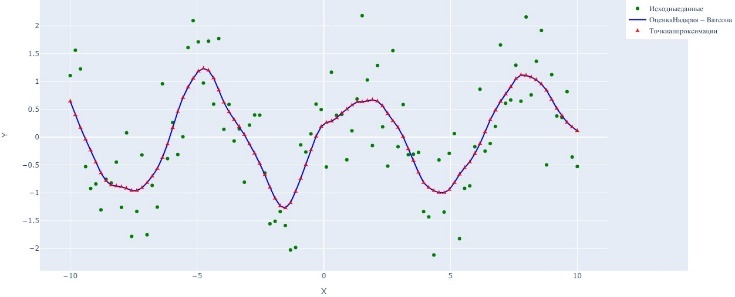
 

Рисунок 6 – Аппроксимация двумерной выборки с 120% шума

Рисунок 7 - Непараметрическая двумерная регрессия с 120% шума

Теперь рассмотрим 3D методы:

**from** approx\_3D **import** approx\_3D  
**from** get\_points **import** get\_data3D  
  
*# get\_data3D(name\_file)[0] - X  
# get\_data3D(name\_file)[1] - Y  
# get\_data3D(name\_file)[2] - Z*proc = 120  
name\_file = **f'dataXYZ\_{**proc**}.txt'**x = get\_data3D(name\_file)[0]  
y = get\_data3D(name\_file)[1]  
z = get\_data3D(name\_file)[2]  
approx\_3D(x, y, z)

Листинг 10 – Программа для запуска аппроксимации функции в 3D

**from** nep\_3D **import** nep\_regression\_3D  
**from** get\_points **import** get\_data3D  
  
*# get\_data3D(name\_file)[0] - X  
# get\_data3D(name\_file)[1] - Y  
# get\_data3D(name\_file)[2] - Z*proc = 120  
name\_file = **f'dataXYZ\_{**proc**}.txt'**x = get\_data3D(name\_file)[0]  
y = get\_data3D(name\_file)[1]  
z = get\_data3D(name\_file)[2]  
nep\_regression\_3D(x, y, z)

Листинг 11 – Программа для запуска непараметрической регрессии функции в трёхмерном пространстве.

Для более наглядного представления методов трёхмерной регрессии рассмотрим некоторые графики с 0, 50 и 120 процентами шума

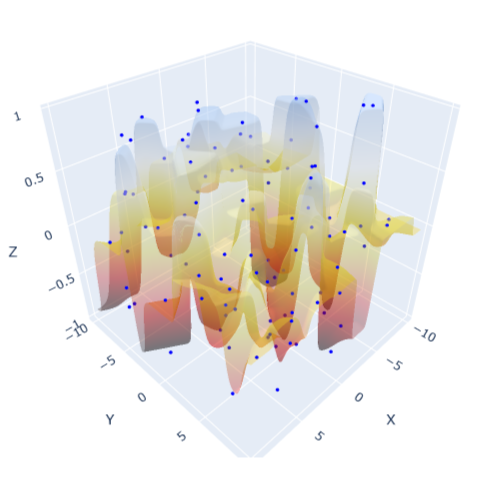
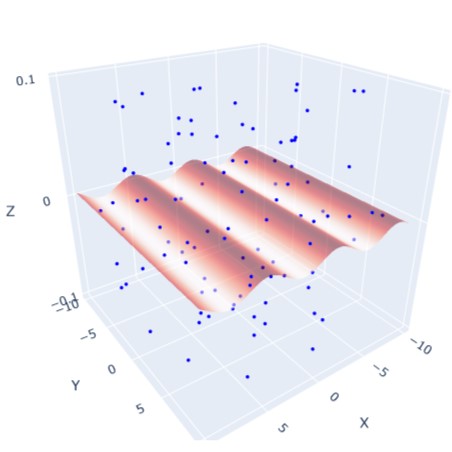


Рисунок 8 – Аппроксимация трёхмерной выборки с 50% шума

Рисунок 9 - Непараметрическая трёхмерная регрессия с 50% шума

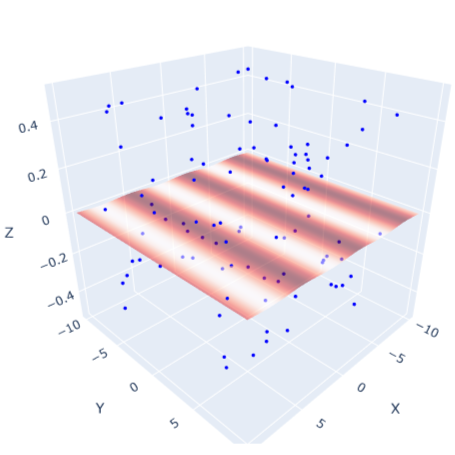
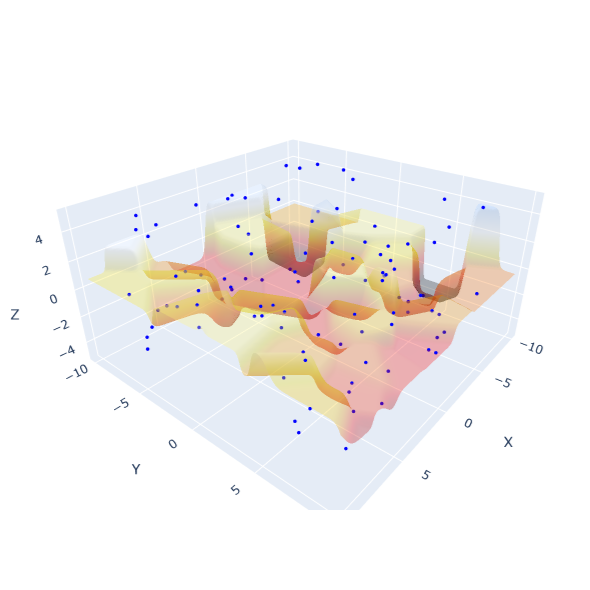
 

Рисунок 8 – Аппроксимация трёхмерной выборки с 50% шума

Рисунок 9 - Непараметрическая трёхмерная регрессия с 50% шума

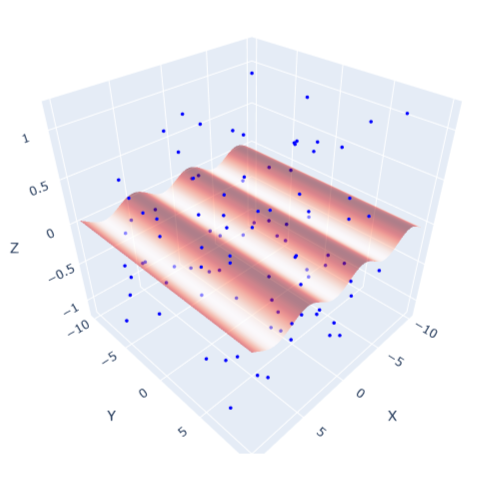
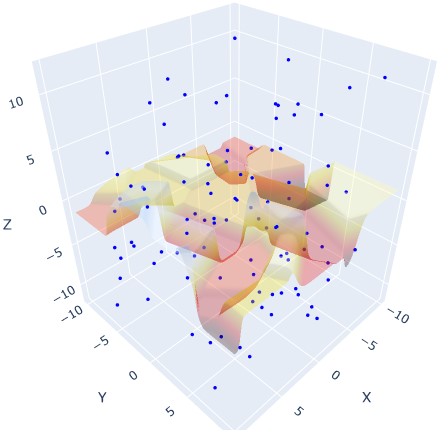
 

Рисунок 8 – Аппроксимация трёхмерной выборки с 120% шума

Рисунок 9 - Непараметрическая трёхмерная регрессия с 120% шума

# 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. Бесстремянная, Г. Е. Применение ядерных и параметрических регрессий для оценки влияния страховых медицинских организаций на качество региональных систем здравоохранения [Текст] / Г. Е. Бесстремянная, 2015. - 18 c.
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 c.