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 …, ГБОУ Школа №1532,
…

Москва, 2024

Оглавление

Введение

В работе рассмотрены методы статистического моделирования, на различных выборках с известным параметром и нет. Для решения данной задачи применяется регрессионная модель, основанная на непараметрической оценке Надарая-Ватсона, а также аппроксимация с подгонкой по функции.

Современные методы моделирования данных предлагают широкий спектр инструментов для анализа и прогнозирования различных явлений. Среди них особое внимание привлекают параметрические и непараметрические методы, каждый из которых имеет свои особенности и применимость в различных сценариях.

В данном проекте проводится сравнительный анализ эффективности непараметрических методов, таких как регрессия на основе оценки Надарая-Ватсона, с параметрическими методами, включая аппроксимацию с подгонкой по функции, на различных выборках данных. Путем сравнительного анализа мы стремимся обеспечить более глубокое понимание того, какие методы моделирования следует предпочитать в различных контекстах и при различных условиях данных.

Актуальность работы

Современная область анализа данных и статистики стремительно развивается, и в силу этого актуальность исследований по сравнению методов моделирования остается на высоком уровне. В контексте данного проекта, где основное внимание уделяется сопоставлению непараметрических и параметрических методов, существует несколько ключевых моментов, которые делают данное исследование весьма актуальным:

1. Гибкость и адаптивность методов: Сложные структуры данных могут поддаваться более успешному моделированию с использованием непараметрических методов, в то время как параметрические методы могут быть предпочтительны в случаях, когда структура данных более предсказуема. Исследование эффективности каждого из подходов становится важным шагом для выбора оптимального метода в зависимости от особенностей данных.

2. Практическая применимость: С популяризацией методов машинного обучения и статистического моделирования, важно понять, какие методы наиболее подходят для конкретных задач. Исследование различных методов на разнообразных выборках данных предоставляет ценную информацию для практикующих специалистов в области анализа данных.

3. Разнообразие областей применения: Методы моделирования широко используются в различных областях, таких как экономика, медицина, биология и социальные науки. Сравнение методов на разнообразных выборках позволяет обобщить результаты и делает исследование более универсальным в контексте различных дисциплин.

4. Оптимизация ресурсов: Эффективное использование ресурсов, таких как вычислительная мощность, время и данные, является критическим вопросом в современных исследованиях. Понимание, какие методы более эффективны для конкретных сценариев, может значительно сэкономить ресурсы и повысить эффективность аналитических процессов.

С учетом этих факторов, данное исследование о сравнении непараметрических и параметрических методов моделирования 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.
3. Реализовать непараметрические методы моделирования.
4. Рассмотреть каждый метод для определённой выборки.
5. Подвести итоги проделанной работы.

Методика выполнения исследования

Первый этап – Подготовка экспериментальных данных

Создадим различные имитации выборок как для 3D, так и для 2D моделирования и запишем их в отдельный файл.

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

Реализуем программу:

```python
# импортируем необходимые модули
import numpy as np
import random

# генерация 100 случайных точек
x = np.linspace(-10, 10, 100)
y1 = [random.uniform(-10, 10) for _ in range(100)]
data1 = np.array([x, y1]).T

# добавим параметр в данные
y2 = np.sin(x)
data2 = np.array([x, y2]).T

# добавим шум в данные
y3 = np.sin(x) + np.random.normal(0, 0.1, 100)
data3 = np.array([x, y3]).T

# запись данных в файл:

# без параметра
with open('random_dataXY.txt', 'w') as f:
    [print(i, j, file=f) for i, j, in data1]

# с параметром без шума
with open('dataXY.txt', 'w') as f:
    [print(i, j, file=f) for i, j, in data2]

# с параметром с шумом
with open('dataXY_with_hindrance.txt', 'w') as f:
    [print(i, j, file=f) for i, j, in data3]
```

Рисунок 1 – Скрипт для генерации данных в 2D пространстве

Теперь аналогично сгенерируем точки в 3D пространстве

```python
# импортируем необходимые модули
import numpy as np
import random

# генерация 100 случайных точек
x = np.linspace(-10, 10, 100)
y = [random.uniform(-10, 10) for _ in range(100)]
z1 = [random.uniform(-10, 10) for _ in range(100)]
data1 = np.array([x, y, z1]).T

# добавим параметр в данные
z2 = np.sin(x * y)
data2 = np.array([x, y, z2]).T

# добавим шум в данные
z3 = np.sin(x * y) + np.random.normal(0, 0.1, 100)
data3 = np.array([x, y, z3]).T

# запись данных в файл:

# без параметра
with open('random_dataXYZ.txt', 'w') as f:
    [print(i, j, z, file=f) for i, j, z in data1]

# с параметром без шума
with open('dataXYZ.txt', 'w') as f:
    [print(i, j, t, file=f) for i, j, t in data2]

# с параметром с шумом
with open('dataXYZ_with_hindrance.txt', 'w') as f:
    [print(i, j, z, file=f) for i, j, z in data3]
```

Рисунок 2 – Скрипт для генерации данных в 3D пространстве

Что у нас получилось:

Мы создали шесть файлов с примерами различных выборок, состоящих из 100 точек, сгенерированных по-разному.

Далее приведены все варианты с описанием

‘random_dataXY.txt’ - два независящих друг от друга массива 2D

‘dataXY.txt’ - два массива с параметром без шума

‘dataXY_with_hindrance.txt’ - два массива с параметром с шумом

‘random_dataXYZ.txt’ - три независящих друг от друга массива 3D

‘dataXYZ.txt’ - три массива с параметром без шума

‘dataXYZ_with_hindrance.txt’ - три массива с параметром с шумом

Теперь давайте сделаем программу для удобного получения данных из файла.

```
6 usages
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


8 usages
def get_data3D(name_file):
    with open(name_file) as f:
        a = f.readlines()
        x, y, z = [], [], []
        for i in a:
            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
```

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

Параметрическое моделирование — моделирование (проектирование) с использованием параметров элементов модели и соотношений между этими параметрами. Параметризация позволяет за короткое время «проиграть» (с помощью изменения параметров или геометрических соотношений) различные конструктивные схемы и избежать принципиальных ошибок.

https://ru.wikipedia.org/wiki/Параметрическое_моделирование

Нам необходимо создать функции, которые будет удобно применить к каждому из примеров на языке *Python*

Реализуем аппроксимацию для 2D пространства*:*

```python
# импортируем необходимые модули
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import curve_fit


# создадим функцию для аппроксимации на вход
# которой подаются два массива x и y
2 usages
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)
    plt.plot(x, func(x, *popt), color='green',
             label="Синусоидальная функция")
    plt.legend(loc='best')
    plt.scatter(x, y, color='red', s=15)
    plt.xlabel('x')
    plt.ylabel('y')
    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 реализуем аппроксимацию для 3D пространства*:*

Создадим, аналогичную прошлой, функцию с некоторыми нюансами

```python
# импортируем необходимые модули
import numpy as np
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt

# создадим функцию для аппроксимации на вход которой
# подаются три массива X и Y и Z
2 usages
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(float, 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 = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    ax.scatter(x, y, z, color='blue')
    x_range = np.linspace(-10, 10, 50)
    y_range = np.linspace(-10, 10, 50)
    X, Y = np.meshgrid(x_range, y_range)
    Z = func((X, Y), *popt)
    ax.plot_surface(X, Y, Z, color='red', alpha=0.5)
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('Z')
    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

Углубимся в непараметрические методы

Непараметрические методы – это количественные методы статистической обработки данных, применение которых не требует знания закона распределения изучаемых признаков в совокупности и вычисления их основных параметров. - *ПРОВЕРКА СТАТИСТИЧЕСКИХ ГИПОТЕЗ Смирнова З.М., Крейнина М.В.*

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

Формула для непараметрической оценки регрессии Надарая-Ватсона

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

Реализуем непараметрический метод моделирования для 2D пространства:

```python
# импортируем необходимые модули
import numpy as np
import matplotlib.pyplot as plt


# создадим функцию, которая будет оценивать значения в определённой точке
1 usage
def f_nadaray_watson(x, y, query_x, h):
    weights = np.exp(-0.5 * ((x - query_x) / h) ** 2)  # вычисление весов точек
    numerator = np.sum(weights * y)
    denominator = np.sum(weights)
    return numerator / denominator  # оценка значения в заданной точке


# создадим функцию, которая будет реализовывать непараметрическую регрессию,
# на вход принимаются два массива X и Y
2 usages
def nep_regression_2D(x, y):
    x = np.array(list(map(float, x)))
    y = np.array(list(map(float, y)))
    # Задаем точки, в которых хотим получить оценку
    query_x = np.array([i for i in x])

    # Вычисляем оценки значений в заданных точках
    query_y = [f_nadaray_watson(x, y, q, 0.5) for q in query_x]

    # График
    plt.scatter(x, y, label='Исходные данные')
    plt.plot(query_x, query_y, label='Оценка Надарая-Ватсона', color='blue')
    plt.legend()
    plt.show()
```

Рисунок 6 – Функция для непараметрической регрессии в 2D

Реализуем непараметрические методы моделирования для 3D пространства:

```python
# импортируем необходимые модули
import numpy as np
import matplotlib.pyplot as plt


# Функция для вычисления оценки Надарая-Ватсона
1 usage
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
2 usages
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, z, x_query, h=0.1)  # Вычисление
                                                    # значения на поверхности

    # Построение графика в 3D
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    ax.scatter(x, y, z, c='blue', label='Исходные данные')
    ax.plot_surface(X_grid, Y_grid, Z_grid, cmap='gnuplot2_r', alpha=0.5,
                    label='Регрессия Надарая-Ватсон')
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('Z')
    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.

```python
from approx_2D import approx_2D
from get_points import get_data2D

# get_data2D(name_file)[0] - X
# get_data2D(name_file)[1] - Y

name_file = 'random_dataXY.txt'
# name_file = 'dataXY.txt'
# name_file = 'dataXY_with_hindrance.txt'
x = get_data2D(name_file)[0]
y = get_data2D(name_file)[1]
approx_2D(x, y)
```

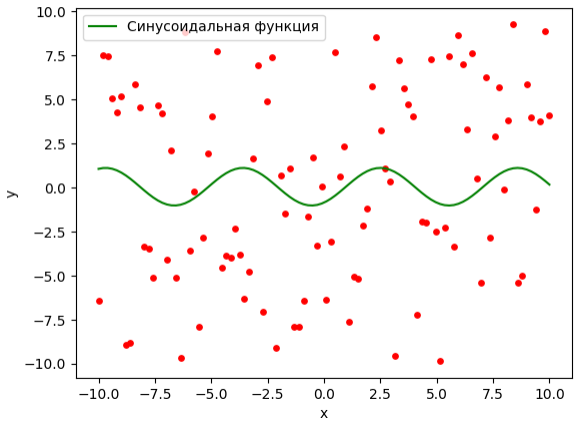
Рисунок 8 – Программа для запуска аппроксимации функции в 2D

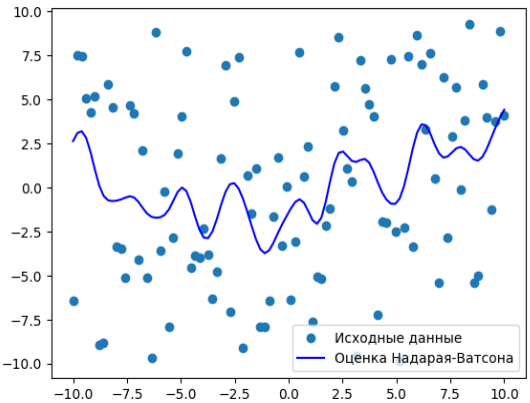
```python
from nep_2D import nep_regression_2D
from get_points import get_data2D

# get_data2D(name_file)[0] - X
# get_data2D(name_file)[1] - Y

name_file = 'random_dataXY.txt'
# name_file = 'dataXY.txt'
# name_file = 'dataXY_with_hindrance.txt'
x = get_data2D(name_file)[0]
y = get_data2D(name_file)[1]
nep_regression_2D(x, y)
```

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

Рассмотрим результаты:

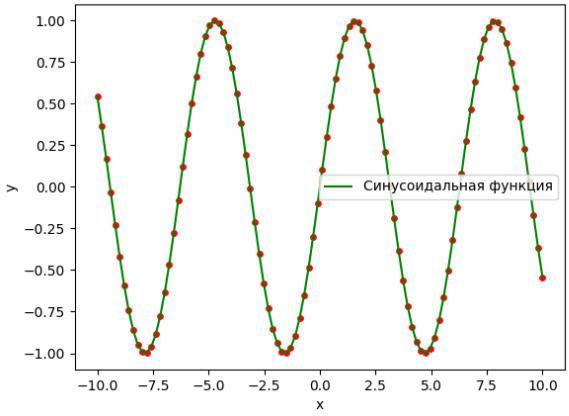
Для файла 'random_dataXY.txt':

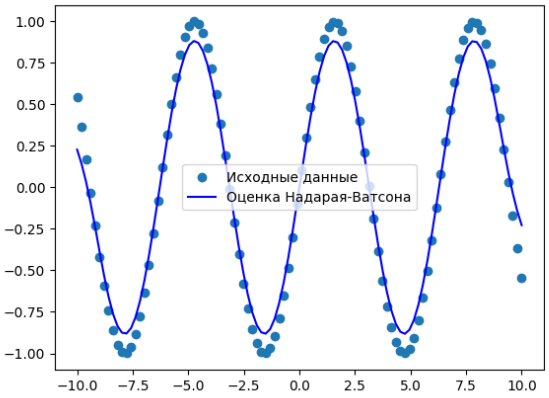




……

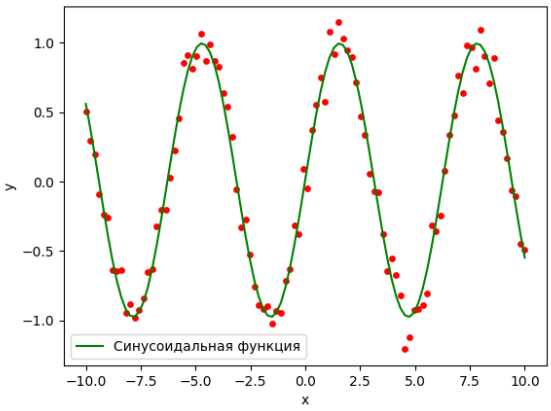
Для файла 'dataXY.txt':

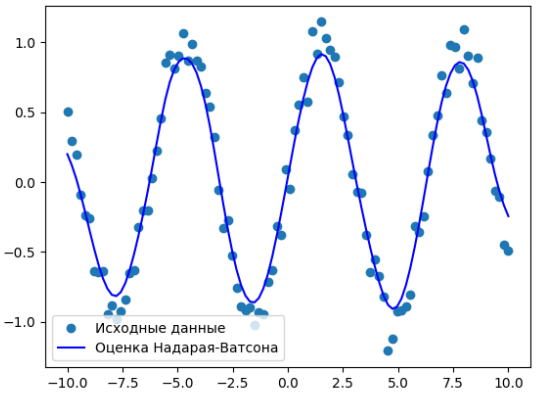




.....

Для файла 'dataXY_with_hindrance.txt':





.....

Теперь рассмотрим 3D методы:

```python
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

name_file = 'random_dataXYZ.txt'
# name_file = 'dataXYZ.txt'
# name_file = 'dataXYZ_with_hindrance.txt'
x = get_data3D(name_file)[0]
y = get_data3D(name_file)[1]
z = get_data3D(name_file)[2]
approx_3D(x, y, z)
```

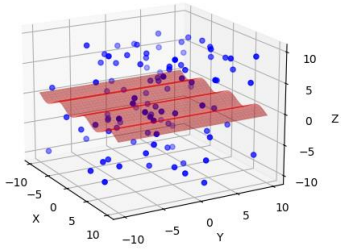
Рисунок 10 – Программа для запуска аппроксимации функции в 3D

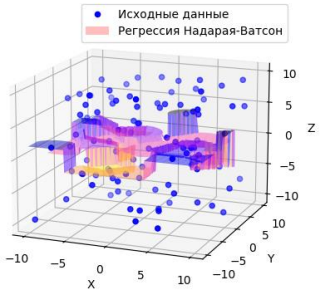
```python
from nep_3D import nep_regression_3D
from get_points import get_data3D

# get_data3D(name_file)[0] - X
# get_data3D(name_file)[1] - Y
# get_data3D(name_file)[2] - Z

name_file = 'random_dataXYZ.txt'
# name_file = 'dataXYZ.txt'
# name_file = 'dataXYZ_with_hindrance.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)
```

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

Рассмотрим результаты:

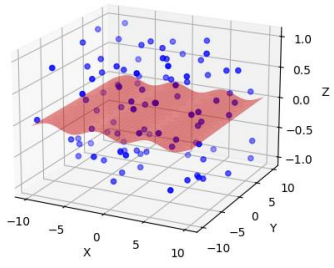
Для файла 'random_dataXYZ.txt':

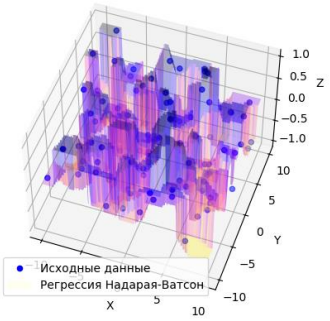




.....

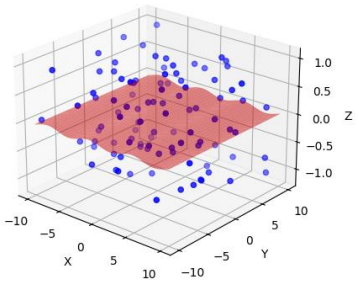
Для файла 'dataXYZ.txt':

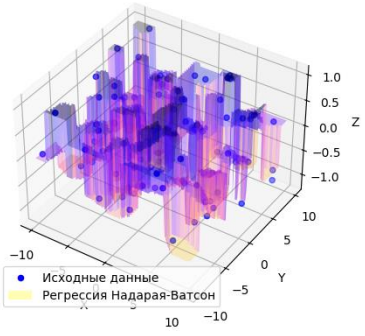




.....

Для файла 'dataXYZ_with_hindrance.txt':





.....

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 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.