Московский авиационный институт

(Национальный исследовательский университет)

Курсовая работа

по теме «Регрессионный анализ»

Студент: Цыкин И.А.

Группа: М8О-401Б-19

Вариант: 12

Москва 2022

Оглавление

0. Задание

1. Теоретическая часть Разрывный дизайн

https://colorfulsmeltead.notion.site/RegressionDiscontinuityDesigna6b019a0c0ca42769961ee2849cc110b

2. Практическая часть

Работа выполняется с помощью R, Python, Matlab, C++, Java. Можно использовать готовые функции из библиотек или написать свои.

Вариант:

Вариант	Функция $f(h)$	Носитель для h	Дисперсия шума $\sigma 2$
12	$sin(3x) + 1.5cos(4x) + 2x$	$-2 < h < 2$	2

2.1 Модельная часть

Смоделировать данные самостоятельно в соответствии с вариантом

$$X_k = f(h_k) + e_k, k = 1, \ldots, 60$$

где e_k — независимый случайные величины с распределением $N(0, \sigma 2)$. Точки внутри носителя для h выбирать равномерно. Смоделировать тестовую выборку объема 40, половина значений правее наблюдаемых значений, половина левее.

2.2. Метод наименьших квадратов

Для регрессии вида

$$Y_k = \theta_0 + \theta_1 h_k, k = 1..60$$

- Найти МНК-оценки неизвестных параметров.

- Построить график, на котором отобразить наблюдения, исходную функцию и линию регрессию.

- Вычислить коэффициент детерминации и найти оценку ковариационной матрицы МНК-оценки.

- Найти значения информационных критериев

- С помощью критерия Фишера проверить гипотезу $\theta_0 = 0$, $\theta_1 = 0$

- Построить доверительный интервал надежности 0.95 и 0.8 для полезного сигнала при h $X = \theta_0 + \theta_1 h$ из исходного носителя ±50%.

- Построить оценку метода наименьших модулей, отобразить ее на графике

- Оценить качество построенных регрессий на тестовой выборке

Для остатков $e_k = X_k - \dot{X_k}$

- Построить гистограмму, ядерную оценку плотности распределения

- По остаткам проверить гипотезу, что e имеет гауссовское распределение с помощью одного из критериев

● критерий Шапиро-Уилка (Shapiro–Wilk) [1];

● критерий D'Agostino K 2 [1];

● критерий ХаркеБера (Jarque–Bera) [1].

- Проверить наличие автокорреляции с помощью критерия Дарбина-Уотсона.

- Проверить наличие гетероскедастичности с помощью одного из критериев.

- Выводы.

2.3. Полиномиальная регрессия

Построить регрессию с помощью МНК

$$Y_k = \theta_0 + \theta_1 h_k + \theta_2 {h_k}^2 + .. + \theta_p {h_k}^p.$$

Порядок полинома p подбирать несколькими способами:

- по значению среднеквадратической погрешности МНК-оценки (на обучающей и/или тествой)

- по значению статистики критерия Фишера для гипотезы $\theta_p = 0$

- по MSE на тестовой выборке

- ваш способ

Выбираем единственное значение p.

Провести анализ остатков по схеме из пункта 2.2.

Построить график, на котором отобразить наблюдения, исходную функцию и линию регрессию.

Проверить для подобранной модели является ли матрица $H^T H$ мультиколлиниарной, если да, то построить оценку параметров с помощью метода редукции (ридж-оценка).

Выводы

2.4. Регрессия для наблюдений с выбросами

Смоделировать ошибки для модели регрессии (1) с помощью распределения Тьюки, приняв долю выбросов $\delta = 0.08$, номинальную дисперсию $\sigma_0^2 = \sigma^2$, дисперсию аномальных наблюдений $\sigma_1^2 = 100\sigma^2$. Построить МНК-оценку неизвестных параметров для модели (1) и оценить ее качество.

Построить МНК-оценку неизвестных параметров для модели (1) и оценить ее качество.

Провести анализ остатков по схеме из пункта 2.2.

Построить график, на котором отобразить наблюдения, исходную функцию и линию регрессию.

Провести отбраковку выбросов и пересчитать МНК-оценку, и оценить качество оценки.

Построить график, на котором отобразить наблюдения, исходную функцию и линию регрессию.

Провести анализ остатков по схеме из пункта 2.2.

Построить оценку метода наименьших модулей.

Построить график, на котором отобразить наблюдения, исходную функцию и линию регрессию.

Провести анализ остатков по схеме из пункта 2.2.

* Построить робастную оценку Хубера *2] (дополнительное задание)

Выводы

2.5. Квантильная регрессия

Смоделировать несимметричные ошибки для исходных данных, заменив у 90% отрицательных ошибок знак с минуса на плюс.

Построить МНК и МНМ оценки для получившихся наблюдений и регрессии (1).

Построить несколько квантильных регрессий (для разных значений параметра $\boldsymbol{\alpha}$) и оценить их качество.

Построить график, на котором отобразить наблюдения, исходную

функцию и линии регрессии.

Выводы

1. Теоретическая часть

1.1 Введение

Разрывный дизайн впервые был предложен в Thistlethwaite & Campbell (1960) как альтернатива рандомизированным экспериментам для оценивания эффекта воздействия программ. В своей статье авторы, анализируя влияние наличия у части студентов именных стипендий на результаты успеваемости, использовали тот факт, что присуждение этих наград было основано на наблюдаемых значениях специального экзаменационного теста. Идея состояла в том, что исследователь может использовать информацию об установленном пороге (разрыве) в значениях теста (выше или ниже минимальной оценки теста), чтобы выявить эффект влияния именной стипендии на уровень успеваемости среди лиц, которые получили оценки маржинально выше или ниже пороговой оценки. При определенных условиях сравнение среднего эффекта воздействия для объектов, находящихся в непосредственной близости от порогового значения, может помочь в выявлении причинно-следственных связей, имеющих социально-экономическое содержание, важное для принятия адекватных управленческих решений.

1.2 Постановка

Пусть у нас есть N объектов, где каждый объект обозначен индексом $i,\ i = 1..N$, определено, что каждый из них подвержен одному из двух типов воздействия: $W_i = 0$, если i объект подвергнут одному воздействию, и $W_i = 1,$ если i объект подвергнут альтернативному воздействию. Предполагается, что для каждого i объекта определена пара потенциальных исходов, $Y_i(0)$ и $Y_i(1)$, и эффект воздействия определен как $Y_i(1) - Y_i(0)$. Основной проблемой, с которой сталкивается исследователь, является то, что невозможно наблюдать одновременно оба исхода $Y_i(0)$ и $Y_i(1)$. Таким образом, для каждого объекта мы наблюдаем исход Y_i, выраженный как:

$$Y_i = (1 - W_i)Y_i(0) + W_iY_i(1) = \begin{cases} Y_1(0), \text{при } W_i = 0 \\ Y_1(1), \text{при } W_i = 1 \end{cases}, \text{где } W_i \in \{0, 1\}.$$

Основная идея разрывного дизайна состоит в том, что воздействие определяется полностью или частично значением переменной X_i. Эта переменная может коррелировать с потенциальными исходами, но предполагается, что эта корреляция распределена равномерно с каждой стороны разрыва и, таким образом, разрыв в условном распределении исходов, индексированных значением переменной X_i, можно интерпретировать как причинно-следственный эффект в непосредственной близости от точки разрыва.

Выделяют два типа разрывного дизайна: четкий (sharp) и нечеткий (fuzzy).

1.3 Четкий (sharp) дизайн

При четком дизайне распределение W_i является детерминированной функцией переменной X_i:

$$W_i = I\{X_i\},$$

где c – пороговое значение, I – индикаторная функция, при значениях $X_i \geq c$ объекты приписываются к одной группе воздействия, а при значениях $X_i \leq c$ наблюдения приписываются к другой. В случае четкого дизайна при расчете воздействий разрыв рассматривается как условное математическое ожидание исхода воздействия таким образом, что:

$$\lim_{x \to c+} M[Y_i|X_i = x] - \lim_{x \to c-} M[Y_i|X_i = x] = \lim_{x \to c+} M[Y_i(1)|X_i = x] - \lim_{x \to c-} M[Y_i(0)|X_i = x]$$

Таким образом, это можно интерпретировать как причинно-следственный эффект в непосредственной близости от точки разрыва:

$$\tau_{SRD} = M[Y_i(1) - Y_i(0)|X_i = c]$$

Для того чтобы данный эффект являлся причинно-следственным, используется допущение о сглаживаемости.

Если $M[Y_i(1)|X_i = x]$ и $M[Y_i(0)|X_i = x]$ являются непрерывными по x, то:

$$\tau_{SRD} = \lim_{x \to c+} M[Y_i|X_i = x] - \lim_{x \to c-} M[Y_i|X_i = x]$$

Непременным условием для применения разрывного дизайна является то, что все остальные переменные, определяющие Y, должны быть гладкими функциями по X. Если одна или несколько переменных, определяющих Y, резко изменяются в точке c, то параметр τ будет смещенной оценкой воздействия.

1.4 Нечеткий (fuzzy) дизайн

В нечетком дизайне вероятность воздействия не меняется с нуля на единицу в точке разрыва. Вместо этого при нечетком дизайне вероятность изменения воздействия в точке разрыва всегда меньше единицы:

$$\tau_{FRD} = \frac{\lim_{x \to c+} M[Y_i|X_i = x] - \lim_{x \to c-} M[Y_i|X_i = x]}{\lim_{x \to c+} M[W_i|X_i = x] - \lim_{x \to c-} M[W_i|X_i = x]}$$

Пусть $Wi(x)$ – невозрастающая функция в точке x при условии, что x = c.
Мы можем определить статус соответствия, если наблюдение имеет статус соответствия, если:

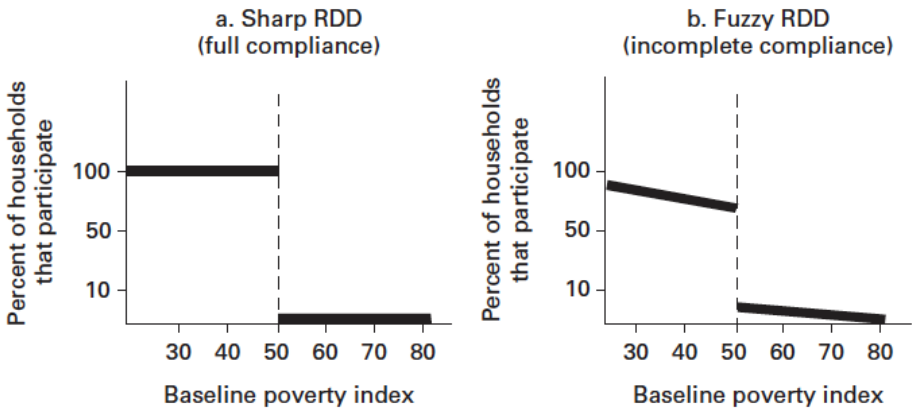
$$\lim_{x \to X_i-} W_i(x) = 0 \quad \lim_{x \to X_i+} W_i(x) = 1$$

Тогда формула позволяет оценить средний эффект участия для объектов с $X_i = c$ и только для наблюдений со статусом соответствия:

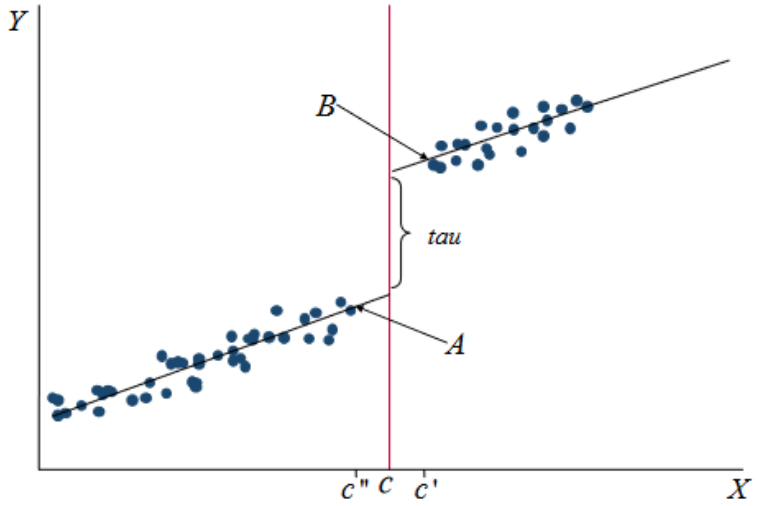
$$\tau_{FRD} = M[Y_i(1) - Y_i(0)|X_i = x \text{ и наблюдение i имеет статус соответствия}]$$

1.5 Применение

Для начала рассмотрим графическую интерпретацию влияния четного и нечетного взаимодействия.



Рассмотрим графический пример разрывного дизайна.



В случае использования ограниченной выборки необходимо полагаться на аппроксимацию, использовать наблюдения на достаточно широком отрезке, таком как c′ и c″, для того чтобы оценить эффект воздействия в точке c. В действительности, если взаимосвязь между Y и X линейная, то оценка параметра τ с помощью простой линейной регрессии в форме:

$$Y = \alpha + \tau W + \beta X + \varepsilon$$

Таким образом, мы получим наилучшею несмещенную оценку.

1.6 Пример

Я реализовал выборку X размером 100, каждый x_i из выборки зависит от t_i, где $t \in T$, $T = 0..10, i = 1..100$. Ошибки имеют нормальное распределение $e \sim N(0,4)$. На участке $a = (0..5)$ функция имеет вид $X_i = 2 * t_i + e_i$, а на остальном участке $X_i = 30 - 1.5 * t_i + e_i$.

Сначала найдем МНК-оценку для модели $X = \theta_0 + \theta_1 t$

```
print('МНК-оценка:', tetha[0], tetha[1])
sigma_e2 = df['e2'].sum()/(100-2)
print(f'Несмещенная оценка квадратов ошибки для МНК - {sigma_e2}')
```

```
МНК-оценка: 2.048148461330541 1.9891456046243023
Несмещенная оценка квадратов ошибки для МНК - 20.767403401973635
```

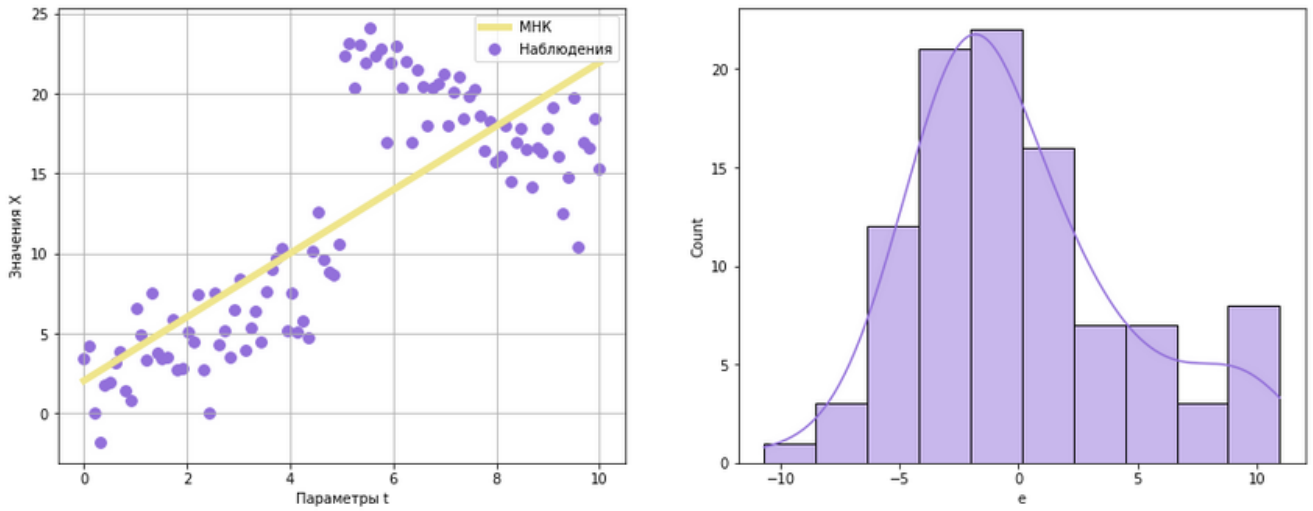


График 1-1. Демонстрация МНК-оценки для модели $X = \theta_0 + \theta_1 t$ и гистограмма ошибок

По графикам и несмещенной с.к-погрешности видно, что этот метод плохо подходит, это и требовалось ожидать. Можно попробовать другую модель, которая лучше будет подходить, но сейчас опробуем использовать разрывный дизайн.

```
rdd_df = df.assign(threshold=(df["t_"] > 5).astype(int))
model = smf.wls("x~t_*threshold", rdd_df).fit()
model.summary().tables[1]
```

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.5294	0.568	2.691	0.008	0.401	2.658
t_	1.5176	0.198	7.668	0.000	1.125	1.910
threshold	28.8099	1.620	17.785	0.000	25.594	32.025
t_:threshold	-3.0639	0.280	-10.947	0.000	-3.619	-2.508

```
predictions = model.fittedvalues
```

```
df['e_'] = df.x - predictions
df['e2_'] = df.e_**2
```

```
sigma_e2 = df['e2_'].sum()/(100-2)
print(f'Несмещенная оценка квадратов ошибки для МНК - {sigma_e2}')
```

```
Несмещенная оценка квадратов ошибки для МНК - 4.075899397054674
```

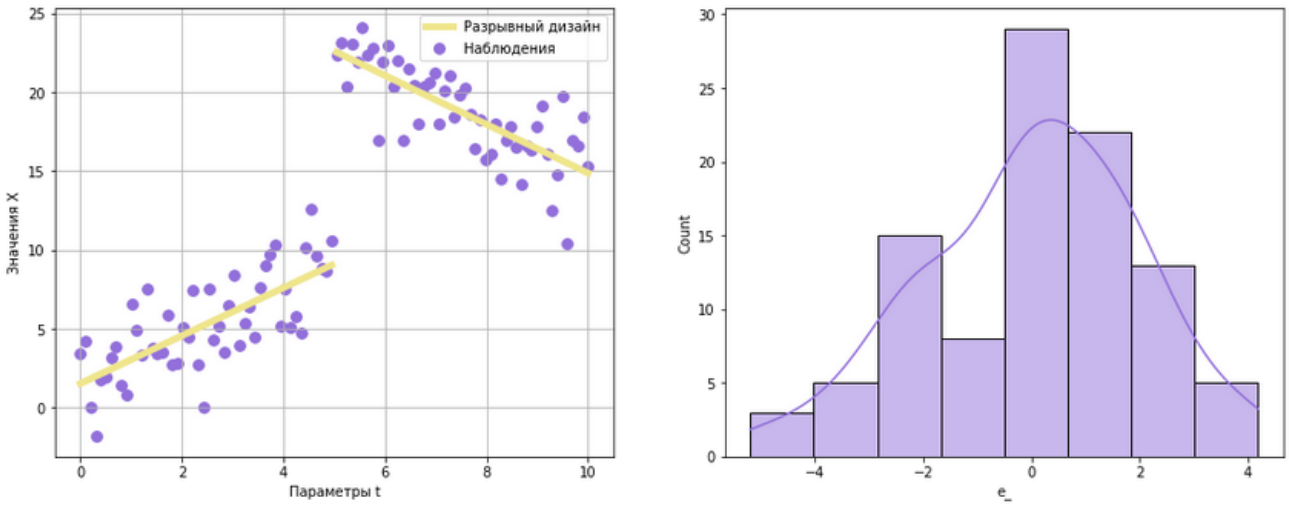


График 1-2. Демонстрация RDD МНК-оценки для модели $X = \theta_0 + \theta_1 t$ и гистограмма ошибок

По результатам видно, что этот методом мы оценили наблюдения лучше.

2. Практическая часть

2.0 Библиотеки

```
import numpy as np
import math
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from scipy.stats import f
from sklearn.linear_model import QuantileRegressor
from scipy.stats import normaltest
from scipy.stats import bartlett
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
import warnings
import random
warnings.filterwarnings('ignore')
```

2.1 Модельная часть

$$f(h) = sin(3x) + 1.5cos(4x) + 2x$$
$$-2 < h < 2$$
$$\sigma^2 = 2$$

```
k = 100
mu, sigma = 0, math.sqrt(2)
t = np.linspace(-2, 2, k)
df = pd.DataFrame(t[20:80], columns=['t'])
df_test = pd.DataFrame(np.append(t[0:20], t[80:100]), columns=['t'])
```

```
def f_(x):
    return math.sin(3*x) + 1.5*math.cos(4*x) + 2*x
```

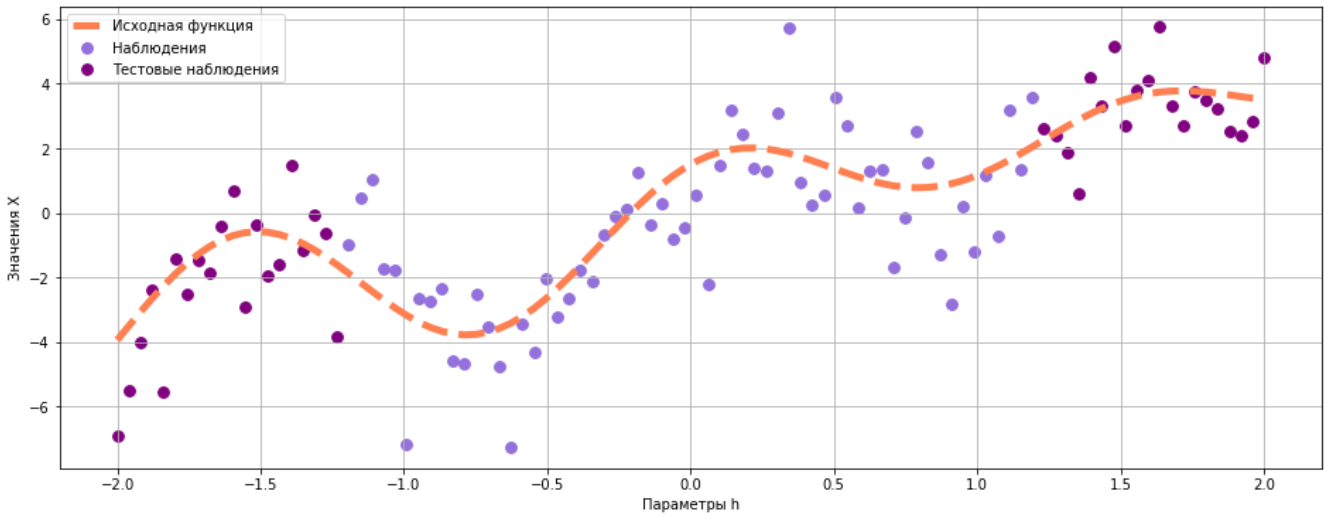


График 2-1. Демонстрация тренировочных и тестовых значений и исходной функции

В результате получилось смоделировать выборку по данным мне условиям. На графике приведены значения исходной функции, она изображена оранжевым пунктиром. Лавандовыми точками изображены наблюдения тренировочной выборки, а фиолетовыми изображены тесовые наблюдения, которые находятся слева и справа от тестовой. Все значения смоделированы согласно условиям, что на исходную функцию влияют ошибки нормального распределения.

2.2 Метод наименьших квадратов

$$\text{Для модели } Y_k = \theta_0 + \theta_1 \cdot h_k,\ k = 1..60$$

– найти МНК – оцеку неизвестных параметров

Найдем ее по формуле :

$$\hat{\Theta} = (H^T \cdot H)^{-1} H^T Y$$

```
H_T = np.array([[1 for i in range(60)], df['t']])
H = H_T.T
H.shape, H_T.shape
```

```
((60, 2), (2, 60))
```

```
W = np.dot(H_T, H)
W = np.linalg.inv(W)
HX = np.dot(H_T, df.x)
tetha = np.dot(W, HX)
print('МНК-оценка:', tetha[0], tetha[1])
```

```
МНК-оценка: -0.5320956476704726 2.173201718708428
```

Полученная МНК-оценка для данной модели получилась: $\theta = [-0.5321\ \ 2.1732]^T$.

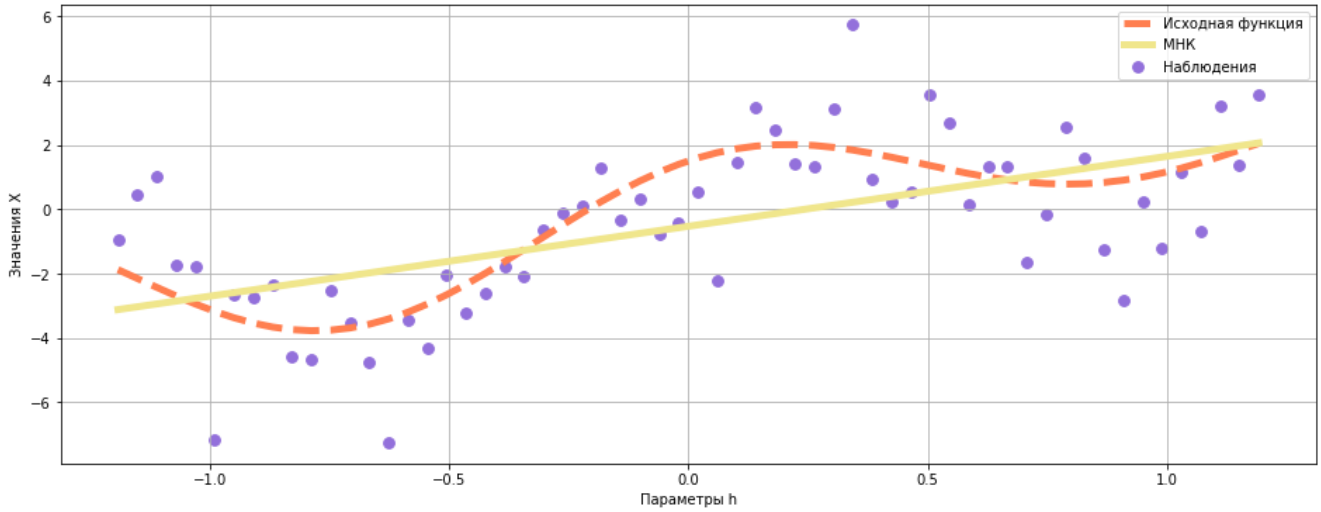


График 2-2. Демонстрация МНК-оценки для модели $X = \theta_0 + \theta_1 t$

Вычислим коэффициент детерминации и найти оценку ковариационной матрицы МНК–оценки.

```
R = 1 - (df.shape[0]-1)/(df.shape[0] - 2)*df['(X_true-Xmnk)^2'].sum()/df['(X_true-Xavg)^2'].sum()
print('Коэффицент детерминации')
print(R)
```

```
Коэффицент детерминации
0.3298872150661596
```

```
sigma_e2 = df['(X_true-Xmnk)^2'].sum()/(k-2)
print(f'Несмещенная оценка квадратов ошибки для МНК - {sigma_e2}')
```

```
Несмещенная оценка квадратов ошибки для МНК - 2.732958031420202
```

```
cov_matrix = sigma_e2 * W
print('Ковариацонная матрица')
print(cov_matrix)
```

```
Ковариацонная матрица
[[ 4.55493005e-02 -5.50859034e-18]
 [-5.50859034e-18  9.30318202e-02]]
```

Значение коэффициента детерминации получилось 0.33. это значение не близко к 1, что показывает, что присутствует несильное превосходство модели над тривиальной p=1(X_{avg}). Несмещенная оценка квадратов ошибок получилась равная 2.73. По значениям корреляционной матрицы видно значение дисперсий МНК значений, а также видно, что значения между собой не коррелируют.

Найти значения информационных критериев.

Информационный критерий — применяемая в эконометрике (статистике) мера относительного качества эконометрических (статистических) моделей, учитывающая степень «подгонки» модели под данные с корректировкой (штрафом) на используемое количество оцениваемых параметров. То есть критерии основаны на неком компромиссе между точностью и сложностью модели. Критерии различаются тем, как они обеспечивают этот баланс.

```
AIC = 2*2/60+np.log(sigma_e2)
print(f'AIC = {AIC}')
```

```
AIC = 1.0720512172635157
```

С помощью критерия Фишера проверить гипотезу $H0$: $\theta 0 = 0, \theta 1 = 0$:

```
A = np.array([[1, 0],[0, 1]])
a = np.array([[0],[0]])
f1 = A.dot(np.resize(tetha,(2,1)))-a
f2 = A.dot(np.linalg.inv(H.T.dot(H))).dot(A)
F = f1.T.dot(np.linalg.inv(f2)).dot(f1)/(df['(X_true-Xavg)^2'].sum()*2/58)
print('Критерий Фишера')
print(F[0][0])
```

```
Критерий Фишера
11.107802246483208
```

```
rv = f(2, 58)
vals = f.ppf([0.95], 2, 58)
vals[0]
if F[0][0] < vals[0]:
    print('H0 принимается')
else:
    print('H0 не принимается')
```

```
H0 не принимается
```

В результате расчетов видно, что значение критерия Фишера получилось 11.1. Таким образом значение критерия Фишера попадает в критическую область, из этого следует, что гипотеза $H0$ на уровне значимости $\alpha = 0.05$ не принимается.

Построить доверительный интервал надежности 0.95 и 0.8

```
sigma_1 = [np.dot(np.dot(np.array([1, i]), cov_matrix), np.array([[1], [i]]))[0] for i in t]
sigma_1 = np.sqrt(sigma_1) * 1.96
```

```
sigma_2 = [np.dot(np.dot(np.array([1, i]), cov_matrix), np.array([[1], [i]]))[0] for i in t]
sigma_2 = np.sqrt(sigma_2) * 1.28
```

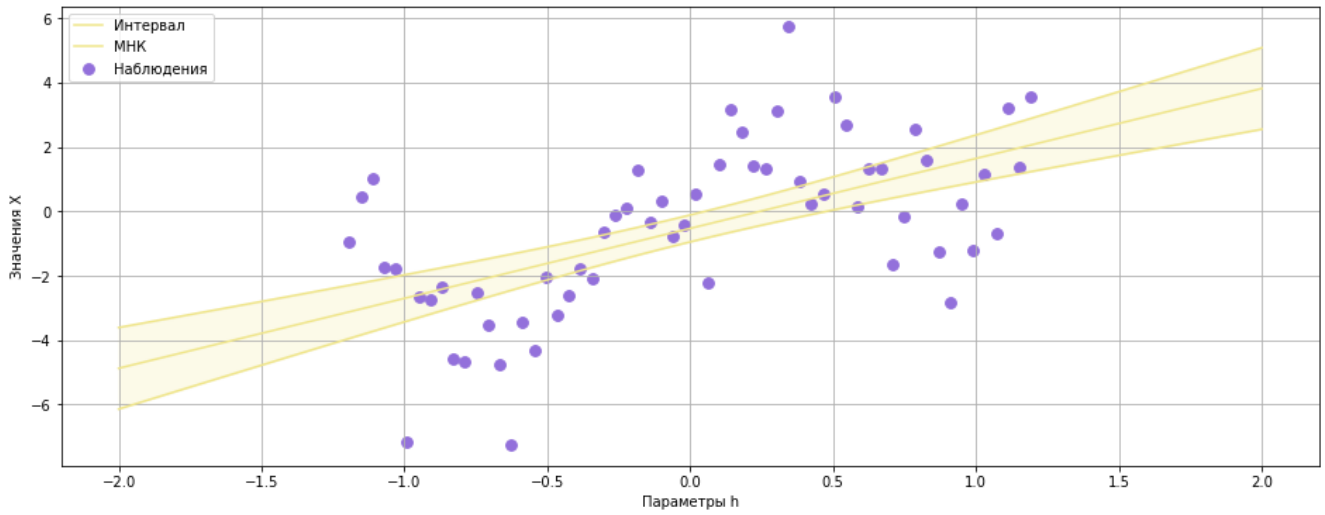


График 2-3. Доверительный интервал уровня 0.95

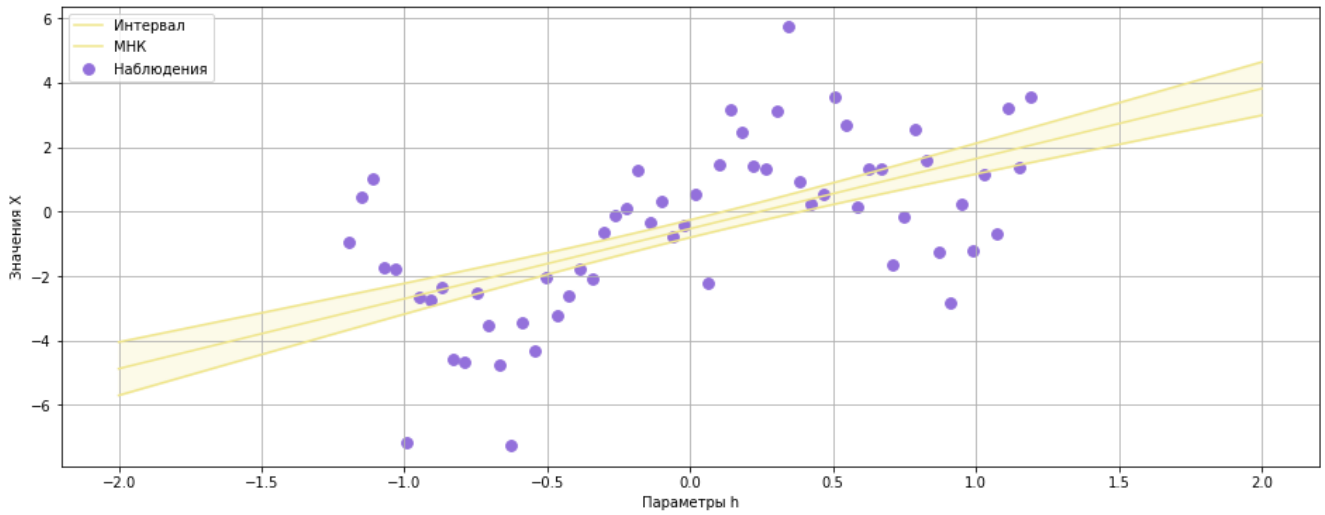


График 2-4. Доверительный интервал уровня 0.8

Построить оценку метода наименьших модулей.

Для построения этой оценки я использовал квантильную регрессию. Построив модель со значением 0.5, мы как раз получим МНМ-оценку.

```
reg = QuantileRegressor(quantile=0.5, alpha=0)
```

```
reg.fit(H, df.x)
```

```
QuantileRegressor(alpha=0)
```

```
tetha_MNM = reg.coef_
```

```
print('МНМ-оценка:', tetha_MNM[0], tetha_MNM[1])
```

```
МНМ-оценка: -0.20224671552378415 2.3813878414275225
```

Полученная МНМ-оценка для данной модели получилась: $\theta = [-0.20225 \;\; 2.3814]^T$.

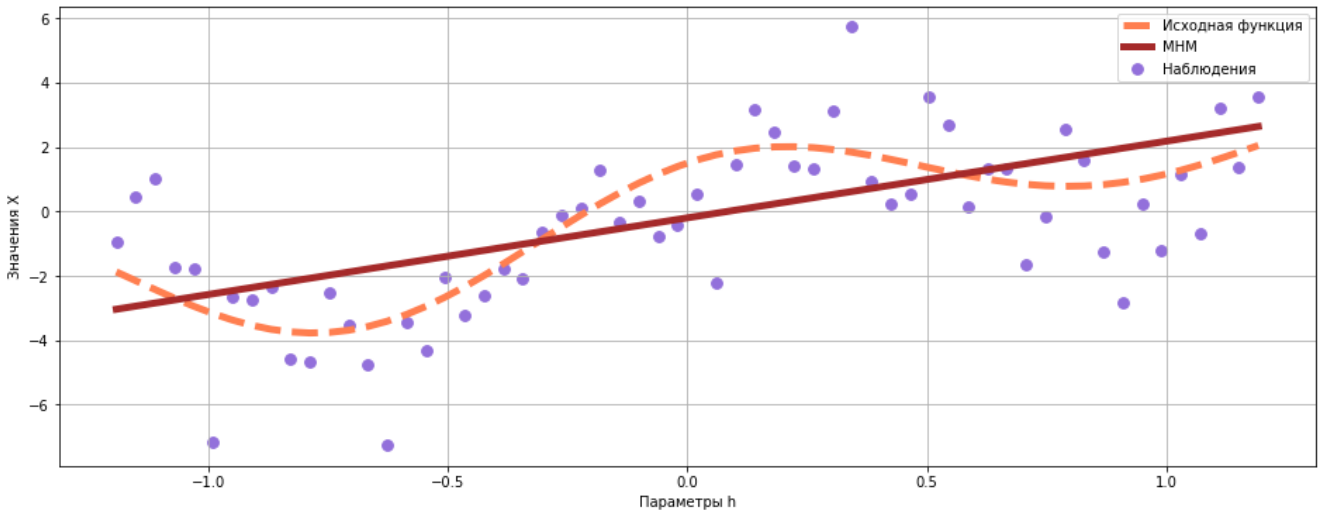


График 2-5. Демонстрация МНМ-оценки для модели $X = \theta_0 + \theta_1 h$

Оценить качество построенных регрессий на тестовой выборке:

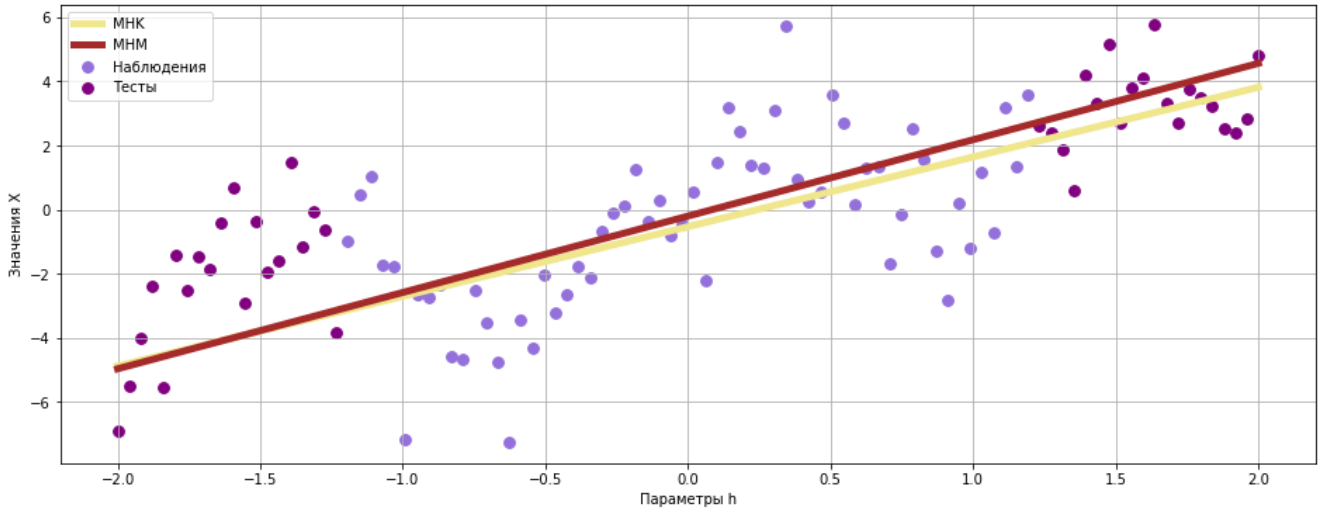


График 2-5. Демонстрация МНМ-оценки и МНМ-оценки для модели $X = \theta_0 + \theta_1 h$ на тестовой выборке

```
e_mnk = df_test.x - df_test.Xmnk
e_mnm = df_test.x - df_test.Xmnm
```

```
mse_mnk = (e_mnk**2).sum()/40
mse_mnm = (e_mnm**2).sum()/40
```

```
print(f'MSE на тесте для МНК - {mse_mnk}')
print(f'MSE на тесте для МНМ - {mse_mnm}')
```

```
MSE на тесте для МНК - 4.232402056048923
MSE на тесте для МНМ - 4.24110521551521
```

По полученным результатам видно, что обе модели выдают близкие по значениям результаты, но МНК-оценка лучше предсказала тесты на 0.01. Посмотрим на остатки.

Для остатков $\widehat{ek} = Xk - \widehat{Xk}$:

Построить гистограмму и ядерную оценку

График 2-6. Гистограмма и ядерная оценка для остатков МНК-оценки

По графику видно, что остатки очень похожи на нормальное распределение. Рассмотрим критерии, определяющие это, критерии определения автокорреляции и гетероскидотичности.

Критерий Дагостини

```
res = normaltest(df['X_true-Xmnk'])
if res.pvalue > 0.05:
    print('Есть основания считать, что ошибки имеют нормальное распределение')
else:
    print('Ошибки не имеют нормальное распределение')
```

```
Есть основания считать, что ошибки имеют нормальное распределение
```

Критерий Дарбина-Уотсона

```
e__ = []
for i in range(1, df.shape[0]):
    e__.append((df['X_true-Xmnk'][i] - df['X_true-Xmnk'][i-1])**2)
DW = sum(e__)/df['(X_true-Xmnk)^2'].sum()
print(f'Значение Дарбина-Уотсона = {DW}')
```

```
Значение Дарбина-Уотсона = 0.9741712652152426
```

```
dl = 1.51
du = 1.65
if 0 < DW < dl:
    print('Есть положительная корреляция')
elif dl < DW < du:
    print('Гипотиза о наличии корреляции не отвергается и не принимается')
elif du < DW < 4-du:
    print('Нет корреляции')
elif 4-du < DW < 4-dl:
    print('Гипотиза о наличии корреляции не отвергается и не принимается')
elif 4-dl < DW < 4:
    print('Есть положительная корреляция')
```

```
Есть положительная корреляция
```

Критерий Бартлетта

```
a = df['X_true-Xmnk'].to_numpy()
a = a[0:60:3]
b = a[1:60:3]
c = a[2:60:3]
res = bartlett(a, b, c)
if res.pvalue > 0.05:
    print('Есть основания считать, что остатки равномерно разбросаны')
else:
    print('Ошибки не имеют нормальное распределение')
```

```
Есть основания считать, что остатки равномерно разбросаны
```

Согласно каждому из критериев мы можем сделать вывод. Первый критерий был критерий Дагостини, на основании которого определяется наличие оснований для нормального распределения. Получилось, что у нас есть основания считать, что ошибки имеют нормальное распределение. С помощью критерия Дарбина–Уотсона необходимо определить наличие автокорреляции. Согласно этому критерию у нас получилось значение 0.9, это попадает в первую область, что следует, что у нас присутствует положительная корреляция ($\rho \neq 0$). С помощью критерия Бартлетта мы пытались определить наличие гетероскедастичности. По результатам получилось, что у нас есть основания, что остатки равномерно разбросаны (наблюдения однородны).

2.3 Полиномиальная регрессия

Подберем параметр для модели $X = \theta_0 + \theta_1 h + \theta_2 h^2 + .. + \theta_p h^p$. Первый способ – с.к. ошибки на тренировочной выборке.

```
er1 = 2
er2 = 1
i = 2
while er1 - er2 > 0.00001:
    poly_reg=PolynomialFeatures(degree = i)
    X_poly=poly_reg.fit_transform(np.resize(df.t, (60,1)))
    lin_reg = LinearRegression()
    lin_reg.fit(X_poly,df.x)
    y_pred = lin_reg.predict(X_poly)
    er1 = er2
    er2 = (np.sum(df.x-y_pred)**2)/(60-i)**0.5
    i+=1
print(i)
```

```
4
```

```
tetha_poly1 = lin_reg.coef_
```

```
print('МНК-оценка:', tetha_poly1[0], tetha_poly1[1], tetha_poly1[2], tetha_poly1[3])
```

```
МНК-оценка: 0.0 5.104248338580225 -0.6061627956691069 -3.3270624387755325
```

Полученная МНК-оценка для данной модели получилась: $\theta = [\,0 \;\; 5.1042 \;\; -0.6062 \;\; -3.327]^T$.

Получившееся значение параметра получилось равное 4 => Полином 3 степени. Выведем график, который будем еще и включать значения тестовой выборки.

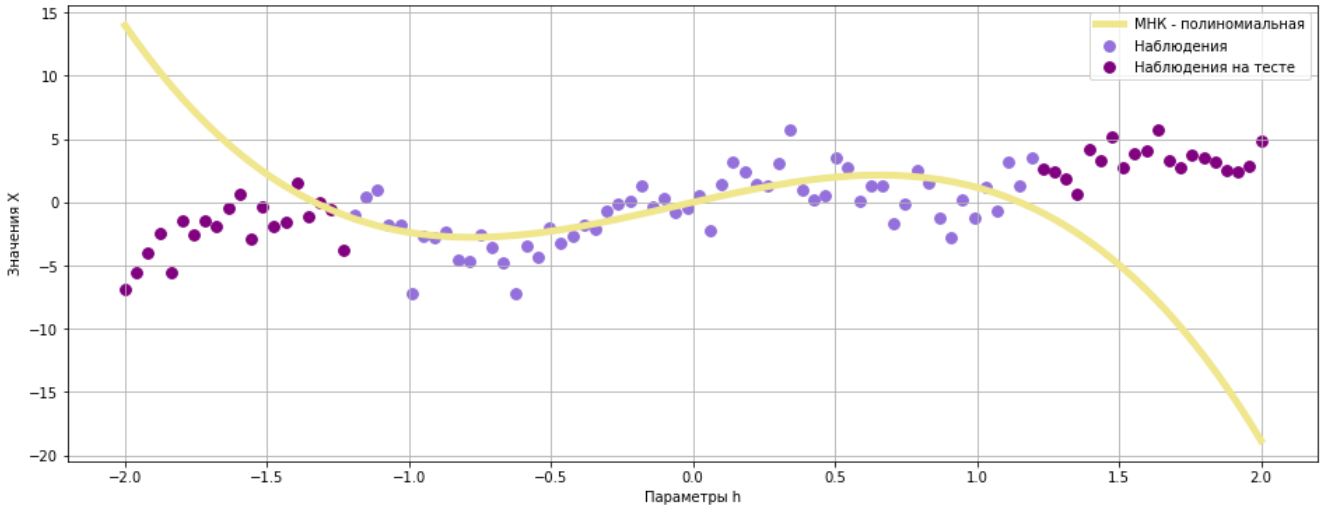


График 2-7. График для модели 3 степени

Как получилось из решения, так и на графике видно, что модель хорошим образом предсказывает тренировочные наблюдения, а на тесте очень сильно ошибается. Попробуем поиск параметра p по подсчету Mse на тесте.

```
Mse1 = 2
Mse2 = 1
i = 2
while Mse1 - Mse2 > 0.00001:
    poly_reg=PolynomialFeatures(degree = i)
    X_poly=poly_reg.fit_transform(np.resize(df.t, (60,1)))
    lin_reg = LinearRegression()
    lin_reg.fit(X_poly,df.x)
    X_poly=poly_reg.fit_transform(np.resize(df_test.t, (40,1)))
    y_pred = lin_reg.predict(X_poly)
    Mse1 = Mse2
    Mse2 = (np.mean(df_test.x-y_pred)**2)
    i+=1
print(i)
```

```
3
```

```
tetha_poly2 = lin_reg.coef_
print('МНК-оценка:', tetha_poly2[0], tetha_poly2[1], tetha_poly2[2])
```

```
МНК-оценка: 0.0 2.1732017187084285 -0.6061627956691091
```

Полученная МНК-оценка для данной модели получилась: $\theta = [\,0 \;\; 2.1732 \;\; -0.6062\,]^T$.

Как можно заметить обе модели присваивают θ_0 значение 0 (нет смещения). Таким образом модель приобретает вид $X = \theta_1 h + \theta_2 h^2$.

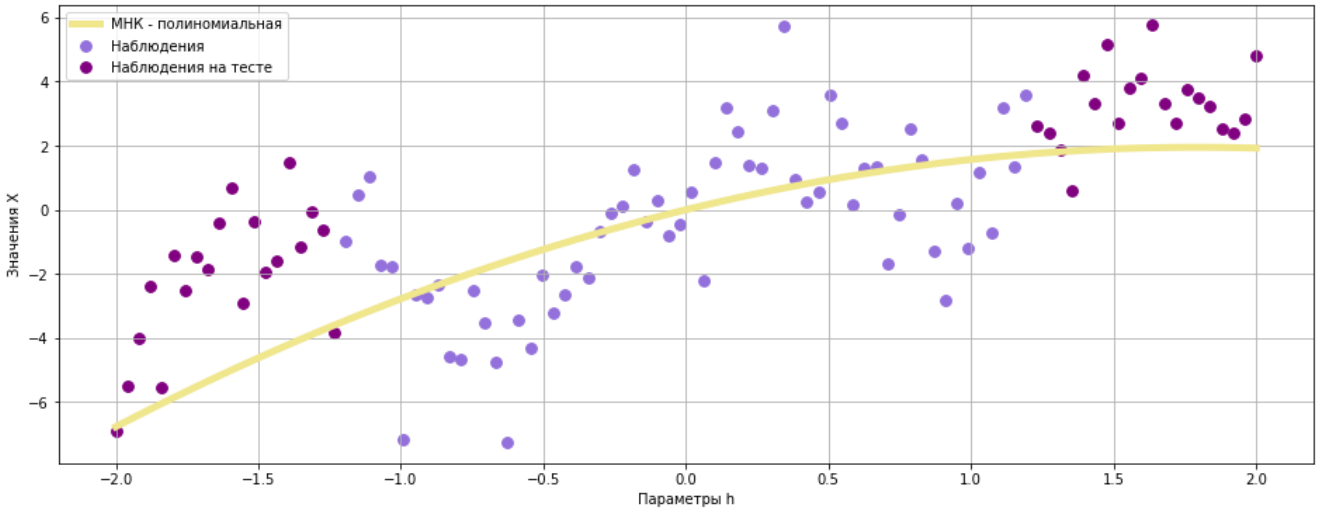


График 2-8. График для модели 2 степени

Так как необходимо спроектировать поведение исходной модели на любых значениях, поэтому Я выберу значение p=2, потому что видно, что на тесте у нас получится наилучшее значение.

Найдем значение несмещённой с.к ошибки на тренировочной выборке.

```
sigma_e2 = np.array([a[i]**2 for i in range(len(a))]).sum()/(60-3)
print(f'Несмещенная оценка квадратов ошибки для МНК - {sigma_e2}')
```

```
Несмещенная оценка квадратов ошибки для МНК - 3.783252135910052
```

Как видно, что все равно такая модель все равно имеет большую ошибку, чем первая.

Для остатков:

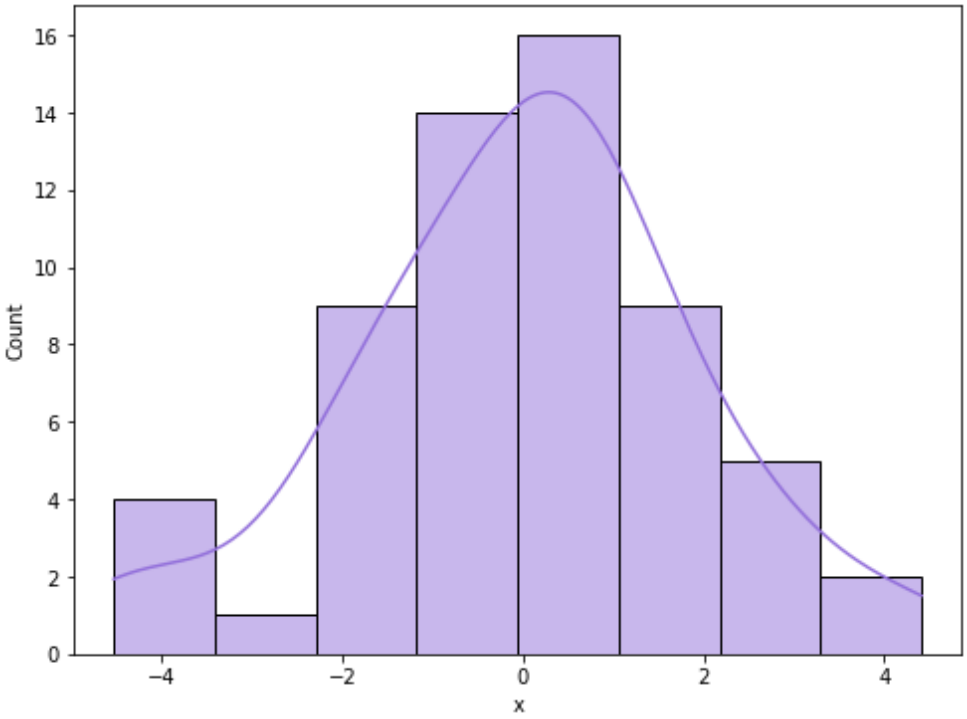


График 2-9. Гистограмма и ядерная оценка ошибок

Критерий Дагостини

```
res = normaltest(a)
if res.pvalue > 0.05:
    print('Есть основания считать, что ошибки имеют нормальное распределение')
else:
    print('Ошибки не имеют нормальное распределение')
```

```
Есть основания считать, что ошибки имеют нормальное распределение
```

Критерий Дарбина-Уотсона

```
e__ = []
for i in range(1, a.shape[0]):
    e__.append((a[i] - a[i-1])**2)
DW = sum(e__)/sum([a[i]**2 for i in range(len(a))])
print(f'Значение Дарбина-Уотсона = {DW}')
```

```
Значение Дарбина-Уотсона = 1.2188379116161898
```

```
dl = 1.48
du = 1.69
if 0 < DW < dl:
    print('Есть положительная корреляция')
elif dl < DW < du:
    print('Гипотиза о наличии корреляции не отвергается и не принимается')
elif du < DW < 4-du:
    print('Нет корреляции')
elif 4-du < DW < 4-dl:
    print('Гипотиза о наличии корреляции не отвергается и не принимается')
elif 4-dl < DW < 4:
    print('Есть положительная корреляция')
```

```
Есть положительная корреляция
```

Критерий Бартлетта

```
a1 = a[0:60:3]
b = a[1:60:3]
c = a[2:60:3]
res = bartlett(a1, b, c)
if res.pvalue > 0.05:
    print('Есть основания считать, что остатки равномерно разбросаны')
else:
    print('Ошибки не имеют нормальное распределение')
```

```
Есть основания считать, что остатки равномерно разбросаны
```

По критерию Дагостини вышло, что у нас есть основания считать ошибки нормально распределенными. По Дарбину–Уотсону значение попало в 1 область, из чего следует, что есть положительная корреляция. По критерию Бартлетта – есть основания, что остатки равномерно разбросаны.

Исследуем полученную модель на мультиколлинеарность

```
W = np.dot(X_poly.T, X_poly)
con = np.linalg.cond(W)
print(f'cond(W) = {con}')
if con > 1:
    print('Матрицу следует признать мультиколлинеарной')
else:
    print('Нет мультиколлинеарной')
```

```
cond(W) = 32.72621002458616
Матрицу следует признать мультиколлинеарной
```

По полученному значению $cond(W)$ получилось, что матрица является мультиколлиниарной, поэтому необходимо использовать ридж–оценку, чтобы избавиться от нее.

```
poly_reg=PolynomialFeatures(degree = 3)
X_poly=poly_reg.fit_transform(np.resize(df.t, (60,1)))
ridge = Ridge(alpha=1.0)
ridge.fit(X_poly, df.x)
y_pred = ridge.predict(X_poly)
```

```
tetha_ridge = ridge.coef_
```

```
print('МНК-оценка Ридж:', tetha_ridge[0], tetha_ridge[1], tetha_ridge[2])
```

```
МНК-оценка Ридж: 0.0 3.8612196607556006 -0.5576576100126747
```

Полученная МНК-оценка для данной модели получилась с Ридж регуляризацией: $\theta = [\ 0\ \ 3.8612\ \ -0.5577\ ]^T$.

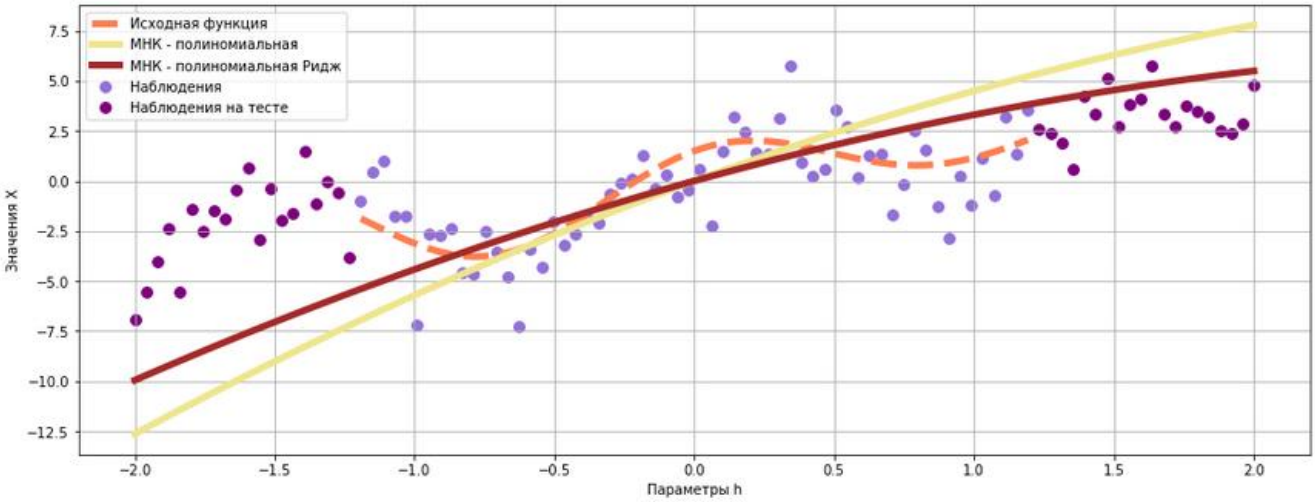


График 2-10. Сравнение МНК-оценок

```
a = df.x - y_pred
sigma_e2 = np.array([a[i]**2 for i in range(len(a))]).sum()/(60-3)
print(f'Несмещенная оценка квадратов ошибки для МНК метод Ридж - {sigma_e2}')
```

```
Несмещенная оценка квадратов ошибки для МНК метод Ридж - 3.9136483874141956
```

В результате регуляризации несмещенная с.к. ошибка на тренировочных наблюдения увеличилась, но по графику видно, что на тестовые лучше предсказываются.

```
print(f'MSE на тесте для МНК - {a2.sum()/40}')
```

```
MSE на тесте для МНК - 18.778892849066427
```

Из–за использования ридж–оценки построенная линия притянулась к левым тестовым значениям, что улучшило прогноз модели. Поэтому наилучшей полиномиальной моделью оказалась $Yk = \theta 0 + \theta 1hk + \theta 2hk^2$ с использованием ридж–регуляризацией. Но в сравнении с первой моделью, эта модель проигрывает МНК и МНМ оценкам. Значение MSE получилось 18.8.

2.4 Регрессия для наблюдений с выбросами

Моделируем данные.

```
delta = 0.08
e0 = np.array(np.random.normal(mu, sigma, 100))
e1 = np.array(np.random.normal(mu, 10*sigma, 100))
e = (1-delta)*e0 + delta*e1
```

$$\text{Для модели } Y_k = \theta_0 + \theta_1 \cdot h_k,\ k = 1..60$$

– найти МНК – оцеку неизвестных параметров

Найдем ее по формуле :

$$\hat{\Theta} = (H^T \cdot H)^{-1} H^T Y$$

```
H_T = np.array([[1 for i in range(60)], df['t']])
H = H_T.T
H.shape, H_T.shape
W = np.dot(H_T, H)
W = np.linalg.inv(W)
HX = np.dot(H_T, df2.x)
tetha = np.dot(W, HX)
print('МНК-оценка:', tetha[0], tetha[1])
```

```
МНК-оценка: -0.2545524223805595 2.658961772560884
```

Полученная МНК-оценка для данной модели получилась: $\theta = [-0.25455\ \ 2.6589]^T$.

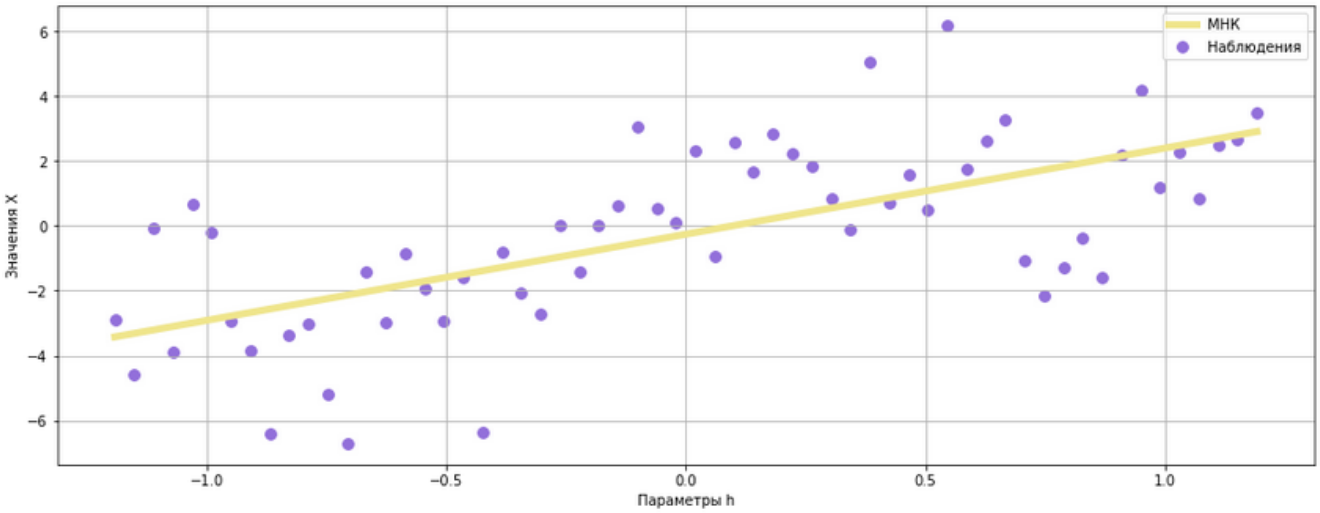


График 2-11. Демонстрация выборки и МНК-оценки

```
sigma_e2 = df2['(X_true-Xmnk)^2'].sum()/(60-2)
print(f'Несмещенная оценка квадратов ошибки для МНК - {sigma_e2}')
```

```
Несмещенная оценка квадратов ошибки для МНК - 4.614537281320333
```

Для остатков.

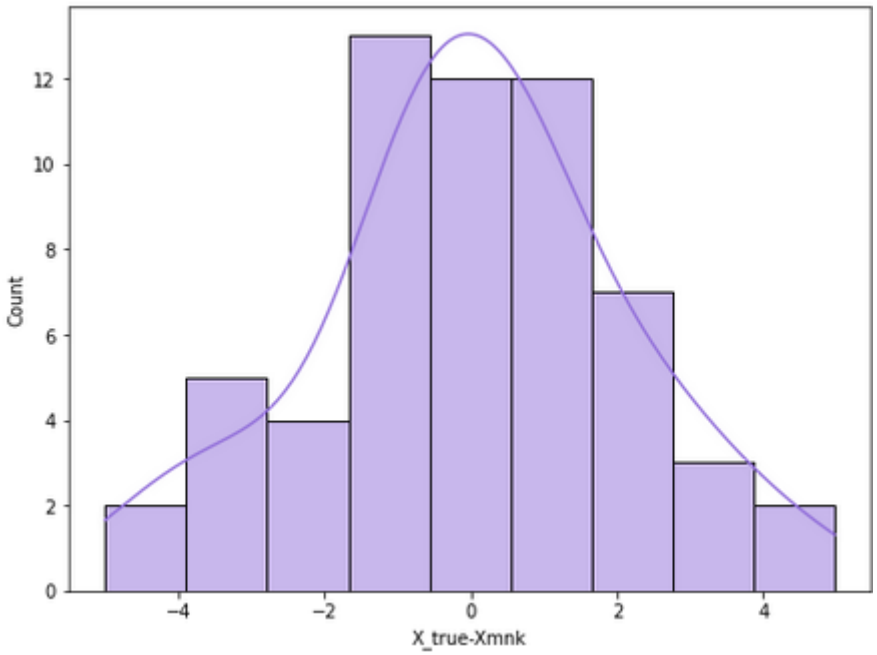


График 2-11. Гистограмма и ядерная оценка ошибок

Критерий Дагостини

```
res = normaltest(df2['X_true-Xmnk'])
if res.pvalue > 0.05:
    print('Есть основания считать, что ошибки имеют нормальное распределение')
else:
    print('Ошибки не имеют нормальное распределение')
```

```
Есть основания считать, что ошибки имеют нормальное распределение
```

Критерий Дарбина-Уотсона

```
e__ = []
for i in range(1, df2.shape[0]):
    e__.append((df2['X_true-Xmnk'][i] - df2['X_true-Xmnk'][i-1])**2)
DW = sum(e__)/df2['(X_true-Xmnk)^2'].sum()
print(f'Значение Дарбина-Уотсона = {DW}')
```

```
Значение Дарбина-Уотсона = 1.4887236384075972
```

```
dl = 1.51
du = 1.65
if 0 < DW < dl:
    print('Есть положительная корреляция')
elif dl < DW < du:
    print('Гипотиза о наличии корреляции не отвергается и не принимается')
elif du < DW < 4-du:
    print('Нет корреляции')
elif 4-du < DW < 4-dl:
    print('Гипотиза о наличии корреляции не отвергается и не принимается')
elif 4-dl < DW < 4:
    print('Есть положительная корреляция')
```

```
Есть положительная корреляция
```

Критерий Бартлетта

```
a = df2['X_true-Xmnk'].to_numpy()
a = a[0:60:3]
b = a[1:60:3]
c = a[2:60:3]
res = bartlett(a, b, c)
if res.pvalue > 0.05:
    print('Есть основания считать, что остатки равномерно разбросаны')
else:
    print('Ошибки не имеют нормальное распределение')
```

```
Есть основания считать, что остатки равномерно разбросаны
```

По критерию Дагостини вышло, что у нас есть основания считать ошибки нормально распределенными. По Дарбину–Уотсону значение попало в 1 область, из чего следует, что есть положительная корреляция. По критерию Бартлетта – есть основания, что остатки равномерно разбросаны.

Попробуем отбросить выбросы.

```
k = abs(df2['X_true-Xmnk']).median()
print(k)
```

```
1.1768325869149838
```

```
df_norm = df2[abs(df2['X_true-Xmnk']) < 2*k/0.675]
df_lol = df2[abs(df2['X_true-Xmnk']) > 2*k/0.675]
```

```
H_T = np.array([[1 for i in range(len(df_norm))], df_norm.t])
H = H_T.T
l = 60 - H.shape[0]
H.shape, H_T.shape, l
```

```
((51, 2), (2, 51), 9)
```

```
W = np.dot(H_T, H)
W = np.linalg.inv(W)
HX = np.dot(H_T, df_norm.x)
tetha_new = np.dot(W, HX)
print('МНК-оценка:', tetha_new[0], tetha_new[1])
```

```
МНК-оценка: -0.16605365296014135 2.5437147160171945
```

Полученная МНК-оценка для данной модели получилась: $\theta = [\,-0.166\;\; 2.5437]^T$.

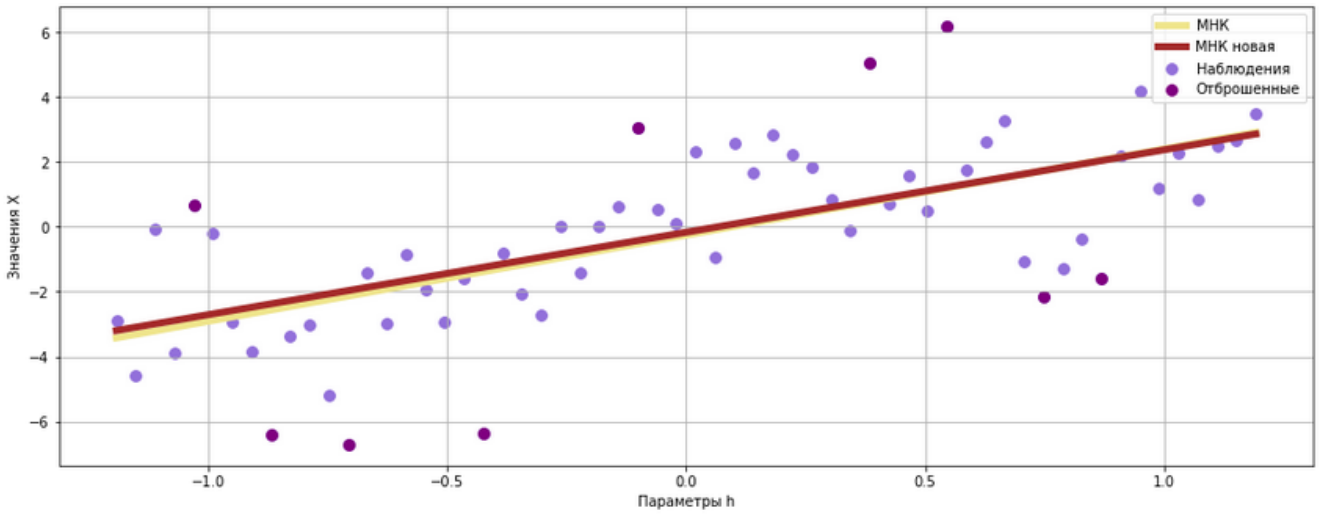


График 2-12. МНК-оценка после того, как мы отбросили большие значения

Для остатков:

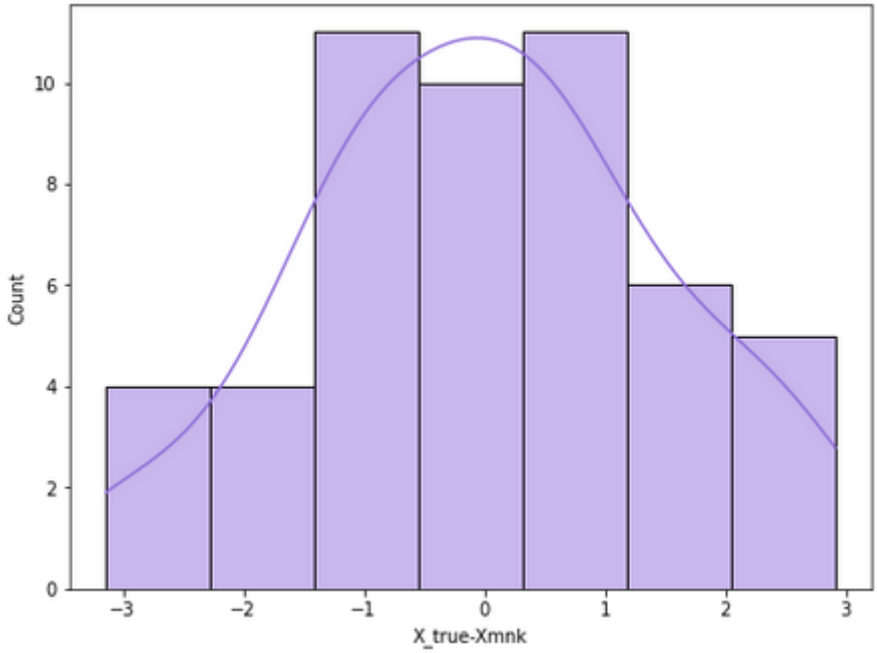


График 2-13. Гистограмма и ядерная оценка ошибок после того, как мы отбросили большие значения

Критерий Дарбина-Уотсона

```
e_ = (df_norm.x - df_norm['Xmnk']).to_numpy()
```

```
e__ = []
for i in range(1, len(e_)):
    e__.append((e_[i] - e_[i-1])**2)
DW = sum(e__)/df_norm['(X_true-Xmnk)^2'].sum()
print(f'Значение Дарбина-Уотсона = {DW}')
```

```
Значение Дарбина-Уотсона = 1.7070391503821578
```

Критерий Дарбина-Уотсона

```
e_ = (df_norm.x - df_norm['Xmnk']).to_numpy()
```

```
e__ = []
for i in range(1, len(e_)):
    e__.append((e_[i] - e_[i-1])**2)
DW = sum(e__)/df_norm['(X_true-Xmnk)^2'].sum()
print(f'Значение Дарбина-Уотсона = {DW}')
```

```
Значение Дарбина-Уотсона = 1.7070391503821578
```

```
dl = 1.51 - l*0.004
du = 1.65 - l*0.02
du, dl
```

```
(1.47, 1.474)
```

```
if 0 < DW < dl:
    print('Есть положительная корреляция')
elif dl < DW < du:
    print('Гипотиза о наличии корреляции не отвергается и не принимается')
elif du < DW < 4-du:
    print('Нет корреляции')
elif 4-du < DW < 4-dl:
    print('Гипотиза о наличии корреляции не отвергается и не принимается')
elif 4-dl < DW < 4:
    print('Есть положительная корреляция')
```

```
Нет корреляции
```

Критерий Бартлетта

```
a = df_norm['X_true-Xmnk'].to_numpy()
a = a[0:51:3]
b = a[1:51:3]
c = a[2:51:3]
res = bartlett(a, b, c)
if res.pvalue > 0.05:
    print('Есть основания считать, что остатки равномерно разбросаны')
else:
    print('Ошибки не имеют нормальное распределение')
```

```
Есть основания считать, что остатки равномерно разбросаны
```

По критерию Дагостини вышло, что у нас есть основания считать ошибки нормально распределенными. По Дарбину–Уотсону значение попало в 3 область, из чего следует, что нет корреляция. По критерию Бартлетта – есть основания, что остатки равномерно разбросаны.

Построим МНМ-оценку

```
reg = QuantileRegressor(quantile=0.5, alpha=0)
reg.fit(H, df_norm.x)
tetha_MNM = reg.coef_
print('МНМ-оценка:', tetha_MNM[0], tetha_MNM[1])
```

```
МНМ-оценка: -0.14660330207976568 2.743621539343104
```

Полученная МНМ-оценка для данной модели получилась: $\theta = [\,-0.1466 \;\; 2.7436]^T$.

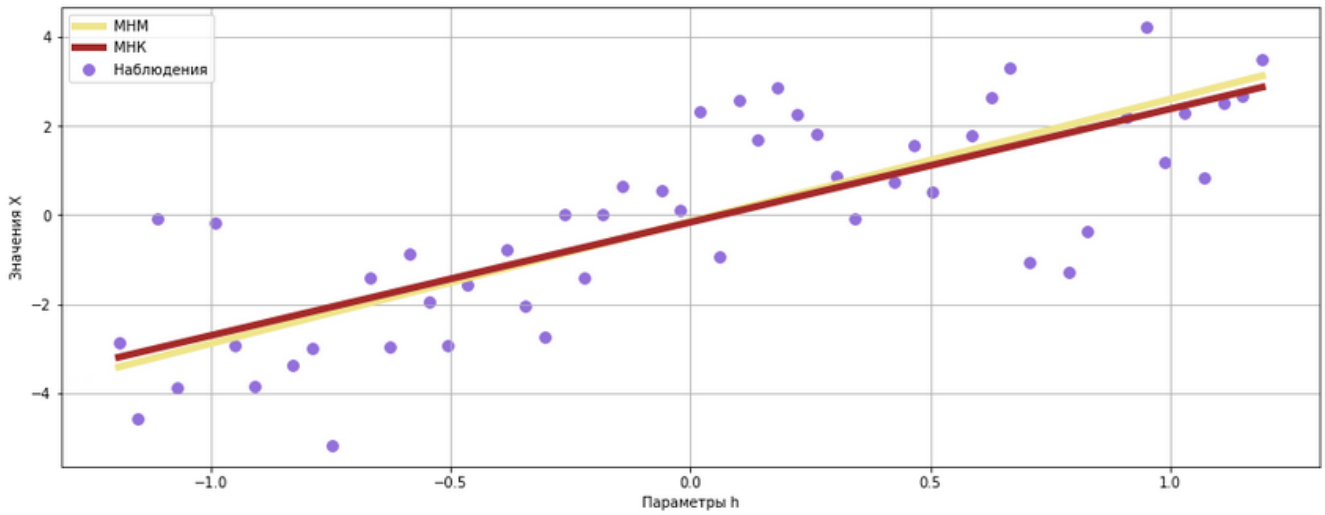


График 2-14. МНМ-оценка после того, как мы отбросили большие значения

Для остатков:

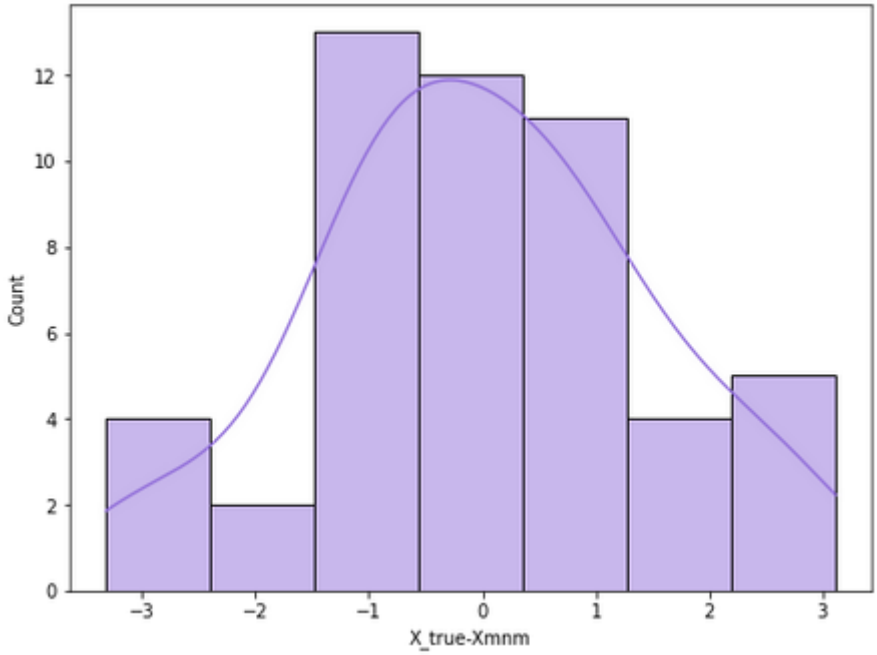


График 2-15. Гистограмма и ядерная оценка ошибок после того, как мы отбросили большие значения

Критерий Дагостини

```
res = normaltest(df_norm['X_true-Xmnm'])
if res.pvalue > 0.05:
    print('Есть основания считать, что ошибки имеют нормальное распределение')
else:
    print('Ошибки не имеют нормальное распределение')
```

```
Есть основания считать, что ошибки имеют нормальное распределение
```

Критерий Дарбина-Уотсона

```
e_ = (df_norm.x - df_norm['Xmnk']).to_numpy()
e__ = []
for i in range(1, len(e_)):
    e__.append((e_[i] - e_[i-1])**2)
DW = sum(e__)/df_norm['(X_true-Xmnk)^2'].sum()
print(f'Значение Дарбина-Уотсона = {DW}')
```

```
Значение Дарбина-Уотсона = 1.7070391503821578
```

```
if 0 < DW < dl:
    print('Есть положительная корреляция')
elif dl < DW < du:
    print('Гипотиза о наличии корреляции не отвергается и не принимается')
elif du < DW < 4-du:
    print('Нет корреляции')
elif 4-du < DW < 4-dl:
    print('Гипотиза о наличии корреляции не отвергается и не принимается')
elif 4-dl < DW < 4:
    print('Есть положительная корреляция')
```

```
Нет корреляции
```

Критерий Бартлетта

```
a = df_norm['X_true-Xmnm'].to_numpy()
a = a[0:51:3]
b = a[1:51:3]
c = a[2:51:3]
res = bartlett(a, b, c)
if res.pvalue > 0.05:
    print('Есть основания считать, что остатки равномерно разбросаны')
else:
    print('Ошибки не имеют нормальное распределение')
```

```
Есть основания считать, что остатки равномерно разбросаны
```

По критерию Дагостини вышло, что у нас есть основания считать ошибки нормально распределенными. По Дарбину–Уотсону значение попало в 3 область, из чего следует, что нет корреляция. По критерию Бартлетта – есть основания, что остатки равномерно разбросаны.

Выполняя данный блок, нам удалось улучшить модель, отбросив слишком большие значения. Тем самым урегулировали значение и избавились от корреляции ошибок, таким образом, модель стала устойчивой.

2.5 Квантильная регрессия

Моделируем выборку

```
df3 = pd.DataFrame(t[20:80], columns=['t'])
df3_test = pd.DataFrame(np.append(t[0:20], t[80:100]), columns=['t'])
```

```
e = np.random.normal(mu, sigma, 100)
```

```
neg_e = []
for e_ in e:
    if e_ < 0:
        neg_e.append(e_)
len(neg_e)
```

```
45
```

```
neg_e = random.sample(neg_e, int(len(neg_e)*0.9))
for i, e_ in enumerate(e):
    if e_ in neg_e:
        e[i] = -e_
```

Построить и отобразить на графике МНК и МНМ оценки:

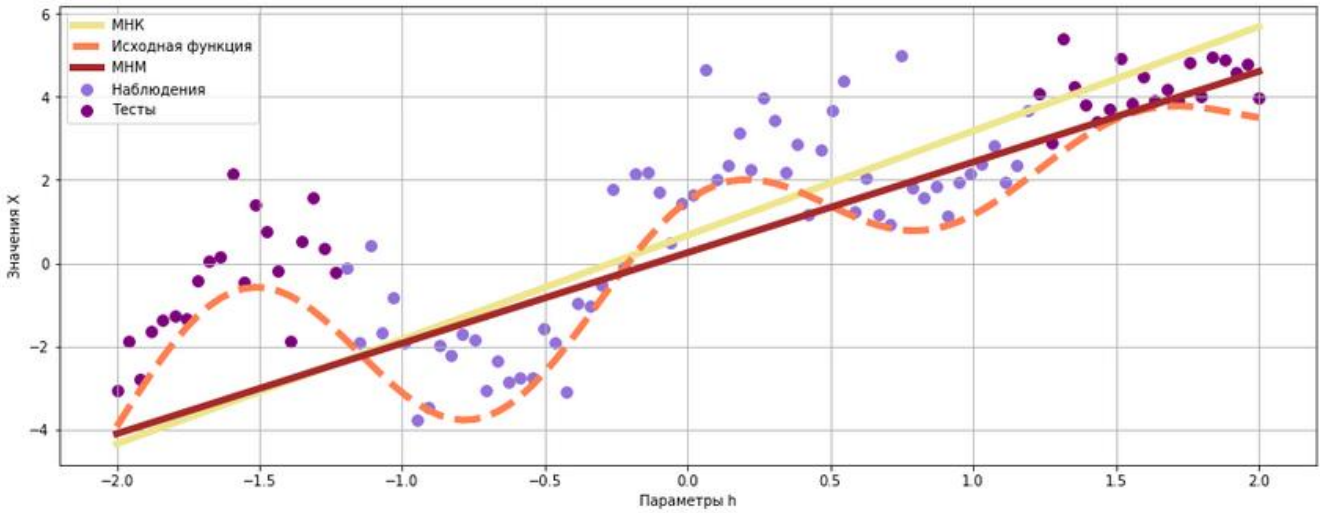


График 2-15. МНМ и МНК-оценки

```
sigma_mnke2 = df3['(X_true-Xmnk)^2'].sum()/(60-2)
print(f'Несмещенная оценка квадратов ошибки для МНК - {sigma_mnke2}')
```

```
Несмещенная оценка квадратов ошибки для МНК - 2.3118902325552777
```

```
sigma_mnme2 = df3['(X_true-Xmnm)^2'].sum()/(60-2)
print(f'Несмещенная оценка квадратов ошибки для МНК - {sigma_mnme2}')
```

```
Несмещенная оценка квадратов ошибки для МНК - 2.5478851908879143
```

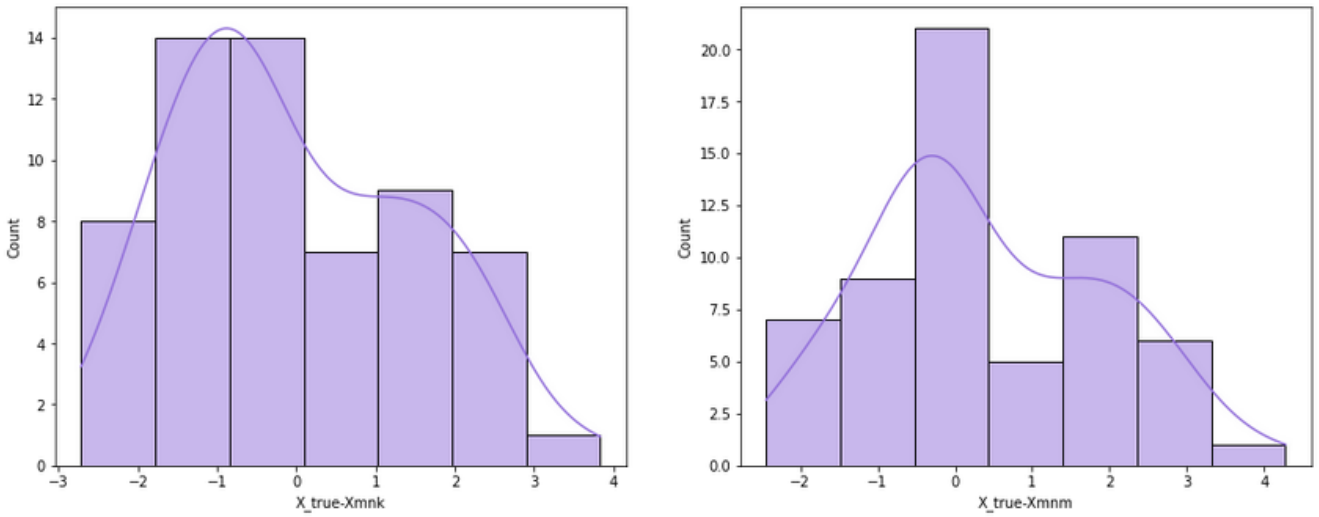
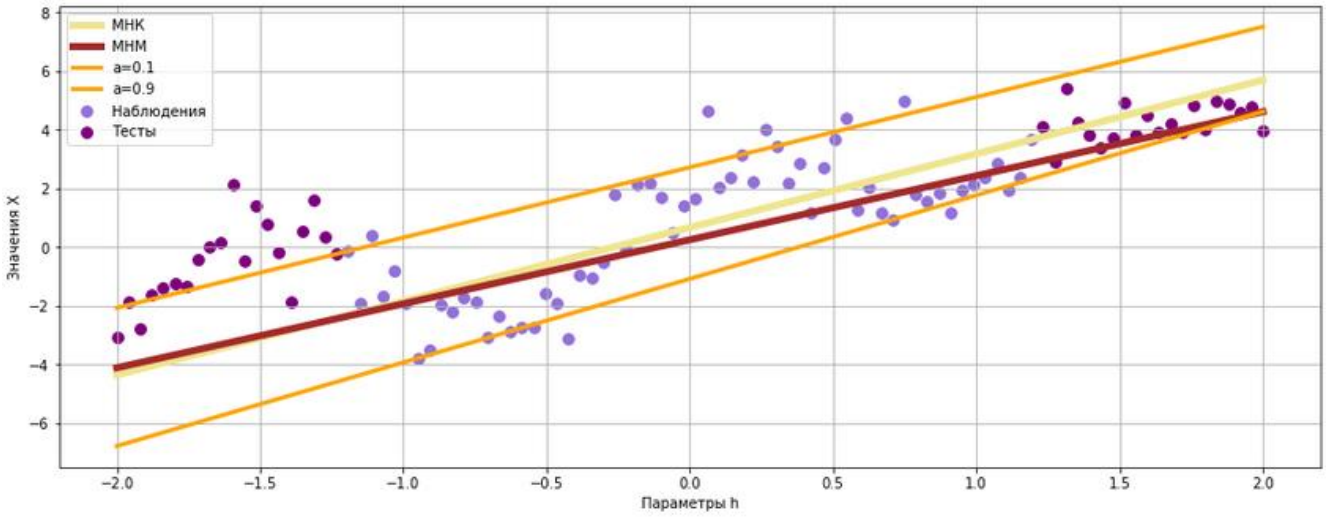


График 2-16. Гистограмма и ядерная оценка для МНК и МНМ-оценок

Построим квантильную регрессию со значениями квантилей 0.1 и 0.9:

```
reg = QuantileRegressor(quantile=0.1, alpha=0)
reg.fit(H, df3.x)
x_mnm01 = reg.predict(H2)
```

```
reg = QuantileRegressor(quantile=0.9, alpha=0)
reg.fit(H, df3.x)
x_mnm09 = reg.predict(H2)
```



Как видно из графика квантильная регрессия позволяет построить модель, которая отсекает какое-либо квантильное значение наблюдений. Ее легко можно интерпретировать как доверительный интервал.

Список литература

1. RDD - https://file.magzdb.org/ul/873/%D0%9A%D0%B2%D0%B0%D0%BD%D1%82%D0%B8%D0%BB%D1%8C%2007.pdf
2. RDD example - https://matheusfacure.github.io/python-causality-handbook/16Regression-Discontinuity-Design.html
3. Регрессия – методы восстановления зависимостей и записи лекций
4. Критерии - https://habr.com/ru/post/693402/
5. Полиномиальная регрессия- https://www.askpython.com/python/examples/polynomial-regression-in-python
6. Квантильная регрессия - https://www.codecamp.ru/blog/quantile-regression-in-python/?ysclid=lb9105pl19397307780
7. Гитхаб - https://github.com/youngtommypickles/regression