|  |  |  |
| --- | --- | --- |
| Министерство науки и высшего образования  Российской Федерации | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования | | |
| «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра теоретической и прикладной информатики | | |
|  | | |
| Лабораторная работа № 4 | | |
| по дисциплине «Статистические методы анализа данных» | | |
|  | | |
|  | | |
|  | | |
|  | Факультет: | ПМИ |
| Группа: | ПМИ-02 |
| Вариант: | 6 |
| Студент: | Сидоров Даниил, |
|  | Дюков Богдан |
| Преподаватель: | Попов Александр Александрович. |
|  |  |
|
|  |  |
| Новосибирск | | |
| 2023 | | |

1. **Постановка задачи**
2. Провести моделирование регрессионного процесса с гетероскедастичным возмущением.
3. Полученные данные проверить по тестам на наличие гетероскедастичности.
4. Оценить параметры регрессионной модели по доступному обобщенному МНК и по обыкновенному МНК.
5. Сравнить эффективность оценок в этих двух случаях по квадрату их расстояния до известных истинных значений параметров.

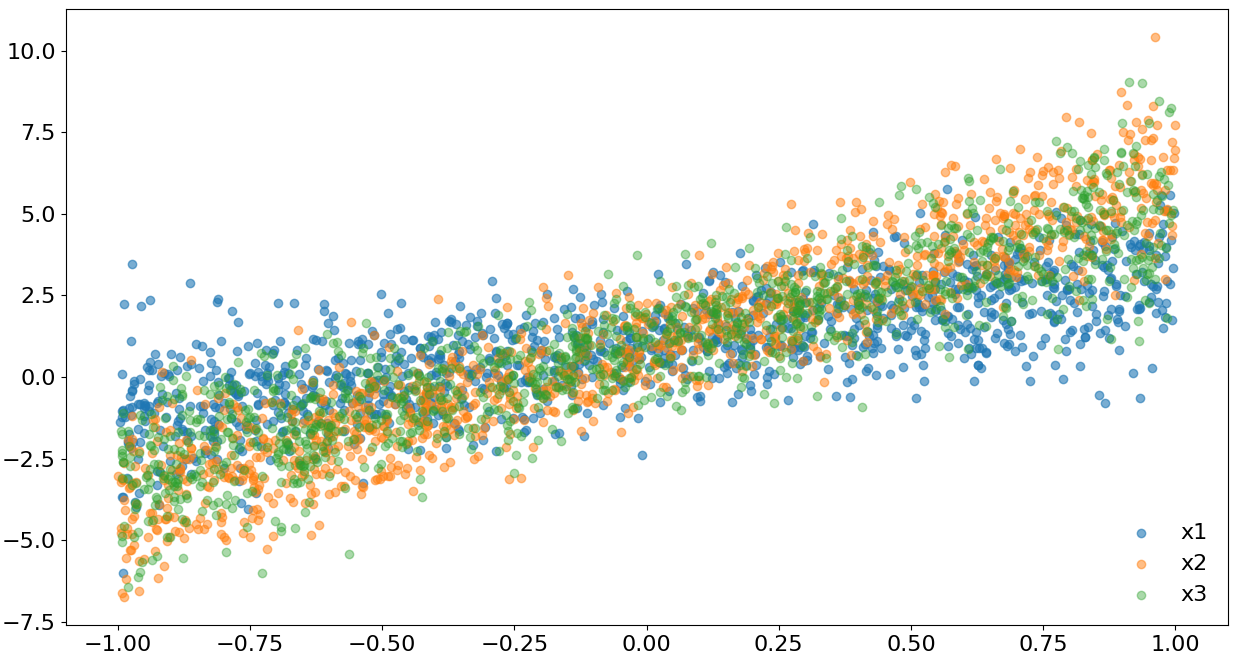
Дисперсия возмущений для 6 варианта – экспонента от взвешенной суммы квадратов факторов.

1. **Ход работы**

**Моделирование регрессионного процесса**

;

Точечная диаграмма зависимости отклика от значений факторов:



Можно видеть гетероскедастичность.

**Проверка гетероскедастичности с помощью теста Бреуша-Пагана**

Вектор известных переменных:

Вектор неизвестных параметров:

Гипотеза о гомоскедастичности в нашем случае имеет вид:

* оценим исходное уравнения по МНК, а также дисперсию:
* построим регрессию с откликом:

и найдем предсказанные значения нормированных квадратов остатков:

Гипотеза о гомоскедастичности будет принята, если

Получаем:

Гипотеза о гомоскедастичности не принимается.

**Проверка гетероскедастичности с помощью теста Голдфельда-Квандтона**

Предположение: источник нарушения гомоскедастичности взят в форме:

* Упорядочим последовательность наблюдений в соответствии с величиной:

.

* опустим наблюдений, оказавшихся в середине упорядоченной выборки:
* Гипотеза о гомоскедастичности будет принята, если:

где k = m – число параметров в регрессии.

Получаем:

Гипотеза о гомоскедастичности не принимается.

**Оценка параметров регрессионной модели**

Истинные значения параметров:

Оценки параметров регрессионной модели по обыкновенному МНК:

Оценки параметров регрессионной модели по доступному обобщенному МНК:

Сравним эффективность оценок в этих двух случаях по квадрату их расстояния до истинных значений:

Доступный обобщенный МНК эффективнее обыкновенного МНК.

1. **Код программы**

import numpy as np

from scipy.stats import f

import matplotlib.pyplot as plt

np.random.seed(10)

# Генерация комбинаций факторов

def generate\_random(n):

x1\_list = [np.random.uniform(-1, 1) for \_ in range(n)]

x2\_list = [np.random.uniform(-1, 1) for \_ in range(n)]

x3\_list = [np.random.uniform(-1, 1) for \_ in range(n)]

return map(np.array, [x1\_list, x2\_list, x3\_list])

def get\_dataframe\_for\_graphs(x1, x2, x3):

# Вычисление отклика без шума для каждого фактора при нулевых значениях остальных факторов

u1 = theta[0] + theta[1] \* x1 + theta[6] \* x1 \*\* 2

u2 = theta[0] + theta[2] \* x2 + theta[7] \* x2 \*\* 2

u3 = theta[0] + theta[3] \* x3

# Вычисление весов для каждого фактора

weights = np.array([0.8, 0.8, 0.9])

# Вычисление взвешенной суммы квадратов факторов

weighted\_sum\_squares\_1 = weights[0] \* x1 \*\* 2

weighted\_sum\_squares\_2 = weights[1] \* x2 \*\* 2

weighted\_sum\_squares\_3 = weights[2] \* x3 \*\* 2

# Вычисление sigma2 как экспоненты от взвешенной суммы квадратов факторов

sigma2\_1 = np.exp(weighted\_sum\_squares\_1)

sigma2\_2 = np.exp(weighted\_sum\_squares\_2)

sigma2\_3 = np.exp(weighted\_sum\_squares\_3)

e1 = np.random.normal(0, np.sqrt(sigma2\_1), size=n)

e2 = np.random.normal(0, np.sqrt(sigma2\_2), size=n)

e3 = np.random.normal(0, np.sqrt(sigma2\_3), size=n)

y1 = u1 + e1

y2 = u2 + e2

y3 = u3 + e3

# Построение точек для каждого фактора с использованием разных цветов и прозрачности

fig = plt.figure(figsize=(15, 8))

plt.scatter(x1, y1, alpha=0.6, label='x1')

plt.scatter(x2, y2, alpha=0.5, label='x2')

plt.scatter(x3, y3, alpha=0.4, label='x3')

plt.legend(loc='lower right', frameon=False, prop={'size': 16})

plt.xticks(fontsize=16)

plt.yticks(fontsize=16)

plt.show()

n=1200

m=8

# Определение параметров

theta = np.array([1, 2, 5, 4, 1.5, 2.5, 0.02, 0.01])

#---------- Задание №1 ----------#

# Генерация комбинаций факторов

x1, x2, x3 = generate\_random(n)

# Вычисление истинного отклика без шума

u = theta[0] + theta[1]\*x1 + theta[2]\*x2 + theta[3]\*x3 \

+ theta[4]\*x1\*x2 + theta[5]\*x1\*x3 \

+ theta[6]\*x1\*\*2 + theta[7]\*x2\*\*2

# Вычисление весов для каждого фактора

weights = np.array([0.8, 0.8, 0.9])

# Вычисление взвешенной суммы квадратов факторов

weighted\_sum\_squares = weights[0]\*x1\*\*2 + weights[1]\*x2\*\*2 + weights[2]\*x3\*\*2

# Вычисление sigma2 как экспоненты от взвешенной суммы квадратов факторов

sigma2 = np.exp(weighted\_sum\_squares)

e = np.random.normal(0, sigma2, size=n)

y = u + e

get\_dataframe\_for\_graphs(x1,x2,x3)

#---------- Задание №2.1. Тест Бреуша-Пагана ----------#

X = np.column\_stack((np.ones(len(x1)), x1, x2, x3, x1\*x2, x1\*x3, x1\*\*2, x2\*\*2))

# МНК-оценки

theta\_hat = np.linalg.inv(X.T @ X) @ X.T @ y

e\_t = y - X @ theta\_hat

sigma\_hat\_squared = np.sum(pow(e\_t, 2) / n)

print(sigma\_hat\_squared);

# Построение регрессии

c\_t = e\_t\*\*2 / sigma\_hat\_squared

z\_t = np.column\_stack((np.ones(len(x1)), np.exp(weights[0]\*x1\*\*2 + weights[1]\*x2\*\*2 + weights[2]\*x3\*\*2)))

alpha\_hat = np.linalg.inv(z\_t.T @ z\_t) @ z\_t.T @ c\_t

# Вычисление предсказанных значений

c\_hat = z\_t @ alpha\_hat

# Вычисление ESS

ESS = np.sum((c\_hat - np.mean(c\_t))\*\*2)

# Вычисление статистики теста Бреуша-Пагана

BP = ESS / 2

FT = f.ppf(1-0.05, 1, n)

if BP > FT:

print("{0} > {1} => гипотеза о гомоскедастичности не принимается".format(BP.round(5), FT.round(5)))

else:

print("{0} < {1} => Гипотеза о гомоскедастичности принимается".format(BP.round(5), FT.round(5)))

#---------- Задание №2.2. Тест Голдфельда-Квандтона ----------#

# Упорядочивание наблюдений по взвешенной сумме

order = np.argsort(weighted\_sum\_squares)

x1\_ordered = x1[order]

X\_ordered = X[order]

y\_ordered = y[order]

# Определение n\_c и разделение данных на две части

n\_c = n // 3

X1 = X\_ordered[:(n-n\_c)//2]

y1 = y\_ordered[:(n-n\_c)//2]

X2 = X\_ordered[(n+n\_c)//2:]

y2 = y\_ordered[(n+n\_c)//2:]

# Оценка параметров модели для каждой части данных

theta\_hat1 = np.linalg.inv(X1.T @ X1) @ X1.T @ y1

theta\_hat2 = np.linalg.inv(X2.T @ X2) @ X2.T @ y2

# Вычисление RSS для каждой части данных

RSS1 = np.sum((y1 - X1 @ theta\_hat1)\*\*2)

RSS2 = np.sum((y2 - X2 @ theta\_hat2)\*\*2)

# Вычисление статистики теста Голдфельда-Квандтона

GQ = RSS2 / RSS1

# Вычисление критического значения F-распределения

F\_crit = f.ppf(1-0.05, (n-n\_c-2\*m)/2, (n-n\_c-2\*m)/2)

if GQ > F\_crit:

print("{0} > {1} => гипотеза о гомоскедастичности не принимается".format(GQ.round(5), F\_crit.round(5)))

else:

print("{0} < {1} => Гипотеза о гомоскедастичности принимается".format(GQ.round(5), F\_crit.round(5)))

#---------- Задание №3 ----------#

# Вычисление весов

sigma\_hat\_squared = np.exp(z\_t @ alpha\_hat)

W = np.diag(sigma\_hat\_squared)

theta\_hat\_fgls = np.linalg.inv(X.T @ np.linalg.inv(W) @ X) @ X.T @ np.linalg.inv(W) @ y

print("Оценки параметров модели:", theta)

print("Оценки параметров модели по обыкновенному МНК:", theta\_hat.round(4))

print("Оценки параметров модели по обобщенному МНК:", theta\_hat\_fgls.round(4))

#---------- Задание №4 ----------#

# Вычисление квадрата расстояния между оценками параметров и истинными значениями

distance\_squared\_ls = np.sum((theta\_hat - theta) \*\* 2)

distance\_squared\_fgls = np.sum((theta\_hat\_fgls - theta) \*\* 2)

print("Квадрат расстояния для оценок МНК:", distance\_squared\_ls.round(5))

print("Квадрат расстояния для оценок обобщенного МНК:", distance\_squared\_fgls.round(5))