МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ «ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра алгоритмической математики

ОТЧЁТ
по практической работе №5
по дисциплине «Статистический анализ»
Тема: Элементы регрессионного анализа. Выборочные прямые среднеквадратической регрессии. Корреляционное отношение

Студент гр. 9372	____________________	Иванов Р. С.
Преподаватель	____________________	Сучков А. И.

Санкт-Петербург
2021

Цель работы

Ознакомление с основными положениями метода наименьших квадратов (МНК), со статистическими свойствами МНК оценок, с понятием функции регрессии и роли МНК в регрессионном анализе, с корреляционным отношением, как мерой тесноты произвольной (в том числе и линейной) корреляционной связи.

Основные теоретические положения

Алгоритм поиска линейной зависимости методом наименьших квадратов

Даны пары чисел (x_i, y_i), которые расположены вдоль прямой линии $y = kx + b$. Угловой коэффициент $\rho_{YX} = k$ называется *выборочным коэффициентом регрессии*. Найдем k и b с помощью метода наименьших квадратов. Введем функцию F:

$$F(k,b) = \sum_{i=1}^{k}(y_i - f(x_i,k,b))^2 \to \min \Leftrightarrow$$

$$\Leftrightarrow \begin{cases} \dfrac{\partial F}{\partial k} = \sum\limits_{i=1}^{k} 2x_i(kx_i + b - y_i) = 0 \\ \dfrac{\partial F}{\partial b} = \sum\limits_{i=1}^{k} 2(kx_i + b - y_i) = 0 \end{cases} \Leftrightarrow$$

$$\Leftrightarrow \begin{cases} k\overline{x^2} + b\overline{x} - \overline{xy} = 0 \\ k\overline{x} + b - \overline{y} = 0 \end{cases} \Rightarrow \boxed{\rho_{\mathrm{YX}} = k = \frac{\overline{xy} - \overline{x}\cdot\overline{y}}{\overline{x^2} - (\overline{x})^2} = \frac{\overline{xy} - \overline{x}\cdot\overline{y}}{\sigma_X^2}}$$

Таким образом, уравнение прямой линии регрессии Y на X выглядит

$$y_x - \overline{y} = \rho_{YX}\cdot(x - \overline{x})$$

X на Y:

$$x_y - \overline{x} = \rho_{XY}\cdot(y - \overline{yx})$$

Постановка задачи

Для заданной двумерной выборки (X,Y) построить уравнения выборочных прямых среднеквадратической регрессии. Полученные линейные функции регрессии отобразить графически. Найти выборочное корреляционное отношение. Полученные результаты содержательно проинтерпретировать.

Выполнение работы

1. Построим график выборки.

Рисунок 1 – График выборки

2. Построим уравнения среднеквадратичной регрессии.

$$\overline{x} = 17111.87$$
$$\overline{y} = 50225.33$$
$$\overline{xy} = 629742536.30$$
$$\sigma_X^2 = 11878.18$$
$$\sigma_Y^2 = 39568.61$$
$$\rho_{YX} = -1.6281$$
$$\rho_{XY} = -0.1467$$

- Y на X:

$$y_x = -1.6281 \cdot (x - 17111.87) + 50225.33$$

- X на Y:

$$x_y = -0.1467 \cdot (y - 50225.33) + 17111.87$$

Напомним, что наша выборка содержит данные о стоимости машин на вторичном рынке и их пробеге. Из графика видно, что можно разделить точки на 2 категории: те, кто лежат непосредственно около прямой, и те, кто лежат от нее далеко. Можно сделать вывод, что для большинства машин стоимость линейно зависит от пробега. Для второй категории людей можно предположить, что они относятся к специфичной категории, возможно это какие-то редкие модели или что-то в этом роде.

3. Составим корреляционную таблицу.

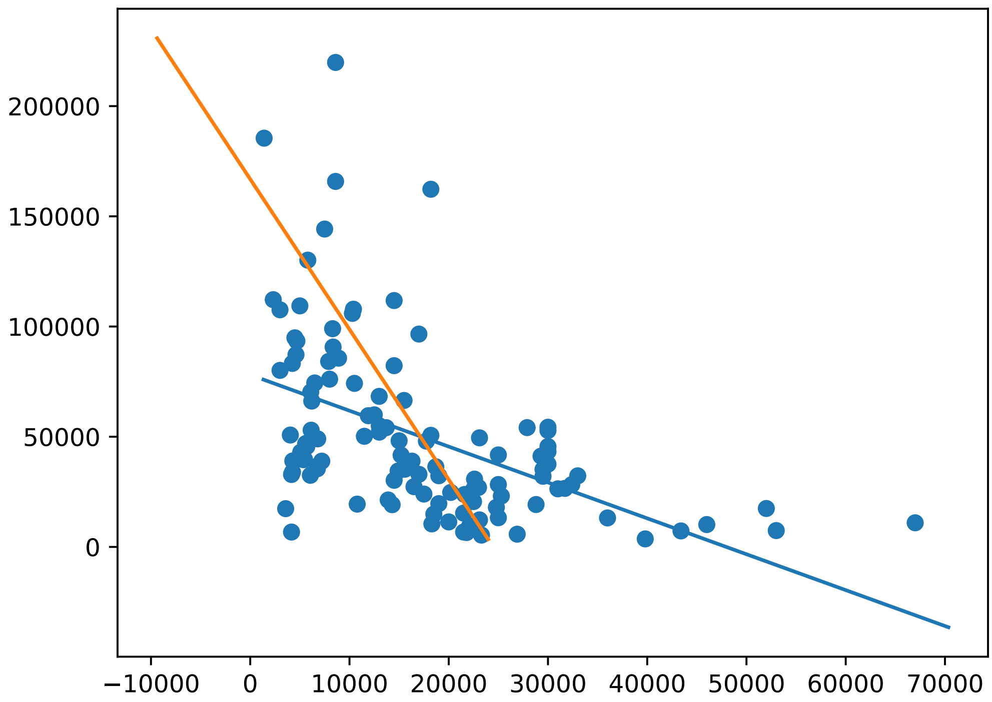

Рисунок 2 – Прямые среднеквадратичной регрессии

X	Y									m_{x_i}
	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	
x_1	2	3	16	7	2	2	2	0	1	35
x_2	16	9	8	10	1	0	0	0	0	44
x_3	5	7	0	2	0	0	0	0	0	14
x_4	6	6	1	0	0	0	0	0	0	13
x_5	3	1	0	0	0	0	0	0	0	4
x_6	1	0	0	0	0	0	0	0	0	1
x_7	1	1	1	0	0	0	0	0	0	3
x_8	0	0	0	0	0	0	0	0	0	0
x_9	0	1	0	0	0	0	0	0	0	1
m_{y_j}	34	28	26	19	3	2	2	0	1	115

Таблица 1 – Корреляционная таблица

Найдем межгрупповую дисперсию

$$\delta_X^2 = 6684$$

$$\delta_Y^2 = 21402$$

корреляционное отношение и выборочный коэффициент корреляции:

$$\eta_{XY} = 0.563$$
$$\eta_{YX} = 0.541$$
$$r_{XY} = -0.489$$

Как можно видеть, соотоношение $\eta \geqslant |r_{XY}|$ выполняется.

4. Найдем уравнение параболической зависимости:

$$y = 3.014 \cdot x^2 - 3.118 \cdot x + 90492$$

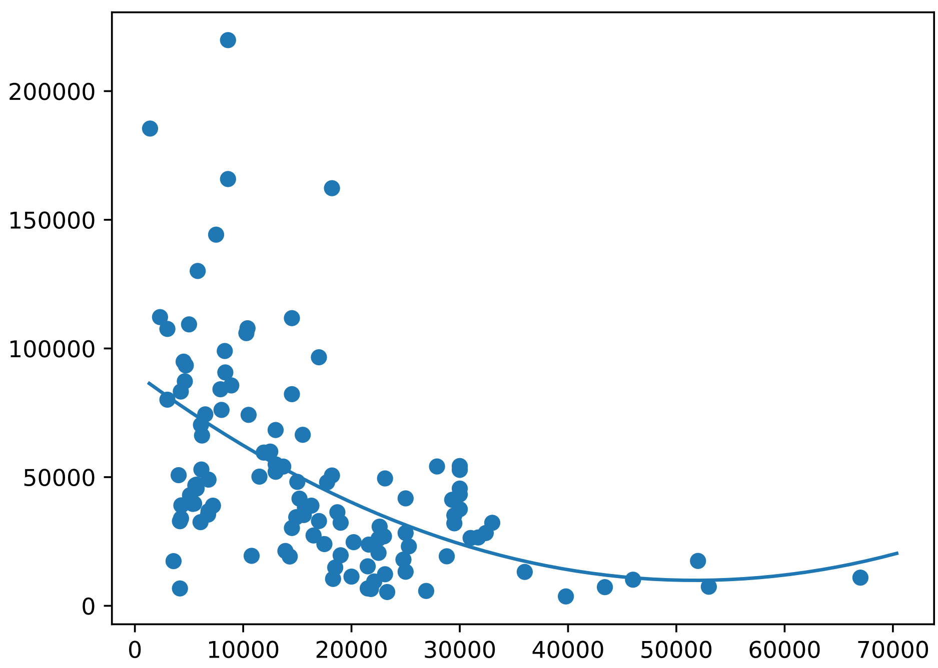

Рисунок 3 – Корреляционная кривая параболического вида

5. Построим линию корреляции вида $y = a \cdot e^{bx}$

$$y = 73866 \cdot e^{-4.04 \cdot x}$$

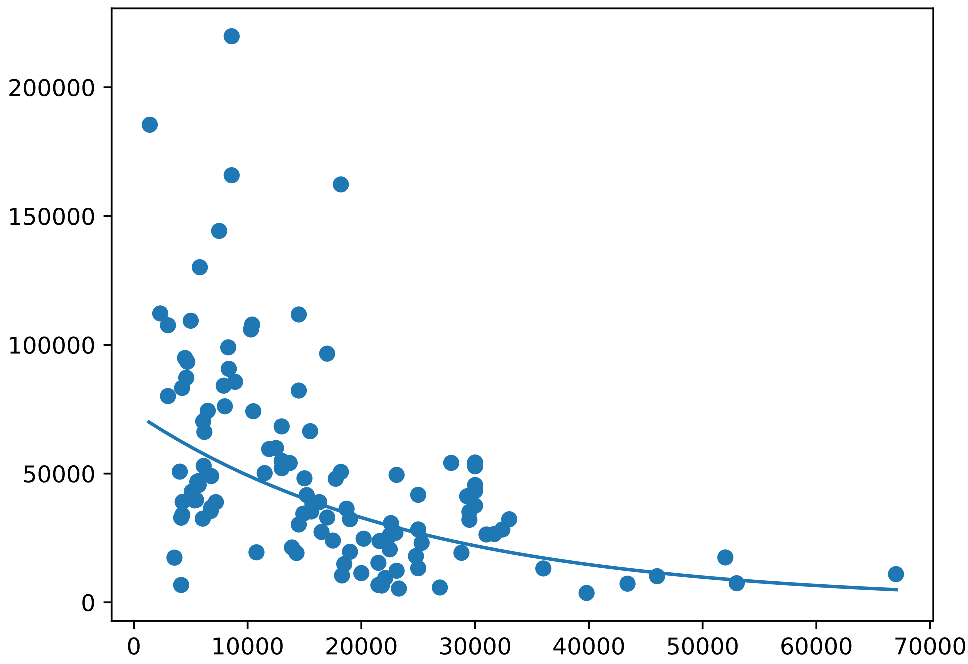

Рисунок 4 – Корреляционная кривая дробно-линейного вида

6. Вычислим показатели качества регрессии для каждой кривой:

- $y = kx + b$:
  - Теоретический коэффициент детерминации:

$$R^2 = 0.2389$$

  - Средняя квадратическая ошибка:

$$S_\varepsilon = 10454$$

  - Средняя ошибка аппроксимации:

$$A = 0.829$$

- $x = ky + b$:
  - Теоретический коэффициент детерминации:

$$R^2 = 0.2389$$

- Средняя квадратическая ошибка:

$$S_\varepsilon = 34825$$

- Средняя ошибка аппроксимации:

$$A = 0.949$$

- $y = ax^2 + bx + c$:
  - Теоретический коэффициент детерминации:

    $$R^2 = 0.2698$$

  - Средняя квадратическая ошибка:

    $$S_\varepsilon = 34261$$

  - Средняя ошибка аппроксимации:

    $$A = 0.841$$

- $y = \dfrac{1}{ax + b}$:
  - Теоретический коэффициент детерминации:

    $$R^2 = 0.215$$

  - Средняя квадратическая ошибка:

    $$S_\varepsilon = 35783$$

  - Средняя ошибка аппроксимации:

    $$A = 0.691$$

Выводы

Мы ознакомились с основными положениями метода наименьших квадратов, со статистическими свойствами МНК оценок, с понятием функции регрессии и роли МНК в регрессионном анализе, с корреляционным отношением, как мерой тесноты произвольной корреляционной связи. Научились строить кривые корреляции для произвольных функций, а также корреляционную таблицу.

ПРИЛОЖЕНИЕ А

```
import csv
import math
from math import sqrt, floor, log2
from itertools import islice
from collections import Counter
import matplotlib.pyplot as plt
import numpy as np

def read_dataset():
    with open('Price-Mileage.csv', 'r') as csv_file:
        reader = csv.reader(csv_file, delimiter=';')
        return [(int(row[0]), int(row[1])) for row in islice(reader, 1, None)]

# 1d sorted dataset
def make_interval_dataset(dataset):
    k = 1 + floor(log2(len(dataset)))
    h = (dataset[-1] - dataset[0]) / k

    def x(i):
        x0 = dataset[0] - h / 2
        return x0 + h * i
    j = 0
    counts = []
    xs = [(x(i), x(i + 1)) for i in range(k + 1)]
    for i in range(k + 1):
        count_i = 0
        while j < len(dataset) and dataset[j] <= x(i + 1):
            count_i += 1
            j += 1
```

```
        counts.append(count_i)
    frequencies = [c / len(dataset) for c in counts]
    return list(zip(xs, counts, frequencies))

def paragraph(i):
    print()
    print('-' * 15 + f' {i} ' + '-' * 15)

dataset = read_dataset()
#dataset = list(filter(lambda p: p[1] <= 18000,
   read_dataset()))
xs = list(map(lambda p: p[0], dataset))
ys = list(map(lambda p: p[1], dataset))

xs_int = make_interval_dataset(sorted(xs))
ys_int = make_interval_dataset(sorted(ys))

# 1
paragraph(1)

fig, ax = plt.subplots()
ax.scatter(xs, ys)
plt.savefig('scatter.pdf')

# 2
paragraph(2)

print('y = kx + b; x = ky + b')
x_avg = sum(xs) / len(xs)
y_avg = sum(ys) / len(ys)
xys = [xs[i] * ys[i] for i in range(len(xs))]
xy_avg = sum(xys) / len(xys)

```

```python
print(f'averages:\n\t x = {x_avg} y = {y_avg} xy = {xy_avg}')


def calc_sdev(a, avg):
    return sqrt(sum(map(lambda x: (x - avg) ** 2, a)) / len(a))

x_sdev = calc_sdev(xs, x_avg)
y_sdev = calc_sdev(ys, y_avg)

print(f'sdevs:\n\t x_sdev = {x_sdev} y_sdev = {y_sdev}')

rho_yx = (xy_avg - x_avg * y_avg) / x_sdev ** 2
rho_xy = (xy_avg - x_avg * y_avg) / y_sdev ** 2

print(f'pho:\n\t rho_yx = {rho_yx} rho_xy = {rho_xy}')

def y(x):
    return rho_yx * (x - x_avg) + y_avg

def x(y):
    return rho_xy * (y - y_avg) + x_avg

max_x = max(xs)
max_y = max(ys)
min_x = min(xs)
min_y = min(ys)

def make_y_plot():
    xs = np.linspace(0.95 * min_x, max_x * 1.05, 20)
    ys = np.linspace(0.95 * min_y, max_y * 1.05, 20)
    ax.plot(xs, y(xs))
```

```
    ax.plot(x(ys), ys)
    plt.savefig('regression_lines.pdf')

make_y_plot()


# parameters
paragraph('parameters')

R2 = rho_xy * rho_yx
print(f'R^2 = {R2}')

R2_y = sum(map(lambda i: (y(xs[i]) - y_avg) ** 2, range(len(xs)))) / (y_sdev ** 2
         * len(ys))
R2_x = sum(map(lambda i: (x(ys[i]) - x_avg) ** 2, range(len(xs)))) / (x_sdev ** 2
         * len(ys))
print(f'R^2_y = {R2_y} R^2_x = {R2_x}')
eps_y = list(map(lambda i: ys[i] - y(xs[i]), range(len(xs))))
eps_x = list(map(lambda i: xs[i] - x(ys[i]), range(len(xs))))
S_eps_y = sqrt(sum(map(lambda i: i ** 2, eps_y)) / (len(xs) - 2))
S_eps_x = sqrt(sum(map(lambda i: i ** 2, eps_x)) / (len(xs) - 2))
print(f'S_eps_y = {S_eps_y}, S_eps_x = {S_eps_x}')
Ay = sum(map(lambda i: abs(eps_y[i] / ys[i]), range(len(xs)))) / len(xs)
Ax = sum(map(lambda i: abs(eps_x[i] / xs[i]), range(len(xs)))) / len(xs)
print(f'Ax = {Ax} Ay = {Ay}')

```

```python
# ------------------- 3 ---------------------
paragraph(3)

def get_x_interval(x):
    for i in range(len(xs_int)):
        if xs_int[i][0][0] <= x <= xs_int[i][0][1]:
            return i
    raise f'incorrect x = {x}'

def get_y_interval(y):
    for i in range(len(ys_int)):
        if ys_int[i][0][0] <= y <= ys_int[i][0][1]:
            return i
    raise f'incorrect y = {y}'

count = Counter([(get_x_interval(x), get_y_interval(y)) for x, y in zip(xs, ys)])

corr_table = [[count[(i, j)] for j in range(len(ys_int))] for i in
        range(len(xs_int))]

m_y = [sum(map(lambda m: m[j], corr_table)) for j in range(len(ys_int))]
assert sum(m_y) == len(dataset)
print(f'm_y = {m_y}')

m_x = [sum(line) for line in corr_table]
assert sum(m_x) == len(dataset)
print(f'm_x = {m_x}')

def get_x(i):
    return (xs_int[i][0][1] + xs_int[i][0][0]) / 2

```

```
def get_y(j):
    return (ys_int[j][0][1] + ys_int[j][0][0]) / 2

def x_y_avg(j):
    s1 = 0
    for i in range(len(xs_int)):
        s1 += get_x(i) * corr_table[i][j]
    s2 = 0
    for i in range(len(xs_int)):
        s2 += corr_table[i][j]
    return s1 / s2

def y_x_avg(i):
    s1 = 0
    for j in range(len(ys_int)):
        s1 += get_y(j) * corr_table[i][j]
    s2 = 0
    for j in range(len(ys_int)):
        s2 += corr_table[i][j]
    return s1 / s2


new_x_avg = 0
for i in range(len(xs_int)):
    new_x_avg += get_x(i) * m_x[i]
new_x_avg /= len(dataset)

print(new_x_avg, x_avg)

delta_x = 0
for j in range(len(ys_int)):
    delta_x += (x_y_avg(j) - x_avg) ** 2 * m_y[j]
delta_x = sqrt(delta_x / len(dataset))

```

```
delta_y = 0
for i in range(len(xs_int)):
    delta_y += (y_x_avg(i) - y_avg) ** 2 * m_x[i]
delta_y = sqrt(delta_y / len(dataset))


print(f'delta_x = {delta_x} delta_y = {delta_y}')
nu_xy = delta_x / x_sdev
nu_yx = delta_y / y_sdev

print(f'nu_xy = {nu_xy} nu_yx = {nu_yx}')


r = (xy_avg - x_avg * y_avg) / (y_sdev * x_sdev)
print(f'r = {r}')

# -------------------- 4 -----------------------
paragraph(4)

def get_degree_sum(i):
    s = 0
    for x in xs:
        s += x ** i
    return float(s)

A = np.array([
    [get_degree_sum(4), get_degree_sum(3),
  get_degree_sum(2)],
    [get_degree_sum(3), get_degree_sum(2),
  get_degree_sum(1)],
    [get_degree_sum(2), get_degree_sum(1),
  get_degree_sum(0)]
])

```

```
x2y_sum = 0
for i in range(len(xs)):
    x2y_sum += xs[i] ** 2 * ys[i]

xy_sum = sum(xys)
y_sum = sum(ys)

B = np.array([
    float(x2y_sum),
    float(xy_sum),
    float(y_sum)
])

a, b, c = np.linalg.solve(A, B)
print('y = ax^2 + bx + c\n\t', f'a = {a}, b = {b}, c =
    {c}', sep='')

def y(x):
    return a * x ** 2 + b * x + c

def make_parab_plot():
    x = np.linspace(0.95 * min_x, max_x * 1.05, 50)
    fig, ax = plt.subplots()
    ax.scatter(xs, ys)
    ax.plot(x, y(x))
    plt.savefig('parab.pdf')

make_parab_plot()

#parameters
paragraph('parameters')

R2 = sum(map(lambda i: (y(xs[i]) - y_avg) ** 2, range(
    len(xs)))) / (y_sdev ** 2
```

```
          * len(ys))
print(f'R^2 = {R2}')
eps_y = list(map(lambda i: ys[i] - y(xs[i]), range(len
    (xs))))
S_eps_y = sqrt(sum(map(lambda i: i ** 2, eps_y)) / (
    len(xs) - 3))
print(f'S_eps_y = {S_eps_y}')
Ay = sum(map(lambda i: abs(eps_y[i] / ys[i]), range(
    len(xs)))) / len(xs)
print(f'Ay = {Ay}')

# -------- 5 --------
# var 3
# y = 1 / (ax + b)
paragraph(5)

xs_stroke = xs
ys_stroke = list(map(lambda y: math.log(y), ys))

x_stroke_avg = sum(xs_stroke) / len(xs_stroke)
y_stroke_avg = sum(ys_stroke) / len(ys_stroke)
xys_stroke = [xs_stroke[i] * ys_stroke[i] for i in
    range(len(xs_stroke))]
xy_stroke_avg = sum(xys_stroke) / len(xys_stroke)

print(f'averages:\n\t x = {x_avg} y = {y_avg} xy = {
    xy_avg}')


def calc_sdev(a, avg):
    return sqrt(sum(map(lambda x: (x - avg) ** 2, a))
    / len(a))

x_stroke_sdev = calc_sdev(xs_stroke, x_stroke_avg)
```

```
y_stroke_sdev = calc_sdev(ys_stroke, y_stroke_avg)

print(f'sdevs:\n\t x_sdev = {x_sdev} y_sdev = {y_sdev}
  ')

rho_yx = (xy_stroke_avg - x_stroke_avg * y_stroke_avg)
   / x_stroke_sdev ** 2
b = rho_yx
a = math.exp(y_stroke_avg - rho_yx * x_stroke_avg)
print(f'y = a*exp(bx):\n\t a = {a}, b = {b}')

def y(x):
    return a*np.exp(x * b)
    #return 1 / ( rho_yx * (x - x_stroke_avg) +
  y_stroke_avg)


max_x = max(xs)
max_y = max(ys)
min_x = min(xs)
min_y = min(ys)

def make_y_plot():
    fig, ax = plt.subplots()
    ax.scatter(xs, ys)
    # ax.set_yscale('log')
    x = np.linspace(0.95 * min_x, max_x, 50)
    ax.plot(x, y(x))
    plt.savefig('regression_lines2.pdf')

make_y_plot()

#params
paragraph('parameters')
```

```

R2 = sum(map(lambda i: (y(xs[i]) - y_avg) ** 2, range(len(xs)))) / (y_sdev ** 2
         * len(ys))
print(f'R^2 = {R2}')
eps_y = list(map(lambda i: ys[i] - y(xs[i]), range(len(xs))))
S_eps_y = sqrt(sum(map(lambda i: i ** 2, eps_y)) / (len(xs) - 2))
print(f'S_eps_y = {S_eps_y}')
Ay = sum(map(lambda i: abs(eps_y[i] / ys[i]), range(len(xs)))) / len(xs)
print(f'Ay = {Ay}')

# --------------- 6 ---------------


plt.show()
```

Листинг 1 – Исходный код программы