Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего образования

“Новосибирский государственный технический университет”

Кафедра теоретической и прикладной информатики

Лабораторная работа №3
по дисциплине “Статистические методы анализа данных”

Факультет:	ПМИ
Группа:	ПМи-51
Вариант	3
Студенты:	Неупокоев М.В. Фатыхов Т.М. Хахолин А.А.
Преподаватель:	Попов А.А.

Новосибирск
2018

Постановка задачи

1. Изменить модель регрессии, добавив в неё дополнительный регрессор, ранее не вошедший в состав модели, порождающей данные. Не генерируя новых данных, найти точечные оценки всех параметров расширенной модели. В дальнейшем при рассмотрении этой расширенной модели анализе должно быть показано, что параметр при дополнительном регрессоре незначим.
2. Построить доверительные интервалы для каждого параметра модели регрессии.
3. Проверить гипотезу о незначимости каждого параметра модели.
4. Проверить гипотезу о незначимости самой регрессии.
5. Рассчитать прогнозные значения для математического ожидания функции отклика
6. По полученным в п. 5 прогнозным значениям построить графики прогнозных значений и доверительной полосы для математического ожидания функции отклика и для самого отклика.
7. Заново смоделировать исходные данные (см. лаб. работу No 1), увеличив мощность случайной помехи до 50...70 % от мощности полезного сигнала и про- вести оценку параметров. Повторить пункты 3, 4 с новыми данными.

Анализ задачи

Формула для вычисления отклика

$$y_j = u_j + e_j = \eta(x_j, \theta) + e_j, j = 1,...,n$$

Определенная ранее имитационная модель $\eta(x_j, \theta)$

$$\eta(x_j, \theta) = \theta * f(x_j^1, x_j^2, x_j^3) = (\theta^1, \theta^2, \theta^3)^T * (f_1, f_2, f_3)$$

$$f(x) = (x_1, x_1^2, x_2, x_3, x_1 * x_2, x_1 * x_3)$$

$$x_i \in [-1,1], i = 1,k;$$

В данной лабораторной работе требуется изменить модель регрессии, добавив в нее дополнительный регрессор, новая модель будет иметь вид:

$$f(x) = (x_1, x_1^2, x_2, x_3, x_1 * x_2, x_1 * x_3, x_3^2 + 4)$$

Ход решения

1. Найдем точечные оценки всех параметров расширенной модели:

```
semi_X = np.apply_along_axis(semi_f, 1, factors)
X = np.apply_along_axis(model.f, 1, factors)

print(semi_X.shape)
```

```
(100, 7)
```

```
semi_theta_hat = get_theta_hat(semi_X, y)
print('theta_hat: \n',semi_theta_hat)
print()
print('theta : \n',model.theta)
```

```
theta_hat:
 [ 0.30097717  0.04799978 -0.41396271  0.05541637 -1.52333036  2.84773693
  0.00451161]

theta :
 [ 0.3     0.025  -0.45    0.0125 -1.55    2.8    ]
```

2. Построим доверительные интервалы для каждого параметра модели регрессии:

```
get_theta_intervals(semi_X, y, alpha=0.05)
print('real theta:', model.theta)
```

```
theta_hat_1
0.3010
[0.2448 ; 0.3571]

theta_hat_2
0.0480
[-0.0649 ; 0.1609]

theta_hat_3
-0.4140
[-0.4906 ; -0.3373]

theta_hat_4
0.0554
[-0.0114 ; 0.1223]

theta_hat_5
-1.5233
[-1.6448 ; -1.4019]

theta_hat_6
2.8477
[2.7359 ; 2.9596]

theta_hat_7
0.0045
[-0.0086 ; 0.0176]

real theta: [ 0.3     0.025  -0.45    0.0125 -1.55    2.8    ]
```

3. Проверим гипотезу о незначимости каждого параметра модели:

```
RSS = (n - m) * var_hat

for i in range(len(semi_theta_hat)):
    # change theta_H according to hypothesis that
    # theta_i = 0
    theta_hat_H = semi_theta_hat.copy()
    theta_hat_H[i] = 0

    # calculate error
    y_hat = semi_X.dot(theta_hat_H)
    e = y - y_hat

    # calculate RSSH for corresponding hypothesis
    RSSH = e.T.dot(e)

    q = semi_f([1,1,1]).shape[0]
    if linear_hypothesis_accepted(RSS, RSSH, q, n, m, 0.05):
        print('* Hypothesis #%d is accepted' % (i+1))
    else:
        print('  Hypothesis #%d is rejected' % (i+1))
    print()
```

```
  F: 16.943048    |       f: 0.305090
  Hypothesis #1 is rejected

  F: 0.206025     |       f: 0.305090
* Hypothesis #2 is accepted

  F: 19.294930    |       f: 0.305090
  Hypothesis #3 is rejected

  F: 0.351878     |       f: 0.305090
  Hypothesis #4 is rejected

  F: 104.055029 |       f: 0.305090
  Hypothesis #5 is rejected

  F: 408.261887 |       f: 0.305090
  Hypothesis #6 is rejected

  F: 0.111652     |       f: 0.305090
* Hypothesis #7 is accepted
```

4. Проверим гипотезу о незначимости самой регрессии.

```
q = m - 1

# take a look above
RSS = RSS

RSSH = np.square((y - y_hat.mean())).sum()


if linear_hypothesis_accepted(RSS, RSSH, q, n, m, 0.05):
    print('* Hypothesis is accepted')
else:
    print('  Hypothesis is rejected')
```

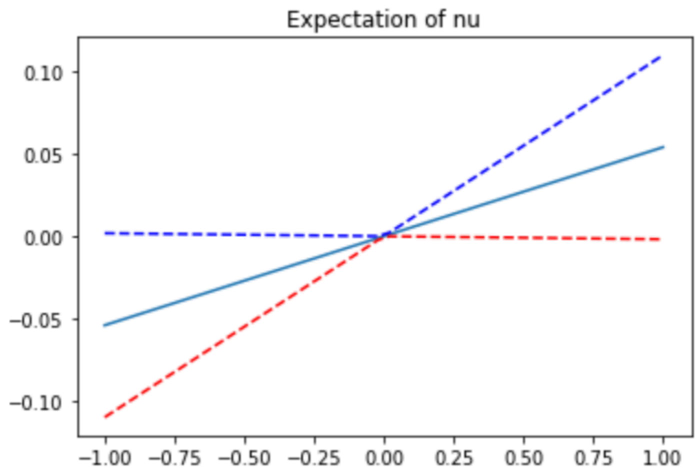
```
  F: 766.811804 |       f: 0.226880
  Hypothesis is rejected
```

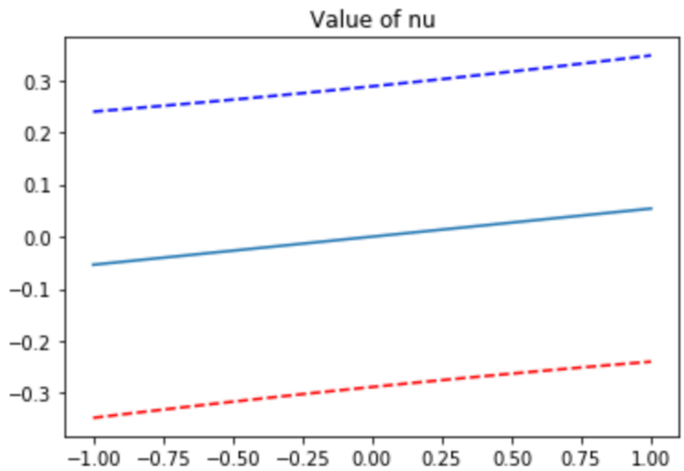
5. Рассчитаем прогнозные значения для математического ожидания функции отклика:

```
# beauty printing
print('x3 = % 7.4f | Exp(nu) = % 7.4f | [% 7.4f ;% 7.4f]' % (x3, nu, nu - epsilan, nu + epsilan))
```

```
x3 = -1.0000 | Exp(nu) = -0.0538 | [-0.1094 ; 0.0018]
x3 = -0.7778 | Exp(nu) = -0.0419 | [-0.0851 ; 0.0014]
x3 = -0.5556 | Exp(nu) = -0.0299 | [-0.0608 ; 0.0010]
x3 = -0.3333 | Exp(nu) = -0.0179 | [-0.0365 ; 0.0006]
x3 = -0.1111 | Exp(nu) = -0.0060 | [-0.0122 ; 0.0002]
x3 =  0.1111 | Exp(nu) =  0.0060 | [-0.0002 ; 0.0122]
x3 =  0.3333 | Exp(nu) =  0.0179 | [-0.0006 ; 0.0365]
x3 =  0.5556 | Exp(nu) =  0.0299 | [-0.0010 ; 0.0608]
x3 =  0.7778 | Exp(nu) =  0.0419 | [-0.0014 ; 0.0851]
x3 =  1.0000 | Exp(nu) =  0.0538 | [-0.0018 ; 0.1094]
```

6. По полученным в п. 5 прогнозным значениям построим графики прогнозных значений и доверительной полосы для математического ожидания функции отклика и для самого отклика:





7. Заново смоделируем исходные данные (см. лаб. работу No 1), увеличив мощность случайной помехи до 50...70 % от мощности полезного сигнала и проведем оценку параметров. Повторить пункты 3, 4 с новыми данными.

пункт 3

```
    if linear_hypothesis_accepted(RSS, RSSH, q, n, m, 0.05):
        print('* Hypothesis #%d is accepted' % (i+1))
    else:
        print('  Hypothesis #%d is rejected' % (i+1))
    print()
```

```
new_var_hat: 0.629081256089
  F: 1.584944    |       f: 0.269219
  Hypothesis #1 is rejected

  F: 0.274460    |       f: 0.269219
  Hypothesis #2 is rejected

  F: 3.330511    |       f: 0.269219
  Hypothesis #3 is rejected

  F: 0.086974    |       f: 0.269219
* Hypothesis #4 is accepted

  F: 3.565690    |       f: 0.269219
  Hypothesis #5 is rejected

  F: 22.677557   |       f: 0.269219
  Hypothesis #6 is rejected
```

пункт 4

```
q = m - 1

# take a look above
RSS = RSS

RSSH = np.square((y_new - y_hat.mean())).sum()

if linear_hypothesis_accepted(RSS, RSSH, q, n, m, 0.05):
    print('* Hypothesis is accepted')
else:
    print('  Hypothesis is rejected')
```

```
  F: 36.117762   |       f: 0.226880
  Hypothesis is rejected
```

Приложения

Текст программы:

```
import sys
sys.path.append('..')

from utils import model
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from scipy.stats import f as statF
from scipy.stats import t

get_ipython().magic('load_ext autoreload')
get_ipython().magic('autoreload 2')
df = pd.read_csv('../1st_lab/1st_lab_result.csv')
df.head(3)

factors = df[['x1', 'x2', 'x3']].values
y = df['y'].values
```

```
print(factors.shape, y.shape)

def semi_f(x):
    semi_result = model.f(x)
    semi_result = np.append(semi_result, x[2]**2 + 4)
    return semi_result

def get_theta_hat(X, y):
    """
    Outputs:
    - (X^T*X)^-1 * X^T * y
    """
    Xsq = X.T.dot(X)
    theta = np.dot(np.linalg.inv(Xsq), X.T)
    theta = np.dot(theta, y)
    return theta

semi_X = np.apply_along_axis(semi_f, 1, factors)
X = np.apply_along_axis(model.f, 1, factors)

print(semi_X.shape)

semi_theta_hat = get_theta_hat(semi_X, y)
print('theta_hat: \n',semi_theta_hat)
print()
print('theta : \n',model.theta)

# calculate semi error
y_hat = semi_X.dot(semi_theta_hat)
e = y - y_hat

# calculate variance estimation
n = len(semi_X)
m = len(semi_theta_hat)
semi_var_hat = e.T.dot(e) / (n - m)
print('var_hat:', semi_var_hat)

def get_theta_intervals(X, y, alpha, verbose=True):
    theta_hat = get_theta_hat(X, y)
    n = X.shape[0]
    m = X.shape[1]
    almost_sigma = semi_var_hat * np.linalg.inv(X.T.dot(X))

    for i in range(len(theta_hat)):
        offset = t.ppf(alpha/2, n - m) * np.sqrt(almost_sigma[i, i])
        if verbose:
            print('theta_hat_%d' % (i+1))
            print('%.4f' % (theta_hat[i]))
            print('[%.4f ; %.4f]' % (theta_hat[i] + offset, theta_hat[i] - offset))
            print()

get_theta_intervals(semi_X, y, alpha=0.05)
print('real theta:', model.theta)
theta_hat = get_theta_hat(X, y)

# calculate error
y_hat = X.dot(theta_hat)
e = y - y_hat

# calculate real variance estimation
n = len(X)
m = len(theta_hat)
var_hat = e.T.dot(e) / (n - m)
print('var_hat:', var_hat)

def linear_hypothesis_accepted(RSS, RSSH, q, n, m, alpha, verbose=True):
    """
    Outputs:
    - Boolean; true if hypothesis is accepted, false otherwise
    """
    F = (RSSH - RSS) * (n - m) / (q * RSS)
    threshold = statF.ppf(alpha, q, n - m)
    if verbose:
        print('  F: %.6f\t|\tf: %.6f' % (F, threshold))
    return F < threshold

RSS = (n - m) * var_hat

for i in range(len(semi_theta_hat)):
    # change theta_H according to hypothesis that
```

```
    # theta_i = 0
    theta_hat_H = semi_theta_hat.copy()
    theta_hat_H[i] = 0

    # calculate error
    y_hat = semi_X.dot(theta_hat_H)
    e = y - y_hat

    # calculate RSSH for corresponding hypothesis
    RSSH = e.T.dot(e)

    q = semi_f([1,1,1]).shape[0]
    if linear_hypothesis_accepted(RSS, RSSH, q, n, m, 0.05):
        print('* Hypothesis #%d is accepted' % (i+1))
    else:
        print('  Hypothesis #%d is rejected' % (i+1))
    print()

q = m - 1

# take a look above
RSS = RSS

RSSH = np.square((y - y_hat.mean())).sum()

if linear_hypothesis_accepted(RSS, RSSH, q, n, m, 0.05):
    print('* Hypothesis is accepted')
else:
    print('  Hypothesis is rejected')

theta_hat = get_theta_hat(X, y)
alpha = 0.05

# calculate error
y_hat = X.dot(theta_hat)
e = y - y_hat

# calculate variance estimation
n = len(X)
m = len(theta_hat)
var_hat = e.T.dot(e) / (n - m)

# we have only 3 factors! fix two first factors as 0,
# changing the remaining
x1, x2 = 0, 0
for i in np.linspace(-1, 1, 10):
    x3 = i
    _x = [x1, x2, x3]

    nu = model.nu(_x, theta=theta_hat)
    Xsq_inv = np.linalg.inv(X.T.dot(X))
    f = model.f(_x)
    sigma = f.T.dot(Xsq_inv).dot(f)
    sigma *= var_hat
    sigma = np.sqrt(sigma)
    epsilan = -t.ppf(alpha, n - m) * sigma

    # beauty printing
    print('x3 = % 7.4f | Exp(nu) = % 7.4f | [% 7.4f ;% 7.4f]' % (x3, nu, nu - epsilan, nu +
epsilan))

# we have only 3 factors! fix two first factors as 0,
# changing the remaining
x1, x2 = 0, 0
e_nu_values, e_epsilan_values = np.array([]), np.array([])
nu_values, epsilan_values = np.array([]), np.array([])
x3_space = np.linspace(-1, 1, 50)
step_width = x3_space[1] - x3_space[0]

for i in x3_space:
    x3 = i
    _x = [x1, x2, x3]

    # expectation of nu
    e_nu = model.nu(_x, theta=theta_hat)
    Xsq_inv = np.linalg.inv(X.T.dot(X))
    f = model.f(_x)
    sigma = f.T.dot(Xsq_inv).dot(f)
    sigma *= var_hat
    sigma = np.sqrt(sigma)
    e_epsilan = -t.ppf(alpha, n - m) * sigma
```

```
    e_nu_values = np.append(e_nu_values, e_nu)
    e_epsilan_values = np.append(e_epsilan_values, e_epsilan)

    # nu
    nu = model.nu(_x, theta=theta_hat)
    sigma = f.T.dot(Xsq_inv).dot(f) + 1
    sigma *= var_hat
    sigma = np.sqrt(sigma)
    epsilan = -t.ppf(alpha, n - m) * sigma

    nu_values = np.append(nu_values, nu)
    epsilan_values = np.append(epsilan_values, epsilan)


plt.title('Expectation of nu')
plt.plot(x3_space, e_nu_values)
plt.plot(x3_space, e_nu_values - e_epsilan_values, 'r--')
plt.plot(x3_space, e_nu_values + e_epsilan_values, 'b--')
plt.show()


plt.title('Value of nu')
plt.plot(x3_space, nu_values)
plt.plot(x3_space, nu_values - epsilan_values, 'r--')
plt.plot(x3_space, nu_values + epsilan_values, 'b--')
plt.show()

# show plot for expectation of nu-values
plt.ylim(-0.2, 0.2)
plt.bar(x3_space, height=e_nu_values,
            width=step_width, align='edge',
            color='blue', label='expectation of nu')
for i, eps in enumerate(e_epsilan_values):
    l = e_nu_values[i] - eps
    h = e_nu_values[i] + eps
    x = [x3_space[i], x3_space[i] + step_width]
    plt.plot(x, [l, l], 'r')
    plt.plot(x, [h, h], 'r')
    plt.plot([x[0], x[0]], [l, h], 'r--')
    plt.plot([x[1], x[1]], [l, h], 'r--')
plt.title('Expectation of nu')
plt.legend()
plt.show()

# show plot for expectation of nu-values
plt.bar(x3_space, height=nu_values,
            width=step_width, align='edge',
            color='blue', label='nu values')
for i, eps in enumerate(epsilan_values):
    l = e_nu_values[i] - eps
    h = e_nu_values[i] + eps
    x = [x3_space[i], x3_space[i] + step_width]
    plt.plot(x, [l, l], 'r')
    plt.plot(x, [h, h], 'r')
    plt.plot([x[0], x[0]], [l, h], 'r--')
    plt.plot([x[1], x[1]], [l, h], 'r--')
plt.title('Values of nu')
plt.legend()
plt.show()

u = np.apply_along_axis(model.nu, 1, factors)
u_hat = u.mean()
u_centered = u - u_hat

w = (u_centered).dot(u_centered) / (len(u) - 1)
print('w = %f' % w )

p = 0.60

sigma = p * w
print('sigma = %f' % sigma)

new_e = np.random.randn(len(u)) * sigma

print('error mean = %f' % new_e.mean())
print('error var  = %f' % new_e.std())

y_new = u + new_e

# get new theta
```

```
new_theta_hat = get_theta_hat(X, y_new)

# calculate new error
y_hat = X.dot(new_theta_hat)
e = y_new - y_hat

# calculate new variance estimation
n = len(u)
m = len(new_theta_hat)
new_var_hat = e.T.dot(e) / (n - m)
print('new_var_hat:', new_var_hat)

RSS = (n - m) * new_var_hat

for i in range(len(new_theta_hat)):
    # change theta_H according to hypothesis that
    # theta_i = 0
    theta_hat_H = new_theta_hat.copy()
    theta_hat_H[i] = 0

    # calculate error
    y_hat = X.dot(theta_hat_H)
    e = y_new - y_hat

    # calculate RSSH for corresponding hypothesis
    RSSH = e.T.dot(e)

    q = model.f([1,1,1]).shape[0]
    if linear_hypothesis_accepted(RSS, RSSH, q, n, m, 0.05):
        print('* Hypothesis #%d is accepted' % (i+1))
    else:
        print('  Hypothesis #%d is rejected' % (i+1))
    print()

q = m - 1

# take a look above
RSS = RSS

RSSH = np.square((y_new - y_hat.mean())).sum()

if linear_hypothesis_accepted(RSS, RSSH, q, n, m, 0.05):
    print('* Hypothesis is accepted')
else:
    print('  Hypothesis is rejected')
```