Теория вероятностей и математическая статистика¶

Урок 7¶
Линейная регрессия. Двухвыборочный t-тест. A/B-тестирование¶
Разбор домашнего задания¶

Задача 1

Дана матрица объект-признак

```
X = [[ 1.22401313,  2.30868478,  3.03636353,  2.69287214],
     [-0.18757272,  1.30337355,  5.12093014,  3.46363202],
     [-0.81094525,  1.82463398,  5.79686488,  1.86159445],
     [ 0.75129018,  2.67392052,  3.65529809,  1.66746094],
     [ 0.00972362,  1.97367255,  2.50594319,  1.69755173],
     [-0.62972637,  0.77750764,  2.84124027,  4.54410559],
     [ 2.29536229,  1.81206697,  1.95026215,  1.51874636],
     [ 0.0920418 ,  2.26971361,  7.47708735,  2.61081203],
     [ 2.39252799,  3.17563985,  3.61420599,  5.10773362],
     [ 0.54983815,  2.87988651,  1.65752765,  1.59635987]]
```

и значения целевой переменной

```
y = [ 9.26193358,  9.700363  ,  8.67214805,  8.74796974,  6.18689108,
      7.53312713,  7.57643777, 12.44965478, 14.29010746,  6.68361218]
```

1. Подберите два признака (из четырёх) так, чтобы уровень линейной зависимости целевой переменной от значений этих признаков был максимальным. Другими словами, модель линейной регрессии на этих признаках должна давать наилучший результат.
2. Является ли значимым получившееся уравнение регрессии?

Решение

In [1]:

```python
import numpy as np
```

In [2]:

```python
X = [[ 1.22401313,  2.30868478,  3.03636353,  2.69287214],
     [-0.18757272,  1.30337355,  5.12093014,  3.46363202],
     [-0.81094525,  1.82463398,  5.79686488,  1.86159445],
     [ 0.75129018,  2.67392052,  3.65529809,  1.66746094],
     [ 0.00972362,  1.97367255,  2.50594319,  1.69755173],
     [-0.62972637,  0.77750764,  2.84124027,  4.54410559],
     [ 2.29536229,  1.81206697,  1.95026215,  1.51874636],
     [ 0.0920418 ,  2.26971361,  7.47708735,  2.61081203],
     [ 2.39252799,  3.17563985,  3.61420599,  5.10773362],
     [ 0.54983815,  2.87988651,  1.65752765,  1.59635987]]
X = np.array(X)

y = [ 9.26193358,  9.700363  ,  8.67214805,  8.74796974,  6.18689108,
      7.53312713,  7.57643777, 12.44965478, 14.29010746,  6.68361218]
y = np.array(y)

X.shape, y.shape
```

Out[2]:

```
((10, 4), (10,))
```

Итак, нашу задачу можно решить, построив для каждой пары признаков модель линейной регрессии на этих признаках, и посчитав её коэффициент детерминации.

Коэффициенты линейной регрессии можно посчитать по формуле: $$b = \left(X^\top X \right)^{-1} X^\top y$$

Здесь X — «расширенная» матрица, т.е. первый столбец этой матрицы соответствует значению 1 при коэффициенте b_0.

После этого для вычисления коэффициента детерминации нам понадобятся предсказанные моделью значения: $$z = X \cdot b$$

По ним вычисляется массив ошибок модели: $$E = y - z$$

Теперь коэффициент детерминации равен: $$R^2 = 1 - \dfrac{SS_{res}^2}{SS_y^2}$$

Обернём всё это в единую функцию:

In [3]:

```python
def sum_of_squares(samples: np.ndarray) -> float:
    """Сумма квадратов отклонений.
    """

    return ((samples - samples.mean()) ** 2).sum()
```

In [4]:

```python
def get_determination_coefficient(x: np.ndarray, y: np.ndarray) -> float:
    """Эта функция строит по матрице объект-признак `x` и массиву значений
    целевой переменной `y` модель линейной регрессии и возвращает её
    коэффициент детерминации.
    """

    ones = np.ones((x.shape[0], 1))
    x = np.hstack([ones, x])

    xtx = x.T.dot(x)
    xtx_inv = np.linalg.inv(xtx)
```

```
    b = xtx_inv.dot(x.T).dot(y)

    z = x.dot(b)
    E = y - z

    return 1 - sum_of_squares(E) / sum_of_squares(y)
```

Итак, переберём все пары признаков и посчитаем для каждой такой пары коэффициент детерминации соответствующей модели.

In [5]:

```
from itertools import combinations
```

In [6]:

```
for i, j in combinations(range(X.shape[1]), 2):
    r = get_determination_coefficient(X[:, [i, j]], y)

    print(f'{i} {j} {r}')
```

```
0 1 0.18113594742585215
0 2 0.7634246238793152
0 3 0.4532966783144077
1 2 0.547948273403901
1 3 0.6062055761129932
2 3 0.622441987650532
```

Видим, что использование первого и третьего признаков даёт наилучшую зависимость. Проверим статистическую значимость этой зависимости. Это можно сделать с помощью распределения Фишера. Используем статистику: $$F = \dfrac{R^2 / m}{(1 - R^2) / (n - m - 1)},$$ где m — число факторов модели. В нашем случае $m = 2$. Такая статистика имеет распределение Фишера с параметрами $k_1 = m$, $k_2 = n - m - 1$.

In [7]:

```
k1 = 2
k2 = X.shape[0] - k1 - 1

R = get_determination_coefficient(X[:, [0, 2]], y)
F = (R / k1) / ((1 - R) / k2)

F
```

Out[7]:

```
11.29443912292265
```

Для уровня значимости $\alpha = 0.05$ найдём нужный квантиль (напомним, что критическая область у распределения Фишера правосторонняя):

In [8]:

```
from scipy import stats
```

In [9]:

```
alpha = 0.05

t = stats.f.ppf(1 - alpha, k1, k2)
t
```

Out[9]:

```
4.73741412777588
```

Итак, критическая область имеет вид: $$\Omega_\alpha = \left(4.737, \infty \right)$$

Значение статистики попало в критическую область, значит, уравнение регрессии признаётся статистически значимым.

Задача 2

Для проведения A/B-тестирования сайта интернет-магазина были получены следующие данные: страница A была посещена 2509 раз, из них 77 закончились совершением покупки, страница B была посещена 1465 раз, 60 из них закончились совершением покупки. Является ли значимым отличие конверсии на страницах A и B?

Подсказка. Реализуйте двухвыборочный t-тест. В качестве выборок здесь можно взять наборы меток совершения покупки (0 или 1) каждым посетителем.

Решение

Итак, запишем выборки:

In [10]:

```
x1 = np.zeros(2509)
x1[np.arange(77)] = 1

x2 = np.zeros(1465)
x2[np.arange(60)] = 1
```

Реализуем двухвыборочный t-тест.

In [11]:

```
n1 = x1.size
n2 = x2.size

s1 = x1.std(ddof=1)
s2 = x2.std(ddof=1)
```

Посчитаем σ_Δ: $$\sigma_\Delta = \sqrt{\dfrac{\sigma_{X_1}^2}{n_1} + \dfrac{\sigma_{X_2}^2}{n_2}}$$

In [12]:

```
s_delta = np.sqrt(s1 ** 2 / n1 + s2 ** 2 / n2)
s_delta
```

Out[12]:

```
0.006220171278295827
```

Значение статистики: $$t = \dfrac{\overline{X_1} - \overline{X_2}}{\sigma_\Delta}$$

In [13]:

```
t = (x1.mean() - x2.mean()) / s_delta
t
```

Out[13]:

```
-1.6504551408398205
```

Посчитаем число степеней свободы: $$df = \dfrac{\left(\dfrac{\sigma_{X_1}^2}{n_1} + \dfrac{\sigma_{X_2}^2}{n_2} \right)^2}{\dfrac{\left(\sigma_{X_1}^2 / n_1 \right)^2}{n_1 - 1} + \dfrac{\left(\sigma_{X_2}^2 / n_2 \right)^2}{n_2 - 1}}$$

In [14]:

```
df = (s1 ** 2 / n1 + s2 ** 2 / n2) ** 2 / ((s1 ** 2 / n1) ** 2 / (n1 - 1) +
(s2 ** 2 / n2) ** 2 / (n2 - 1))
df
```

Out[14]:

```
2732.8025644352133
```

Возьмём уровень значимости $\alpha = 0.05$. Посчитаем квантили распределения Стьюдента:

In [15]:

```
alpha = 0.05

t1 = stats.t.ppf(alpha / 2, df=df)
t2 = stats.t.ppf(1 - alpha / 2, df=df)

t1, t2
```

Out[15]:

```
(-1.9608324352746576, 1.9608324352746571)
```

Итак, критическая область выглядит следующим образом: $$\Omega_\alpha = (-\infty, -1.96) \cup (1.96, \infty)$$

Значение статистики в эту область не попало, поэтому заключаем, что между конверсией на страницах A и B значимого отличия нет.

Задача 3

Квартет Энскомба — популярный в области анализа данных пример наборов данных, у которых практически совпадают все статистические свойства (средние, дисперсии, коэффициенты корреляции, регрессионные линии), однако, существенно отличаются графики. Данный пример призван показать, насколько важна визуализация данных. Датасет представляет собой 4 пары выборок:

```
{
    "x1": [10.0, 8.0, 13.0, 9.0, 11.0, 14.0, 6.0, 4.0, 12.0, 7.0, 5.0],
    "y1": [8.04, 6.95, 7.58, 8.81, 8.33, 9.96, 7.24, 4.26, 10.84, 4.82, 5.68],
    "x2": [10.0, 8.0, 13.0, 9.0, 11.0, 14.0, 6.0, 4.0, 12.0, 7.0, 5.0],
    "y2": [9.14, 8.14, 8.74, 8.77, 9.26, 8.1, 6.13, 3.1, 9.13, 7.26, 4.74],
    "x3": [10.0, 8.0, 13.0, 9.0, 11.0, 14.0, 6.0, 4.0, 12.0, 7.0, 5.0],
    "y3": [7.46, 6.77, 12.74, 7.11, 7.81, 8.84, 6.08, 5.39, 8.15, 6.42, 5.73],
    "x4": [8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 19.0, 8.0, 8.0, 8.0],
    "y4": [6.58, 5.76, 7.71, 8.84, 8.47, 7.04, 5.25, 12.5, 5.56, 7.91, 6.89]
}
```

По каждой паре выборок посчитайте:

1. выборочное среднее и дисперсию каждой выборки,
2. коэффициент корреляции Пирсона и прямую линейной регрессии.

Убедившись в том, что они практически не отличаются, постройте scatter plot по каждой паре выборок.

Решение

Загрузим датасет:

In [16]:

```
data = {
    "x1": [10.0, 8.0, 13.0, 9.0, 11.0, 14.0, 6.0, 4.0, 12.0, 7.0, 5.0],
    "y1": [8.04, 6.95, 7.58, 8.81, 8.33, 9.96, 7.24, 4.26, 10.84, 4.82,
5.68],
    "x2": [10.0, 8.0, 13.0, 9.0, 11.0, 14.0, 6.0, 4.0, 12.0, 7.0, 5.0],
    "y2": [9.14, 8.14, 8.74, 8.77, 9.26, 8.1, 6.13, 3.1, 9.13, 7.26, 4.74],
    "x3": [10.0, 8.0, 13.0, 9.0, 11.0, 14.0, 6.0, 4.0, 12.0, 7.0, 5.0],
    "y3": [7.46, 6.77, 12.74, 7.11, 7.81, 8.84, 6.08, 5.39, 8.15, 6.42,
5.73],
```

```
    "x4": [8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 19.0, 8.0, 8.0, 8.0],
    "y4": [6.58, 5.76, 7.71, 8.84, 8.47, 7.04, 5.25, 12.5, 5.56, 7.91,
6.89]
}
```

In [17]:

```
for i in range(1, 5):
    x = data[f'x{i}']
    y = data[f'y{i}']

    print(f'Выборки {i}')
    print(f'Выборочные средние: {np.mean(x)}, {np.mean(y)}')
    print(f'Выборочные дисперсии: {np.var(x)}, {np.var(y)}')
    print(f'Коэффициент корреляции: {np.corrcoef(x, y)[0, 1]}\n')
```

```
Выборки 1
Выборочные средние: 9.0, 7.500909090909093
Выборочные дисперсии: 10.0, 3.7520628099173554
Коэффициент корреляции: 0.81642051634484

Выборки 2
Выборочные средние: 9.0, 7.50090909090909
Выборочные дисперсии: 10.0, 3.752390082644628
Коэффициент корреляции: 0.8162365060002428

Выборки 3
Выборочные средние: 9.0, 7.5
Выборочные дисперсии: 10.0, 3.747836363636364
Коэффициент корреляции: 0.8162867394895984

Выборки 4
Выборочные средние: 9.0, 7.500909090909091
Выборочные дисперсии: 10.0, 3.7484082644628103
Коэффициент корреляции: 0.8165214368885028
```

Для вычисления коэффициентов парной регрессии воспользуемся формулами: $$b_1 = \frac{\sigma_{XY}}{\sigma^2_X}, \:\: b_0 = \overline{Y} - b_1 \cdot {\overline{X}}$$

Здесь же построим scatter plot по каждой паре выборок.

In [18]:

```
from matplotlib import pyplot as plt

plt.style.use('seaborn-whitegrid')
%config InlineBackend.figure_formats = ['svg']
```

In [19]:

```
fig, axes = plt.subplots(nrows=2, ncols=2)
fig.set_size_inches(10, 6)

ox = np.linspace(3, 20, 10 ** 4)

for i, ax in enumerate(axes.flatten()):
    i += 1

    x = data[f'x{i}']
    y = data[f'y{i}']

    b1 = np.cov(x, y, ddof=0)[0, 1] / np.var(x, ddof=0)
    b0 = np.mean(y) - b1 * np.mean(x)

    print(f'Выборки {i}: b0 = {round(b0, 4)},\tb1 = {round(b1, 4)}')
```

```
        oy = b0 + b1 * ox

        ax.scatter(x, y)
        ax.plot(ox, oy, color='red', alpha=0.5)

        ax.set_xlim(3, 20)
        ax.set_ylim(2, 14)

        ax.set_title(f'Выборки {i}')

    print()
```

```
Выборки 1: b0 = 3.0001, b1 = 0.5001
Выборки 2: b0 = 3.0009, b1 = 0.5
Выборки 3: b0 = 3.0025, b1 = 0.4997
Выборки 4: b0 = 3.0017, b1 = 0.4999
```

2021-01-13T16:32:13.157735 image/svg+xml Matplotlib v3.3.3, https://matplotlib.org/

In [ ]: