Корреляционный анализ

Биометрия.
Проверить: **есть ли зависимость** между возрастом (в месяцах) и весом самок павианов гамадрилов.
Если зависимость есть, то выяснить: **линейна она или нет**.

```
import numpy as np
import pandas as pd
import seaborn as sns

from pylab import rcParams
sns.set()
rcParams['figure.figsize'] = 10, 5
%matplotlib inline
%config InlineBackend.figure_format = 'svg'
import matplotlib.pyplot as plt
```

Данные

```
X = np.array([33, 31, 31, 32, 34, 26, 29.8,
              31, 32.5, 32.5, 24, 28.5, 26, 28.5,
              32, 31.5, 32.5, 34, 33.5, 33.5])

Y = np.array([7.5, 5.7, 5.4, 5.8, 6.8,
              6.2, 8, 6.1, 6.8, 5.6, 5, 5,
              5.4, 6.7, 5.3, 5.5, 6.4,
              6.3, 5.5, 6.0])

data = pd.DataFrame({'Возраст X (мес)':X,
                     'Вес Y (кг)': Y}, index = np.arange(1,len(X)+1))
data
```

	Возраст X (мес)	Вес Y (кг)
1	33.0	7.5
2	31.0	5.7
3	31.0	5.4
4	32.0	5.8
5	34.0	6.8
6	26.0	6.2
7	29.8	8.0
8	31.0	6.1
9	32.5	6.8
10	32.5	5.6
11	24.0	5.0
12	28.5	5.0
13	26.0	5.4
14	28.5	6.7
15	32.0	5.3
16	31.5	5.5

```
17              32.5          6.4
18              34.0          6.3
19              33.5          5.5
20              33.5          6.0
```

```
print(f'Размах возраста X: {X.max() - X.min()}')
print(f'Размах веса Y: {Y.max() - Y.min()}')
```

```
Размах возраста X: 10.0
Размах веса Y: 3.0
```

Коэффициент корреляции

Сперва найдем по известным выборкам коэффциент корреляции, являющийся мерой линейной зависимости

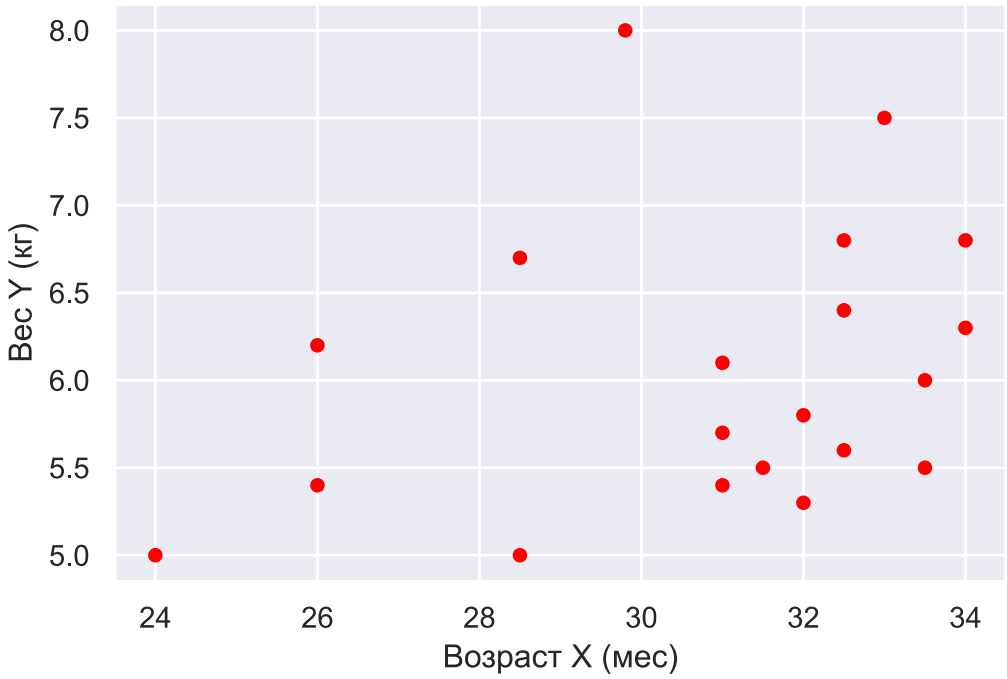
```
covariance = np.mean(X * Y) - np.mean(X) * np.mean(Y)
correlation =  round(covariance / (np.std(X) * np.std(Y)), 3)
print(f'Коэффициент корреляции равен {correlation}')
```

```
Коэффициент корреляции равен 0.288
```

Видим, что коэффициент корреляции больше нуля. Это значит, что большим значениям Y соответствуют большие значения X. Рассмотрим на графике

```
data.plot.scatter(x = 'Возраст X (мес)', y = 'Вес Y (кг)',
color='red')
```

```
<AxesSubplot:xlabel='Возраст X (мес)', ylabel='Вес Y (кг)'>
```



Построение корреляционной таблицы и нахождение характеристик

```
# Создадим пустую корреляционную таблицу
dict_for_correlation_table = {}
for i in np.unique(X):
    dict_for_correlation_table[i] = np.zeros(len(np.unique(Y)))
correlation_table = pd.DataFrame(dict_for_correlation_table,
index=np.unique(sorted(Y)))
correlation_table
```

	24.0	26.0	28.5	29.8	31.0	31.5	32.0	32.5	33.0	33.5	34.0
5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

```
# заполним корреляционную таблицу на основе имеющихся данных
for row in data.itertuples(index=False):
    correlation_table[row[0]][row[1]] += 1
correlation_table
```

	24.0	26.0	28.5	29.8	31.0	31.5	32.0	32.5	33.0	33.5	34.0
5.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5.3	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
5.4	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
5.5	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0
5.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
5.7	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
5.8	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
6.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
6.1	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
6.2	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
6.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
6.7	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0
7.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
8.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

```
# сделаем проверку
correlation_table.sum().sum() == len(X)
```

```
True
```

```
# найдем характеристики
y_points = correlation_table.index.values
x_points = correlation_table.columns.values
```

```
#--------------------Условные
средние-----------------------------------------------
mean_y_i = []
for column in x_points:
    if np.sum(correlation_table[column].values) == 0:
        mean_y_i.append(0.0000000000000001)
        continue
    value = np.sum(correlation_table[column].values * y_points) /
(correlation_table[column].values.sum())
    mean_y_i.append(round(value, 3))
mean_y_i = np.array(mean_y_i)
print('Условные средние y при X = x_i: ', mean_y_i)
```

```
#--------------------Полное
среднее-----------------------------------------------------------
N = (correlation_table.sum().sum()) # всего элементов (длина
изначальной выборки)
mean_y = np.sum(correlation_table.sum(1).values * y_points) / N
mean_y = round(mean_y, 3)
print('Полное среднее: ', mean_y)
```

```
#-----------------Межгрупповая
дисперсия---------------------------------------------------------
variance_between_groups = np.sum(correlation_table.sum(0).values *
(mean_y_i - mean_y)**2) / N
variance_between_groups = round(variance_between_groups, 3)
print('Межгрупповая дисперсия: ', variance_between_groups)
```

```
#-----------------Полная
дисперсия---------------------------------------------------------------
variance =  np.sum(correlation_table.sum(1).values * (y_points -
mean_y)**2) / N
```

```
variance = round(variance, 3)
print('Полная дисперсия: ', variance)
```

```
#----------------Корреляционное
отношение-----------------------------------------------------------------
relation_correlation = np.sqrt(variance_between_groups / variance)
relation_correlation = round(relation_correlation, 3)
print('Корреляционное отношение:', relation_correlation)
```

```
Условные средние y при X = x_i:  [5.    5.8   5.85  8.    5.733 5.5
5.55  6.267 7.5   5.75  6.55 ]
Полное среднее:  6.05
Межгрупповая дисперсия:  0.457
Полная дисперсия:  0.613
Корреляционное отношение: 0.863
```

Гипотеза о значимости корреляционного отношения

Выдвигаем гипотезу:
H_0: $\eta_{yx}=0$ (между Y и X нет зависимости)
Задаем уровень значимости: 0.05
Для проверки используем критерий Стьюдента

```
from scipy.stats import t
t_current = round(relation_correlation * np.sqrt((N-2) / (1-
relation_correlation**2)), 3)
t_theory = round(t.isf(q=0.05, df=N-2), 3)
print(f't критическое = {t_theory}, t наблюдаемое = {t_current}')
```

```
t критическое = 1.734, t наблюдаемое = 7.247
```

Поскольку наблюдаемое значение критерия превышает критическое, то с заданным уровнем значимости 0.05 нулевую гипотезу **отклоняют**.

Гипотеза о линейной связи между X и Y

Зависимость между рассматриваемыми величинами есть (H_0 опровергнута). Встает вопрос: линейная ли это зависимость? Рассмотрим ещё одну статистику - **оценку расхождения корреляционного отношения и коэффициента корреляции**.

```
gamma = relation_correlation**2 - correlation**2
print('Расхождение между коэффициентом корреляции и корреляционным
отношением равно ', gamma)
```

```
std_gamma = round(2 * np.sqrt(gamma - (2-relation_correlation**2-
correlation**2)*gamma**2) / np.sqrt(N), 3)
print('Среднее квадратичное расхождения равно ', std_gamma)
```

```
Расхождение между коэффициентом корреляции и корреляционным отношением равно  0.661825
Среднее квадратичное расхождения равно  0.172
```

Выдвигаем гипотезу:
H_0: связь между X и Y линейна, то есть $\gamma = 0$
Уровень значимости: 0.05
Проверим с помощью критерия Стьюдента

```
t_current = round(gamma / std_gamma, 3)
print(f't критическое = {t_theory}, t наблюдаемое = {t_current}')
```

```
t критическое = 1.734, t наблюдаемое = 3.848
```

Наблюдаемое значения критерия превышает критическое значение, значит с заданным уровнем значимости 0.05 гипотеза о линейной связи между X и Y **отклоняется**.