

```
# Вариант 8
# Переменные: Iris versicolor, Длина чашелистика, Длина лепестка

# Открываем базу данных
df = pd.read_csv("iris.data", delimiter=',')
# Выбираем данные только с versicolor
df = df[df['class'].isin(['Iris-versicolor'])]

# Выбираем данные длины чашелистика и лепестка только с versicolor
length1 = df['length of petal']
length2 = df['length of sepal']
```

3.1. Проверить гипотезу о независимости переменных по критерию Хи-

квадрат

КОД ФУНКЦИИ:

```
def task1(len1, len2):
    # Разбиваем функцией из библиотеки pandas на 4 интервала
    # Теперь в len1 len2 хранятся интервалы в которых находятся длины
    len1 = pd.qcut(len1, 4)
    len2 = pd.qcut(len2, 4)

    # Создаем таблицу из двух столбцов длин
    df = pd.concat([len1, len2], axis=1)

    # Создаем таблицу с количеством пересекающихся интервалов
    table = pd.crosstab(df['length of petal'], df['length of sepal'])

    # Сумма по горизонтали
    table['vi_sum'] = table.sum(axis=1)

    # Сумма по вертикали
    vj_sum = pd.Series(name='vj_sum')
    table = table.append(vj_sum, ignore_index=False)
    table.iloc[4] = table.sum(axis=0)

    print("3.1\nВыведем таблицу сопряженности двух признаков")
    print(table)

    # Статистика критерия независимости по формуле из расчетника
    statistic_sum = 0
    for i in range(4):
        for j in range(4):
            statistic_sum += np.power(table.iloc[j][i], 2) /
            (table.iloc[4][i] * table.iloc[j][4])
    statistic_practic = (statistic_sum - 1) * 50

    # Табличное значение
    statistic_theory = 16.9

    if statistic_theory > statistic_practic:
        print('\nПеременные независимы, т.к', statistic_practic, '<',
        statistic_theory)
    else:
        print('\nПеременные зависимы, т.к', statistic_practic, '>=',
        statistic_theory)
```

Вывод:

3.1

Выведем таблицу сопряженности двух признаков

	(4.899, 5.6]	(5.6, 5.9]	(5.9, 6.3]	(6.3, 7.0]	vi_sum
length_of_petal					
(2.999, 4.0]	11.0	3.0	2.0	0.0	16.0
(4.0, 4.35]	2.0	5.0	1.0	1.0	9.0
(4.35, 4.6]	3.0	1.0	5.0	5.0	14.0
(4.6, 5.1]	0.0	1.0	5.0	5.0	11.0
vj_sum	16.0	10.0	13.0	11.0	50.0

Переменные зависимы, т.к $30.494498793504476 \geq 16.9$

3.2. Вычислить оценку ковариации, коэффициента корреляции

Проверить гипотезу о незначимости коэффициента корреляции

КОД:

```
def task2(width_petal, width_sepal):
    m_len = width_petal.mean()
    m_width = width_sepal.mean()
    alpha = 0.05

    xy = np.sum(np.multiply(width_petal, width_sepal)) # сумма произведений
    xy_m = xy / 50 # среднее от произведения переменных

    s_len = np.sum(np.power(np.subtract(width_petal, m_len), 2)) # сумма
    дисперсий лепестка
    s_len_m = s_len / 50

    s_width = np.sum(np.power(np.subtract(width_sepal, m_width), 2)) # сумма
    дисперсий чашелистника
    s_width_m = s_width / 50

    std_dev_len = np.sqrt(s_len_m)
    std_dev_width = np.sqrt(s_width_m)

    c = xy_m - m_len * m_width
    r = c / (std_dev_len * std_dev_width)

    print("3.2\nКоэффициент корреляции:", r)
    print("Коэффициент ковариации:", c)

    T = (r / np.sqrt(1 - r ** 2)) * (np.sqrt(50 - 2))
    tss = s_width

    # T табличное
    T_table = 1.96

    if T > T_table:
        print("Гипотеза о незначимости принимается\n\n")
```

Вывод:

3.2

Коэффициент корреляции: 0.7540489585920163

Коэффициент ковариации: 0.17924000000000007

Гипотеза о незначимости принимается

3.3. Оценить параметры линейной регрессии, вычислить

коэффициент детерминации, проверить значимость модели по критерию Фишера.

```
x = length1.values.reshape(-1, 1)
y = length2.values.reshape(-1, 1)
reg = LinearRegression()
reg.fit(x, y)
plt.scatter(length1, length2)
plt.plot(length1, reg.predict(x), color='red', linewidth=2)
plt.show()
print("Уравнение линейной регрессии: Y = {:.5} + {:.5}X".format(reg.intercept_[0], reg.coef_[0][0]))

m_len = length2.mean()
m_width = length1.mean()
xy = np.sum(np.multiply(length2, length1)) # сумма произведений
xy_m = xy / 50 # среднее от произведения переменных

tss = np.sum(np.power(np.subtract(length1, m_width), 2)) # сумма дисперсий

sq_len = np.sum(np.power(length2, 2))
beta1_lid = (xy_m - m_len * m_width) / ((sq_len / 50) - (m_len ** 2))
beta0_lid = m_width - beta1_lid * m_len

rss = 0
ess = 0
for i in range(50, 100):
    y_lid = beta0_lid + beta1_lid * length2[i]
    rss += (length1[i] - y_lid) ** 2
    ess += (y_lid - m_width) ** 2

print("tss", tss)
print("rss", rss)
print("ess", ess)
determ_ko = 1 - rss / tss
print("Коэффициент детерминации по формуле 1 - rss/tss", determ_ko)

determ_ko2 = ess / tss
print("Коэффициент детерминации по формуле ess/tss", determ_ko2)
determ_ko3 = r * r
print("Коэффициент детерминации по формуле r*r", determ_ko3)

f = (determ_ko3 / (1 - determ_ko3)) * ((50 - 1 - 1) / 1)
f_table = 4.03
if f_table > f:
    print("регрессия считается незначимой, т.к", f_table, '>', f)
else:
    print("регрессия считается значимой, т.к", f_table, '<=', f)
```

Вывод:

```
Уравнение линейной регрессии:  $Y = 2.4075 + 0.82828X$   
tss 10.820000000000002  
rss 4.667858018260922  
ess 6.1521419817389305  
Коэффициент детерминации по формуле  $1 - \text{rss}/\text{tss}$  0.5685898319537042  
Коэффициент детерминации по формуле  $\text{ess}/\text{tss}$  0.5685898319536903  
Коэффициент детерминации по формуле  $r*r$  0  
регрессия считается незначимой, т.к  $4.03 > 0.0$ 
```

