Часть 2

Задания:

2.1. Вычислить выборочные характеристики (2 балла):

• выборочное среднее,

• выборочную дисперсию,

• несмещенную выборочную дисперсию,

• минимальную порядковую статистику,

• максимальную порядковую статистику,

• размах,

• медиану.

2.2. Построить (3 балла):

• график эмпирической функции распределения,

• гистограмму,

• ядерную оценку функции плотности.

2.3. Построить 99% - доверительный интервал (в предположении, что

выборка подчиняется нормальному распределению с неизвестными

параметрами) (3 балла)

• для математического ожидания

• для дисперсии

2.4. Проверить гипотезу о нормальном законе распределения (2 балла)

• по критерию Колмогорова или Хи-квадрат Пирсона

Код:

```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import math
import scipy.stats as sps
import warnings



def open_file(x):
    with open('iris.data') as f:
        strings = f.readlines()
        for raw in strings:
            if raw == "\n":
                strings.remove(raw)
                continue
            x.append(raw.split(","))
    return x


def sample_average(x):
    return np.mean(x)


def sample_variance(x, x_avg):

    return sum((x - x_avg) ** 2) / (len(x))


def unbiased_sample_variance(x, x_avg):
    return sum((x - x_avg) ** 2) / (len(x) - 1)


def minmax_ordinal_statistics(x):
    x.sort()
    return x[0], x[-1]


def range_(x_min, x_max):
    return x_max - x_min


def median(x):
    if len(x) % 2 == 0:
        mediana = x[int((len(x) + 1) / 2)]
    else:
        mediana = (x[int(len(x) / 2)] + x[int((len(x) / 2) + 1)]) / 2
    return mediana


def empirical_cdf(x):
    plt.title("График эмпирической функции распределения")
    sns.ecdfplot(x)
    plt.show()


def histogram(x):
    plt.title("Гистограмма относительных частот: ")
    plt.hist(x, histtype='bar', bins=7)
    plt.show()
```

```
def kernel_density_estimation(x):
    plt.title("Ядерная оценка функции плотности: ")
    plt.xlabel('Длина лепестка')
    sns.distplot(x, hist=True)
    plt.show()


def confidence_intervals(x, x_avg, v):
    t = 2.57
    interval =  t * v**0.5 /(len(x))**0.5
    print("99% доверительный интервал для мат. ожидания: [", x_avg -
interval, ";", x_avg + interval, "]")

    epsilon = 0.01
    q1 = sps.chi2.ppf(epsilon / 2, len(x))
    q2 = sps.chi2.ppf(1 - epsilon / 2, len(x))
    int1 = (len(x) * v) / q2
    int2 = (len(x) * v) / q1
    print("99% доверительный интервал для дисперсии: [", int1, ";", int2,
"]")


def is_normal_distribution(x, x_avg, var):
    def Dn():
        d_plus = 0
        for i, x_1 in enumerate(x):
            d_plus = max(d_plus, (i + 1) / len(x) - sps.norm(loc=x_avg,
scale=var).cdf(x_1))

        d_minus = 0
        for i, x_1 in enumerate(x):
            d_minus = max(d_minus, sps.norm(loc=x_avg, scale=var).cdf(x_1) -
i / len(x))
        print(d_minus, d_plus)
        return max(d_minus, d_plus)

    def Sk():
        return (6 * len(x) * Dn() + 1) / (6 * math.sqrt(len(x)))

    if Sk() <= 0.9042:
        return print("Да, т.к",Sk(),"<=" ,"0.9042" )
    else:
        return print("нет, т.к",Sk(),">" ,"0.9042" )


def task2(setosa):
    avg = sample_average(setosa)
    var = sample_variance(setosa, avg)
    var1 = unbiased_sample_variance(setosa, avg)
    min_, max_ = minmax_ordinal_statistics(setosa)
    rang = range_(min_, max_)
    med = median(setosa)

    print('2.1 Вычислить выборочные характеристики')
    print("Выборочное среднее: ", avg)
    print("Выборочная дисперсия: ", var)
    print("Несмещенная выборочная дисперсия: ", var1)
    print("Порядковые статистики: ", "минимальная - ", min_, "максимальная -
", max_)
    print("Размах: ", rang)
    print("Медиана: ", med)
    print('2.2 Построить графики')
    empirical_cdf(setosa)
```

```
    histogram(setosa)
    kernel_density_estimation(setosa)
    print('2.3 Построить 99% - доверительный интервал (в предположении, что
выборка подчиняется нормальному распределению с неизвестными параметрами)')
    confidence_intervals(setosa, avg, var)
    print('2.4 Проверить гипотезу о нормальном законе распределения • по
критерию Колмогорова или Хи-квадрат Пирсона')
    print("Является ли нормальным распределение для 0.05?" )
    is_normal_distribution(setosa, avg, math.sqrt(var))


def main():
    warnings.filterwarnings('ignore')
    lines = []
    open_file(lines)

    setosa_petal_length = []

    for i in lines:
        if i[4] == 'Iris-setosa\n':
            setosa_petal_length.append(float(i[3]))


    setosa = setosa_petal_length.copy()
    task2(setosa)
```

Конец кода

Вывод:

2.1

Вычислить выборочные характеристики

Выборочное среднее: 0.244

Выборочная дисперсия: 0.011264000000000001

Несмещенная выборочная дисперсия: 0.011493877551020 41

Порядковые статистики: минимальная - 0.1 максимальная - 0.6

Размах: 0.5

Медиана: 0.2

2.2

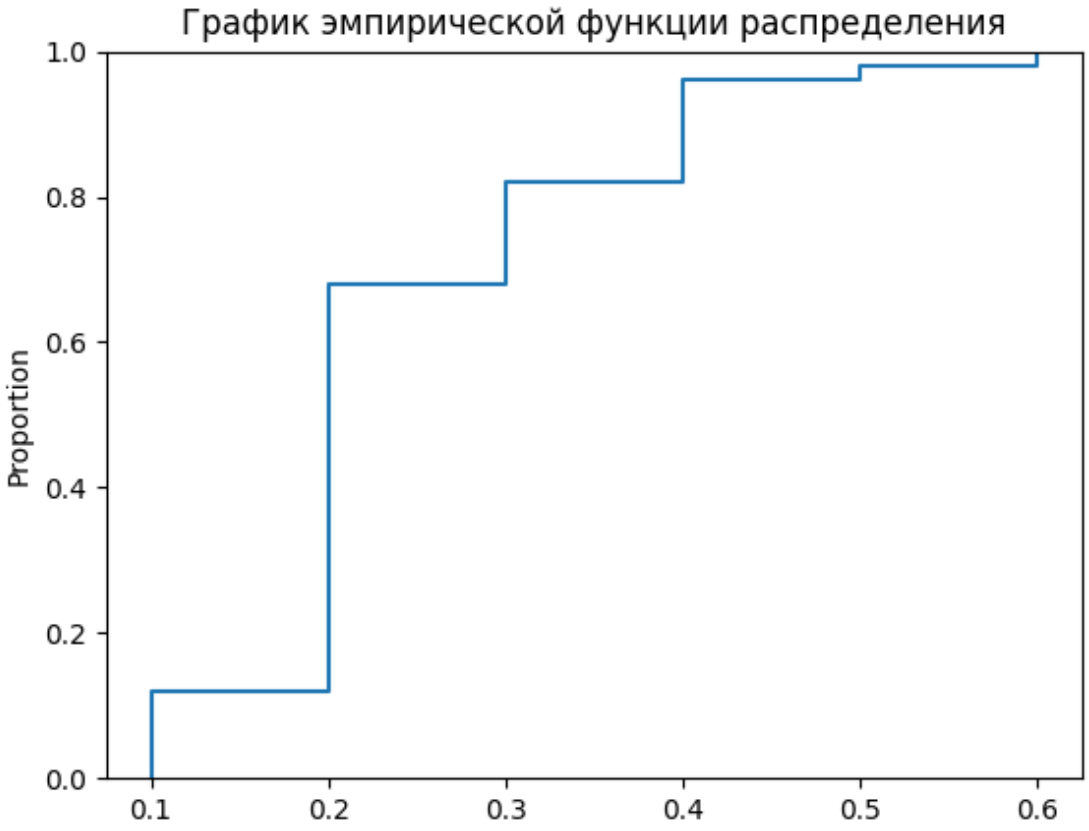
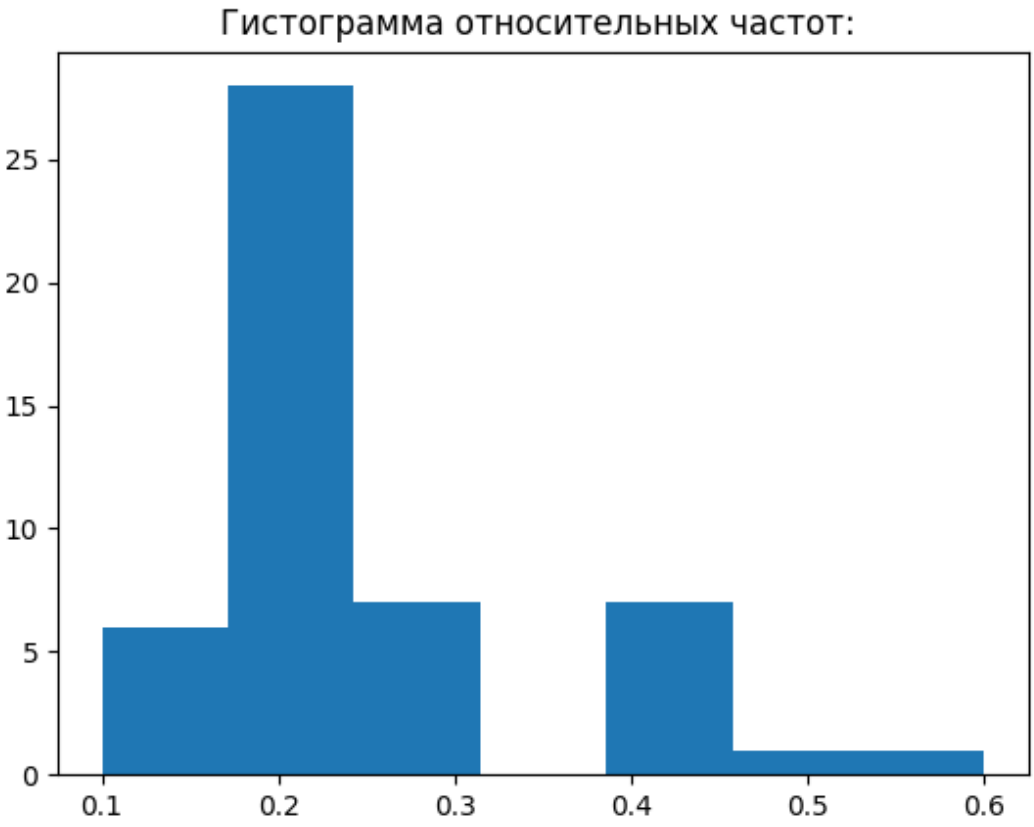
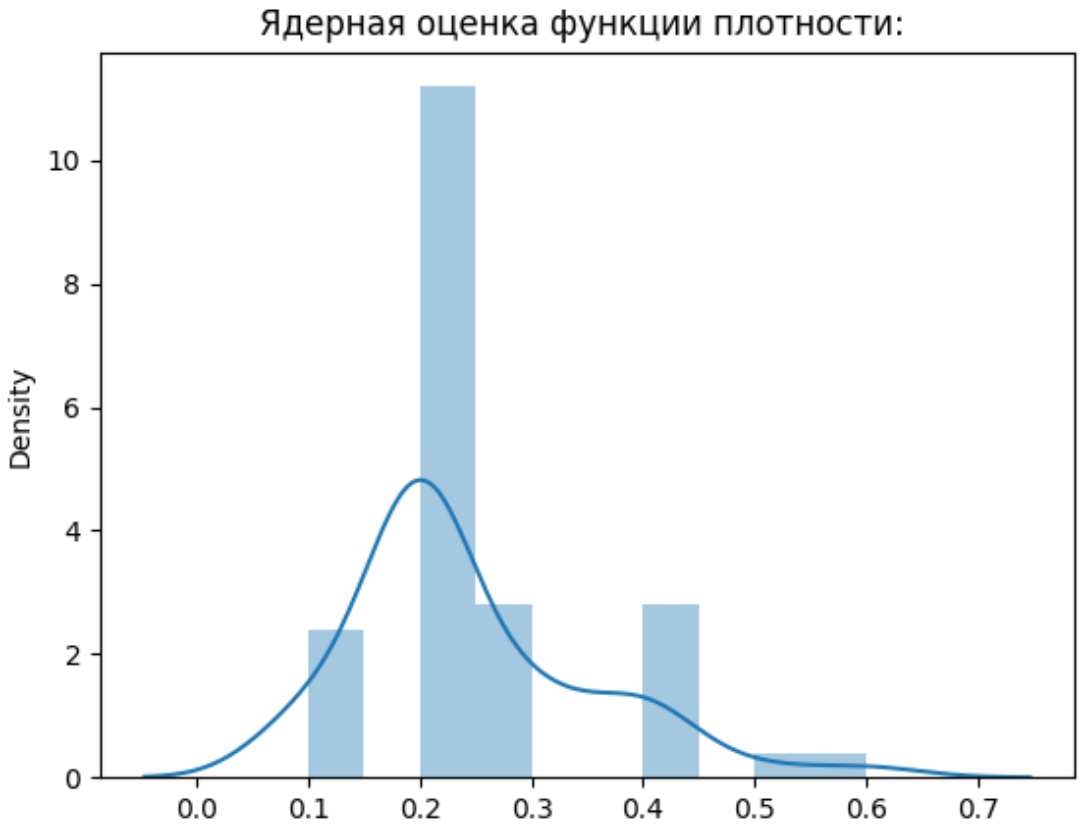


График эмпирической функции распределения
Proportion
1.0
0.8
0.6
0.4
0.2
0.0
0.1
0.2
0.3
0.4
0.5
0.6


Гистограмма относительных частот:
25
20
15
10
5
0
0.1
0.2
0.3
0.4
0.5
0.6



2.3

Построить 99% - доверительный интервал (в предположении, что выборка подчиняется нормальному распределению с неизвестными параметрами)

99% доверительный интервал для мат. ожидания:
[0.2054260207912121; 0.2825739792087879]

99% доверительный интервал для дисперсии:
[0.0070851698649663 46; 0.020120933623058595]

2.4

Проверить гипотезу о нормальном законе распределения • по критерию Колмогорова или Хи-квадрат Пирсона

Является ли нормальным распределение для 0.05?

0.21922539186887757 0.3407746081311225

0.21922539186887757 0.3407746081311225

нет, т.к. 2.4332105886966024> 0.9042