МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Інститут прикладної математики та фундаментальних наук

Кафедра прикладної математики

Звіт

Про виконання лабораторної роботи № 3

з курсу “Математична Статистика”

Виконав:

Студент групи ПМ-33

Маркевич Леонід

Перевірив:

Янішевський В.С.

Львів-2023

Лабораторна робота №3
Варіант 12

Завдання:

- 1) побудувати гістограму накопичених відносних частот;
- 2) апроксимувати гістограму (пункту 1) функцією нормального закону розподілу з параметрами випадкової величини (формула1);
- 3) зробити висновки про узгодження теоретичного й статистичного законів розподілів;
- 4) знайти інтервальні оцінки для математичного сподівання та дисперсії випадкової величини
- 5) перевірити за критерієм узгодження Пірсона відповідність теоретичного розподілу емпіричним даним.

$$x_i = (\sum_{j=1}^{12} r_j - 6)\,\sigma + a$$

де a = 12 -10= 2, σ = 3+ 8/12 = 3.(66);

Хід роботи:
Реалізація виконана на мові програмування Python з використанням бібліотек pandas, numpy, math, matplotlib, scipy

1) побудувати гістограму накопичених відносних частот;

Реалізація формули програмно:

```
class SampleGenerator:
    def __init__(self, v, n, sum_limit, *args, **kwargs):
        self.v = v
        self.n = n
        self.a = self.v - 10
        self.sigma = 3 + self.v / 10
        self.sum_limit = sum_limit


    def generation_x(self):
        r = sum(random.uniform(0, 1) for _ in range(self.sum_limit))
```

```
        return (r - 6) * self.sigma + self.a

    def data_collector(self):
        X = [self.generation_x() for _ in range(self.n)]
        return X



def task_1(data):
    values, base = np.histogram(data, bins=40)
    cumulative = np.cumsum(values)
    relative_frequency = cumulative / cumulative[-1]

    plt.figure(figsize=(10, 6))

    plt.plot(base[:-1], relative_frequency, c='red', label='Накопичені відносні частоти')

    plt.title('Гістограма накопичених відносних частот')
    plt.xlabel('Значення')
    plt.ylabel('Частота')
    plt.legend(loc='upper left')

    plt.show()

sample_generator = SampleGenerator(v=8, sum_limit=12, n=350)

data = sample_generator.data_collector()

task_1(data)
```

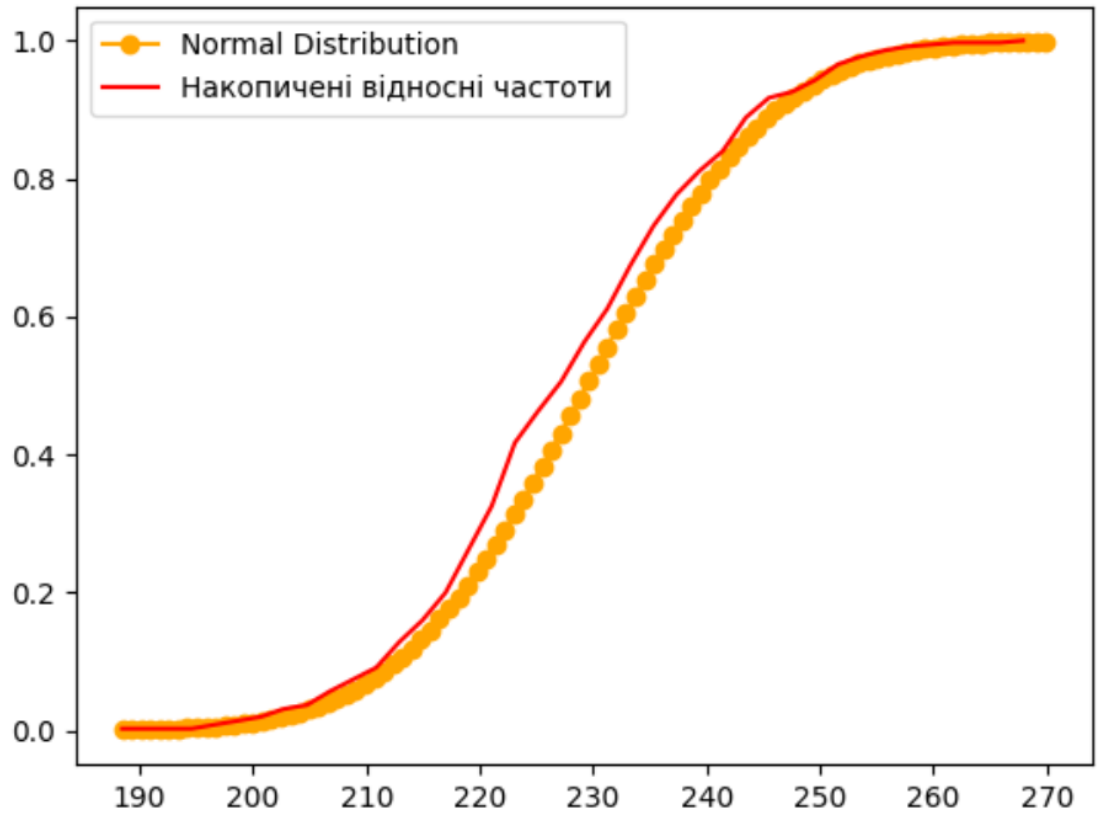
Приклад роботи:

2) апроксимувати гістограму (пункту 1) функцією нормального закону розподілу з параметрами випадкової величини (формула1);

Програмна реалізація:

```
def task_2(data):
    mu = number_analyser.mean(data)
    sigma = number_analyser.sample_root_mean_square_deviation(data=data)

    values, base = np.histogram(data, bins=40)
    cumulative = np.cumsum(values)
    relative_frequency = cumulative / cumulative[-1]

    x = np.linspace(min(data), max(data), 100)
    y = norm.cdf(x, mu, sigma)

    plt.plot(x, y, '-o', color='orange', label='Normal Distribution')
    plt.plot(base[:-1], relative_frequency, c='red', label='Накопичені відносні частоти')
    plt.legend()
    plt.show()

sample_generator = SampleGenerator(v=8, sum_limit=12, n=350)

data = sample_generator.data_collector()

task_2(data)
```

Приклад роботи:



3) зробити висновки про узгодження теоретичного й статистичного законів розподілів;

Аналізуючи зображення вище, можна зробити попередній висновок, що емпірична функція розподілу відповідає теоретичній.

4) знайти інтервальні оцінки для математичного сподівання та дисперсії випадкової величини

```
def task_4(data):
    mu, std = np.mean(data), np.std(data)

    confidence_level = 0.95
    degrees_freedom = len(data) - 1
    confidence_interval = stats.t.interval(confidence_level, degrees_freedom, mu, stats.sem(data))
    print(f'Довірчий інтервал для середнього: {confidence_interval}')
    variance_interval = stats.chi2.interval(confidence_level, degrees_freedom, loc=np.var(data),
                                            scale=std ** 2 / len(data))
    print(f'Довірчий інтервал для дисперсії: {variance_interval}')

sample_generator = SampleGenerator(v=8, sum_limit=12, n=350)


data = sample_generator.data_collector()


task_4(data)
```

Приклад роботи:

Для даного набору даних результатом є такі ряди

```
Довірчий інтервал для середнього: (np.float64(228.02314415079144), np.float64(230.78049260443555))
Довірчий інтервал для дисперсії: (np.float64(318.0563540909004), np.float64(368.7734937113361))
```

5) перевірити за критерієм узгодження Пірсона відповідність теоретичного розподілу емпіричним даним.

```
def task_5(data):
    num_bins = 'auto'

    observed_frequencies, bins = np.histogram(data, bins=num_bins, density=False)

    expected_density, _ = np.histogram(data, bins=bins, density=True)
    bin_widths = np.diff(bins)
    expected_frequencies = expected_density * bin_widths * len(data)

    expected_frequencies[
        expected_frequencies == 0] = 0.000001
```

```
    chi_square_stat, p_value = stats.chisquare(f_obs=observed_frequencies, f_exp=expected_frequencies)

    print("Chi-square Statistic:", chi_square_stat)
    print("P-value:", p_value)

    if p_value > 0.05:
        print('Не відхиляємо нульову гіпотезу, дані відповідають нормальному розподілу')
    else:
        print('Відхиляємо нульову гіпотезу, дані не відповідають нормальному розподілу')



sample_generator = SampleGenerator(v=8, sum_limit=12, n=350)

data = sample_generator.data_collector()

task_5(data)
```

Приклад роботи:

Для даного набору даних результатом є такий рядок

```
The theoretical distribution fits the empirical data.
Chi-squared statistic: 9.666350085263108
P-value: 0.37816495319237015
```

Висновки:

Висновок: Для виконання лабораторної роботи було використано Python разом з бібліотеками math, matplotlib, numpy, scipy. З гістограми накопичених відносних частот апроксимованої до функції нормального розподілу , видно що емпірична функція розподілу наближається до теоретичної. Довірчий інтервал вибіркового середнього і вибіркової дисперсії із заданою надійністю γ=0,95 дорівнюють відповідно
(227.78462318840104,- 230.63466450188292)
і (339.799771220534, 393.984107481455). Оскільки p-значення більша за 0.05, це означає, що дані статистично практино не відхиляються від нормального розподілу.