

Домашняя работа по Теории вероятности и Математической статистике

Работа выполнена студенткой 2-ого курса группы БЭК-212 Торовой Анной

Задача 1

Однажды в Самарканде турист заказывал Яндекс-такси. На десятом заказе впервые приехал таксист, который уже раньше приезжал к туристу. Для упрощения предположим, что все n таксистов Самарканда всегда на работе и приезжают равномерно..

```
In [ ]: import numpy as np
import pandas as pd
import math
import matplotlib.pyplot as plt
from scipy import stats
```

а) Постройте график функции правдоподобия как функции от общего количества такси n . Найдите оценку числа n методом максимального правдоподобия.

Запишем вероятность того, что на 10-ый день приедет таксист, который ранее уже приезжал

$$P = \frac{n \cdot (n-1) \cdot (n-2) \cdot \dots \cdot (n-8) \cdot 9}{n^{10}} = \frac{n!}{(n-9)!} \cdot \frac{9}{n^{10}}$$

Тогда сначала найдем оценку методом максимального правдоподобия

$$L = \ln(P) = \ln\left(\frac{n \cdot (n-1) \cdot (n-2) \cdot \dots \cdot (n-8) \cdot 9}{n^{10}}\right) = \ln\left(\frac{n!}{(n-9)!} \cdot \frac{9}{n^{10}}\right) \rightarrow$$

```
In [ ]: n_hat_ML=np.argmax(probs)+1
print(f'ML оценка числа таксистов = {n_hat_ML}')
```

Получившаяся оценка числа n методом максимального правдоподобия составляет 42

```
In [ ]: def hood(n):
    P = ((n-1)/n)*((n-2)/n)*((n-3)/n)*((n-4)/n)*((n-5)/n)*((n-6)/n)*((n-7)/n)
    return P
values = np.arange(1, 1001)
final = np.apply_along_axis(hood, axis=0, arr=values)
plt.xlabel('n - число таксистов')
plt.ylabel('Значение функции правдоподобия')
```

```
plt.title('График функции правдоподобия')
plt.plot(final)
plt.grid(True)
plt.show()
```

 График функции правдоподобия

б) Постройте график математического ожидания номера заказа, на котором происходит первый повторный приезд, как функции от общего количества такси n . Найдите оценку числа n методом моментов.

```
In [ ]: def prob(n, d):
        p = np.prod((n - np.arange(0, d - 1))/n)
        p *= (d - 1)/n
        return p

n = np.arange(1, 1001)
def exp(n):
    k = np.arange(2, n+2)
    p = np.vectorize(prob)(n, k)
    E = np.sum(p * k)
    return E

E1 = []
for i in n:
    E = exp(i)
    E1.append(E)

plt.figure(figsize = (16, 8), dpi=300)
plt.plot(n, E1, color='blue')
plt.xlabel("Количество такси - n")
plt.ylabel("Математическое ожидание")
plt.title("График математического ожидания номера заказа")
plt.grid(True)
plt.show()
```

Рассчитаем математические ожидания, используя функцию, на входе в которую будет число таксистов, рассматривая только те дни, когда есть вероятность второго таксиста

 Математическое ожидание

```
In [ ]: mom = E1.index(min(E1, key=lambda x: abs(x - 10)))+1
print(f'ММ оценка для количества таксистов = {mom}')
```

При помощи метода моментов нашли оценку числа n - 55

в) Предположим, что настоящее n равно 100. Проведя 10000 симуляций вызовов такси до первого повторного, рассчитайте 10000 оценок методом моментов и 10000 оценок методом максимального правдоподобия. Постройте гистограммы для оценок двух методов. Оцените смещение, дисперсию и среднеквадратичную ошибку двух методов.

Задача 2

Однажды в Самарканде турист заказывал Яндекс-такси. На десятом заказе он обнаружил, что у таксистов было 6 разных имён. Для упрощения предположим, что все n имён среди таксистов встречаются равновероятно и независимо от поездки к поездке.

- а) Постройте график функции правдоподобия как функции от общего количества имён n . Найдите оценку числа n методом максимального правдоподобия.

Задача 3

Иноагент Иннокентий по 20 наблюдениям строит 95%-й доверительный интервал для математического ожидания несколькими способами: классический асимптотический нормальный интервал, с помощью наивного бутстрэпа, с помощью бутстрэпа t -статистики.

- а) Для каждого способа с помощью 10000 симуляций оцените вероятность того, что номинально 95%-й доверительный интервал фактически накрывает математическое ожидание, если наблюдения распределены экспоненциально с интенсивностью 1.

```
In [ ]: import numpy as np
from scipy.stats import norm, expon, t
import scipy.stats as st

n = 10000
sample = 20
confidence = 0.95
alpha = 0.05

d = st.expon.rvs(scale=1, size=(n, sample), random_state=42)
```

Классический асимптотический нормальный интервал

```
In [ ]: c = 0

for i in d:
    # Вычисление 95%-го доверительного интервала с использованием нормаль
    ci = st.norm.interval(confidence, loc=np.mean(i), scale=np.std(i), ddc

    # Проверка, попадает ли математическое ожидание в доверительный интер
    if ci[0] < 1/h and ci[1] > 1/h:
        c += 1

# Вычисление вероятности попадания математического ожидания в доверительн
p = c / n
```

```
print(f'Вероятность, что 95%-ый доверительный интервал фактически покрывает')
```

Получаем, что вероятность, что 95%-ый интервал фактически покрывает математическое ожидание, составляет 0.904 (ответ округлен до тысячных)

Интервал с помощью наивного бутстрэпа

```
In [ ]: c = 0

for i in d:
    # Генерация бутстрэп-выборки
    bootstrap_d = np.random.choice(i.flatten(), size=(size, n), replace=True)

    # Вычисление среднего значения для бутстрэп-выборки
    bootstrap_mean = np.mean(bootstrap_d, axis=1)

    # Вычисление 95%-го доверительного интервала на основе бутстрэп-выборки
    ci = np.percentile(bootstrap_mean, [2.5, 97.5])

    # Проверка, попадает ли математическое ожидание в доверительный интервал
    if ci[0] <= 1 / h <= ci[1]:
        c += 1

# Вычисление вероятности попадания математического ожидания в доверительный интервал
p = c / d.shape[0]

print(f'Вероятность, что 95%-ый доверительный интервал фактически покрывает')
```

Получаем, что вероятность, что 95%-ый интервал фактически покрывает математическое ожидание, составляет 0.903 (ответ округлен до тысячных)

Интервал с помощью бутстрэпа t-статистики

```
In [ ]: c = 0

# Вычисление среднего значения и стандартного отклонения для всего набора данных
d_mean = np.mean(d)
d_std = np.std(d)

for i in d:
    # Генерация бутстрэп-выборки
    bootstrap_d = np.random.choice(i.flatten(), size=(size, n), replace=True)

    # Вычисление среднего значения и стандартного отклонения для бутстрэп-выборки
    bootstrap_mean = np.mean(bootstrap_d, axis=1)
    bootstrap_std = np.std(bootstrap_d, axis=1, ddof=1)

    # Вычисление t-статистики
    t = (bootstrap_mean - d_mean) / (bootstrap_std / np.sqrt(n))
```

```

# Вычисление доверительного интервала на основе t-статистики
ci = d_mean - np.percentile(t, [97.5, 2.5]) * (d_std / np.sqrt(n))

# Проверка, попадает ли 0 в доверительный интервал
if ci[0] <= 0 <= ci[1]:
    c += 1

# Вычисление p-value
p = c / size

```

Получаем, что вероятность, что 95-% интервал фактически покрывает математическое ожидание, составляет 0.904 (ответ округлен до тысячных)

- б) Пересчитайте вероятности покрытия, если наблюдения имеют распределение Стьюдента с тремя степенями свободы

Классический асимптотический доверительный интервал

```

In [ ]: c = 0

for i in range(d.shape[0]):
    # Вычисление среднего значения и стандартного отклонения для текущего
    d_mean = np.mean(d[i])
    d_std = np.std(d[i], ddof=1)

    # Вычисление доверительного интервала на основе нормального распредел
    ci = stats.norm.interval(1 - alpha, loc=d_mean, scale=d_std / np.sqrt

    # Проверка, попадает ли 0 в доверительный интервал
    if ci[0] <= 0 <= ci[1]:
        c += 1

# Вычисление p-value
p = c / d.shape[0]

```

Получаем, что вероятность, что 95-% интервал фактически покрывает математическое ожидание, составляет 0.938 (ответ округлен до тысячных)

Наивный бутстреп

```

In [ ]: c = 0

for i in d:
    # Генерация бутстрэп-выборки
    bootstrap_d = np.random.choice(i.flatten(), size=(size, n), replace=T

    # Вычисление среднего значения для каждой бутстрэп-выборки
    bootstrap_mean = np.mean(bootstrap_d, axis=1)

    # Вычисление доверительного интервала на основе перцентилей средних з
    ci = np.percentile(bootstrap_mean, [2.5, 97.5])

```

```

# Проверка, попадает ли 0 в доверительный интервал
if ci[0] <= 0 <= ci[1]:
    c += 1

# Вычисление p-value
p = c / d.shape[0]

```

Получаем, что вероятность, что 95-% интервал фактически покрывает математическое ожидание, составляет 0.917 (ответ округлен до тысячных)

Бутстреп t-статистики

```

In [ ]: c = 0

for i in d:
    # Генерация бутстреп-выборки
    bootstrap_sample = np.random.choice(i.flatten(), size=(size, n), repl

    # Вычисление среднего значения для каждой бутстреп-выборки
    bootstrap_mean = np.mean(bootstrap_d, axis=1)

    # Вычисление стандартного отклонения для каждой бутстреп-выборки
    bootstrap_std = np.std(bootstrap_d, axis=1, ddof=1)

    # Вычисление t-статистики для каждой бутстреп-выборки
    t = (bootstrap_mean - np.mean(d)) / (bootstrap_std / np.sqrt(n))

    # Вычисление доверительного интервала
    ci = np.mean(d) - np.percentile(t_statistic, [97.5, 2.5]) * (np.std(d

    # Проверка, попадает ли 0 в доверительный интервал
    if ci[0] <= 0 <= ci[1]:
        c += 1

# Вычисление p-value
p = c / size

```

Получаем, что вероятность, что 95-% интервал фактически покрывает математическое ожидание, составляет 0.918 (ответ округлен до тысячных)

Для экспоненциального распределения наилучший результат показал бутстреп t-статистики. Для распределения Стюдента с тремя степенями свободы лучший результат показал классический асимптотический способ.

Задача 4

. Проверьте гипотезу о том, что ожидаемые результаты экзамена по теории вероятностей тех, у кого фамилия начинается с гласной буквы и с согласной буквы, равны. В качестве альтернативной гипотезы возьмите гипотезу о неравенстве.

а) Используйте тест Уэлча.

```
In [ ]: import pandas as pd

df = pd.read_csv('prob_exam_data.csv', delimiter=';')

# Удаление строк с отсутствующими значениями в столбце 'Last name'
df = df.dropna(subset=['Last name'])

# Функция для определения, является ли первая буква слова гласной
def is_vowel(w):
    vowel = 'АОУЮЯЕЁЭИ'
    return w[0].upper() in vowel

# Создание DataFrame для группы с фамилиями, начинающимися на гласные
vow = df[df['Last name'].apply(is_vowel)].reset_index(drop=True)

# Создание DataFrame для группы с фамилиями, начинающимися на согласные
cons = df[~df['Last name'].apply(is_vowel)].reset_index(drop=True)

In [ ]: from scipy.stats import ttest_ind

# Выполнение непараметрического t-теста для двух групп
stat, pvalue = ttest_ind(vow['Result'], cons['Result'], equal_var=False)

# Вывод значения p-value
print(f'P_value = {pvalue:.3f}')

# Проверка статистического вывода на основе значения p-value
if pvalue > alpha:
    print('Статистический вывод: H_0 не отвергается')
else:
    print('Статистический вывод: H_0 отвергается в пользу H_A')
```

P-value 0.4

H0 не отвергается

б) Наивный бутстрэп

```
In [ ]: # Задание seed для воспроизводимости результатов
np.random.seed(42)

# Количество бутстрэп-переменных
n = 100000

# Бутстрэпирование значений для группы "vowel"
vowel_boot = np.random.choice(vowel['Result'], size=(n, len(vow)))

# Бутстрэпирование значений для группы "cons"
cons_boot = np.random.choice(cons['Result'], size=(n, len(cons)))

# Вычисление разности средних между двумя группами
d = np.mean(vow['Result']) - np.mean(cons['Result'])

# Вычисление разности средних для каждого бутстрэп-переменного
d_boot = np.mean(vowel_boot, axis=1) - np.mean(cons_boot, axis=1)
```

```
# Вычисление p-value
pvalue = 2 * min(np.mean(d_boot <= 0), np.mean(d_boot >= 0))

# Вывод результатов
print(f'P_value = {pvalue:.3f}')
if pvalue > alpha:
    print('Статистический вывод: H_0 не отвергается')
else:
    print('Статистический вывод: H_0 отвергается в пользу H_A')
```

P_value = 0.401

H_0 не отвергается

Используйте бутстрэп t-статистики

```
In [ ]: # Вычисление разности средних между двумя группами
d = np.mean(vow['Result']) - np.mean(cons['Result'])

# Вычисление разности средних для каждого бутстрэп-переменного
d_boot = np.mean(vow_boot, axis=1) - np.mean(cons_boot, axis=1)

# Вычисление стандартной ошибки для каждого бутстрэп-переменного
boot_vow_var = np.var(vowel_boot, ddof=1, axis=1) / len(vow)
boot_cons_var = np.var(cons_boot, ddof=1, axis=1) / len(cons)
boot = np.sqrt(boot_vow_var + boot_cons_var)

# Вычисление значения статистики H_boot
h_boot = (d_boot - d) / boot

# Вычисление стандартной ошибки для исходных данных
vow_var = np.var(vow, ddof=1) / len(vow)
cons_var = np.var(cons, ddof=1) / len(cons)
boot_s = np.sqrt(vow_var + cons_var)

# Вычисление значения статистики H
h = d / boot_s

# Вычисление p-value
pvalue = 2 * min(np.mean(h_boot <= h[0]), np.mean(h_boot >= h[0]))

# Вывод результатов
print(f'P_value = {pvalue:.3f}')
if pvalue > alpha:
    print('Статистический вывод: H_0 не отвергается')
else:
    print('Статистический вывод: H_0 отвергается в пользу H_A')
```

P_value = 0.390

H_0 не отвергается

Используйте перестановочный тест

```
In [ ]: # Создание столбца 'First letter' на основе столбца 'Last name' с примене
df['First letter'] = df['Last name'].apply(is_vowel)

np.random.seed(42)
```



```

# Формирование данных для каждых 4 элементов
data_per_4 = df[['Result', 'First letter']]

# Расчет разности средних между группами гласных и согласных
delta = np.mean(vowel['Result']) - np.mean(cons['Result'])

n_per = 10000

deltas_per = []
for i in range(n_per):
    # Перемешивание значений столбца 'First letter'
    data_per_4['First letter'] = np.random.permutation(data_per_4['First letter'])

    # Расчет разности средних для каждой перестановки
    delta_per = data_per_4.loc[data_per_4['First letter'] == True, 'Result'].mean() - data_per_4.loc[data_per_4['First letter'] == False, 'Result'].mean()
    deltas_per.append(delta_per)

# Вычисление p-value
pvalue = 2 * min(np.mean(np.array(deltas_per) <= delta), np.mean(np.array(deltas_per) >= delta))

print(f'P_value = {pvalue:.3f}')
if pvalue > alpha:
    print('Статистический вывод: H_0 не отвергается')
else:
    print('Статистический вывод: H_0 отвергается в пользу H_A')

```

Задача 5

Составьте таблицу сопряжённости, поделив студентов писавших экзамен на четыре группы по двум признакам: набрал ли больше медианы или нет, на согласную или гласную букву начинается фамилия.

Постройте 95% асимптотический интервал для отношения шансов хорошо написать экзамен («несогласных» к «согласным»). Проверьте гипотезу о том, что отношение шансов равно 1 и укажите P-значение.

```

In [ ]: # Вычисление количества студентов с баллами выше и ниже медианного значения
vow_more = len(vow[stud['Result'] > score_median[0]]) # Количество студ
vow_low = len(vow[stud['Result'] < score_median[0]]) # Количество студ
cons_more = len(cons[stud['Result'] > score_median[0]]) # Количество студ
cons_low = len(cons[stud['Result'] < score_median[0]]) # Количество студ

# Создание таблицы с результатами
table = pd.DataFrame({'Группа': ['Гласные', 'Согласные'],
                      'Больше медианы': [vow_more, cons_more],
                      'Меньше медианы': [vow_low, cons_low]})

# Вычисление шансов и отношения шансов
chance_vow = vow_more / vow_low # Шансы гласных
chance_cons = cons_more / cons_low # Шансы согласных
OR = chance_vow / chance_cons # Отношение шансов

# Вычисление стандартной ошибки и границ доверительного интервала
ci = np.sqrt(1 / vow_more + 1 / vow_low + 1 / cons_more + 1 / cons_low)
left_bound = OR * np.exp(-1.96 * ci) # Левая граница доверительного интервала

```

```

right_bound = OR * np.exp(1.96 * ci) # Правая граница доверительного интервала
left_bound, right_bound # Вывод значений левой и правой границы доверительного интервала
print(f'Интервал: [{left_bound:.3f}, {right_bound:.3f}]')

# Расчёт p_value
p_value = (1 - sts.norm.cdf(OR)) * 2
print(f'P_value = {pvalue:.3f}')

```

Pvalue = 0.475

Нулевая гипотеза не отвергается

Постройте 95% асимптотический интервал для отношения вероятностей
хорошо написать экзамен. Проверьте гипотезу о том, что отношение
вероятностей равно 1 и укажите Р-значение.

```

In [ ]: import math
        from scipy.stats import norm

        # Вычисление отношения шансов (odds ratio)
        diff = (cons_more / (cons_more + cons_low)) / (vow_more / (vow_more + vow_low))

        # Вычисление логарифма отношения шансов
        log_diff = math.log(diff)

        # Вычисление стандартной ошибки (standard error)
        se = math.sqrt((1 / (cons_more + cons_low)) + (1 / (vow_more + vow_low)))

        # Вычисление значения z-статистики (z-score)
        z = (log_diff - 0) / se

        # Вычисление 95%-го доверительного интервала
        CI = [np.exp(np.log(diff) - 1.96 * se), np.exp(log_diff + 1.96 * se)]

        # Вычисление p-value
        p_value = 2 * (1 - norm.cdf(abs(z)))

```

P value = 0.282

Нулевая гипотеза не отвергается

Постройте 95% интервал для отношения шансов хорошо написать экзамен с
помощью наивного бутстрэпа. Проверьте гипотезу о том, что отношение
шансов равно 1 и укажите Р-значение.

```

In [ ]: # Заданные переменные
        median = df['Result'].median()
        n = 1000
        R = []

        # Проводим бутстреп
        for i in range(n):
            # Ресэмплирование данных

```

```

resampled_data = df.sample(frac=1, replace=True)

# Вычисление отношения шансов
group_vowel_resampled = resampled_data[resampled_data['Vowel Start']]
group_consonant_resampled = resampled_data[resampled_data['Consonant S

group_vowel_higher_resampled = group_vowel_resampled[group_vowel_resa
group_vowel_lower_resampled = group_vowel_resampled[group_vowel_resam

group_consonant_higher_resampled = group_consonant_resampled[group_co
group_consonant_lower_resampled = group_consonant_resampled[group_con

# Вычисление отношения шансов
ratio = (group_consonant_higher_resampled / group_consonant_lower_res
R.append(ratio)

# Вычисление границ доверительного интервала
lower_bound = np.percentile(R, 2.5)
upper_bound = np.percentile(R, 97.5)

# Вычисление наблюдаемого отношения шансов
observed_ratio = ((group_vowel_higher / group_vowel_lower) / (group_conso

# Вычисление p-value
p_value = (np.abs(np.array(R) - 1) >= np.abs(observed_ratio - 1)).mean()

```

P value = 0.631

Нулевая гипотеза не отвергается

Задача 6

Иноагент Иннокентий Вероятностно-Статистический считает, что длина фамилии положительно влияет на результат экзамена по теории вероятностей. А именно, он предполагает, что ожидаемый результат за экзамен прямо пропорционален длине фамилии, $E(Y_i) = \beta F_i$, где Y_i — результат за экзамен по 30-балльной шкале, F_i — количество букв в фамилии.

Оцените β методом моментов. Рассчитайте выборочную корреляцию

Вычисление среднего значения Y

```
Y = df_students['Score'] Y_mean = df_students['Score'].mean()
```

Вычисление суммы длин фамилий

```
len_sum = df_students['Last name'].apply(lambda x: len(x))
```

Вычисление средней длины фамилий

```
mean_len = len_sum.mean()
```

Вычисление суммы длин фамилий

```
total_len_sum = sum(len_sum)
```

Вычисление коэффициента beta

```
beta = Y_mean * df_students.shape[0] / total_len_sum
```

Вычисление корреляции

```
corr = sum((len_sum - mean_len) * (Y - Y_mean)) / np.sqrt(sum((len_sum - mean_len) ** 2))
```

Получившиеся значения:

Y_mean = 16.2

beta = 2.06

corr = 0.025

Небольшой комментарий:

$$E(Y_i) = E(Y_{mean}) = Y_{mean} E(Y_i) = \frac{1}{n} * \beta * \sum F_i n$$

С помощью перестановочного теста найдите Р-значение и формально протестируйте гипотезу о том, что корреляция равна нулю.

```
In [ ]: import matplotlib.pyplot as plt

# Создаем список из значений len_sum
n = list(len_sum)

# Создаем пустой список для хранения перестановочных корреляций
A = [0] * 10 ** 6

# Выполняем 10^6 итераций перестановки
for i in range(10 ** 6):
    random.seed(10) # Задаем одинаковое начальное состояние генератора с
    random.shuffle(n) # Перемешиваем список n
    n = np.array(n) # Преобразуем список в массив NumPy
    d = sum((n - mean_len) * (Y - Y_mean)) / np.sqrt(sum((n - mean_len) ** 2))
    A[i] = d

# Вычисляем границы доверительного интервала
left = np.quantile(A, alpha/2)
```

```
right = np.quantile(A, 1 - alpha/2)

p_value = len([i for n in A if i > corr]) * 2 / 10 ** 6
```

Задача 7

С помощью chatgpt решите любую задачу из нашего курса теории вероятностей и статистики. Можно брать задачи из прошлых контрольных, лекций, семинаров и даже этого домашнего задания. В качестве ответа приведите полный диалог с chatgpt.

Условие задачи: Величина X равномерна на $[3a - 2; 3a + 7]$. При каких α и β оценка $\hat{a} = \alpha + \beta X$ неизвестного параметра a будет несмещённой?

В изначальном запросе было введено условие задачи

Ответ:

Для оценки

$$\hat{a} = \alpha + \beta X$$

, чтобы она была несмещённой, ожидаемое значение оценки должно быть равно истинному значению параметра a . В данном случае, ожидаемое значение оценки \hat{a} равно

$$E[\hat{a}] = E[\alpha + \beta X]$$

.

Используя свойство линейности математического ожидания, мы можем записать:

$$E[\hat{a}] = E[\alpha] + E[\beta X]$$

Так как X имеет равномерное распределение на интервале

$$[3a - 2, 3a + 7]$$

, то $E[X]$ равно среднему значению этого интервала. Следовательно, $E[X] = (3a - 2 + 3a + 7) / 2 = 6a + 5$.

Таким образом, чтобы оценка $\hat{a} = \alpha + \beta X$ была несмещённой, необходимо, чтобы ожидаемые значения α и βX равнялись нулю: $E[\alpha] + E[\beta X] = 0$

$$E[\alpha] = 0$$

(условие для α)

$$E[\beta X] = 0 \text{ (условие для } \beta X)$$

$$E[\alpha] = \alpha, \text{ поэтому } \alpha = 0.$$

$$E[\beta X] = \beta * E[X] = 0$$

, следовательно,

$$E[X] = 0$$

или

$$\beta = 0.$$

Таким образом, чтобы оценка

$$\hat{a} = \alpha + \beta X$$

была несмещённой, параметр α должен быть равен 0, а параметр β может быть любым. \$\$

Далее спросила у него про условия несмещённости

Условие несмещённой оценки состоит в том, что математическое ожидание оценки должно быть равно истинному значению оцениваемого параметра.

Формально, для оценки \hat{a} параметра a считается несмещённой, если выполняется условие:

$$E[\hat{a}] = a,$$

где $E[\hat{a}]$ обозначает математическое ожидание оценки \hat{a} , а a представляет истинное значение параметра a .

Таким образом, для того чтобы оценка была несмещённой, математическое ожидание оценки должно совпадать с истинным значением параметра.



Уверен?



Далее GPT еще около 5 минут обрабатывал запрос и так и не выдал ответа

В целом рассуждения чата правильные. Наверное, стоило взять более сложную задачу

Задача 8

Укажите любой источник по теории вероятностей или статистике, который вам оказался полезен в течение года. Это может быть статья, видео, задача, всё что угодно. Объясните, с чем конкретно этот источник помог разобраться. Лучше привести в пример внешний источник, не упомянутый на вики курса, но можно и внутренний.

Очень хочется прикрепить данное всеобъемлющее видео - <https://www.youtube.com/watch?v=4Z5-dMRCpnQ> Лучший друг по всем предметам на нашем факультете

Мне потребовались скорее базовые видео, которые объясняют еще раз в общих чертах и простыми словами термины

студент - <https://www.youtube.com/watch?v=1GZVbmOwl2k>

p-value (со стороны фкн) - <https://www.youtube.com/watch?v=vXHulmmXNCo>

Сложно привести все источники, так как обычно смотрю с телефона где-то в транспорте для "внедрения в уши"