

In [166...

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
import pandas as pd
from tqdm import tqdm
```

Задача 3

(a)

Асимптотический доверительный интервал

In [340...

```
np.random.seed(7645)
sample_exp_3 = np.random.exponential(scale = 1, size = (10000,20))
```

In [341...

```
sample_exp_3.mean()
```

Out[341...

1.0000577124221173

In [100...

```
n = 20
means = sample_exp_3.mean(axis = 1)
z = stats.norm.isf(0.025)
```

In [100...

z

Out[100...

1.9599639845400545

асимптотический интервал для **математического ожидания**

$$\bar{X} - z_{0.975} \cdot \sqrt{\frac{\hat{\sigma}^2}{n}} < \mu < \bar{X} + z_{0.975} \cdot \sqrt{\frac{\hat{\sigma}^2}{n}}$$

In [100...

```
expectation_ci = np.array([means - z * np.sqrt(sample_exp_3.var(ddof = 1, axis = 1)/
```

In [100...

```
count_in = 0
for i,j in expectation_ci:
    if i < 1 and 1 < j:
        count_in +=1
```

In [100...

```
count_in/10000
```

Out[100...

0.9008

Получили оценку, что интервал покрывает действительное мо с вероятностью 0.9

Наивный бутстрап

для каждой из 10000 выборок с экспоненциальным распределением генерирую 10000 бутстрап выборок среднего, затем строю по ним 95-% интервал с помощью квантилей

$$q_{left}\left(\frac{1}{\lambda_{boot}}\right) < \frac{1}{\lambda} < q_{right}\left(\frac{1}{\lambda_{boot}}\right)$$

In [166...

```
np.random.seed(2)
ci_covers = 0
for i in tqdm(range(10000)):
    X = sample_exp_3[i]
    bootstrapped_X = np.random.choice(X, size = (10000,20))
    R_hat = bootstrapped_X.mean(axis = 1)
    q_right = np.percentile(R_hat, 97.5)
    q_left = np.percentile(R_hat, 2.5)
    ci = [q_left, q_right]
    if ci[0] < 1 < ci[1]:
        ci_covers += 1
```

100%|██████████| 10000/10000 [00:44<00:00, 222.81it/s]

In [100...

```
ci_covers / 10000
```

Out[100...

0.9018

Получил оценкк вероятности накрытия интервалом действительного параметра равную 0.902

Бутстрэп t-статистики

Тут для каждой изначальной выборки генерирую 10000 бутстрап выборок, затем считаю следующую статистику:

$$\frac{\bar{X}^* - \bar{X}}{\sqrt{\frac{\hat{\sigma}^2}{n}}}$$

По распределнию полученных статистик высчитываю квантили q_{left} и q_{right} и строю 95%-интервал

$$\bar{X} - q_{right} \cdot \sqrt{\frac{\hat{\sigma}^2}{n}} < \frac{1}{\lambda} < \bar{X} - q_{left} \cdot \sqrt{\frac{\hat{\sigma}^2}{n}}$$

In [101...

```
n = 20
```

In [101...

```
np.random.seed(9)
ci_covers = 0
for i in tqdm(range(10000)):
    X = sample_exp_3[i]
    bootstrapped_X = np.random.choice(X, size = (10000, 20))
    R_hat = (bootstrapped_X.mean(axis = 1) - X.mean())/(np.sqrt(bootstrapped_X.var(d
    q_right = np.percentile(R_hat, 97.5)
    q_left = np.percentile(R_hat, 2.5)
    ci = [X.mean() - q_right * np.sqrt(X.var(ddof = 1)/n), X.mean()- q_left * np.sqr
    if ci[0] < 1 < ci[1]:
        ci_covers += 1
```

100%|██████████| 10000/10000 [01:00<00:00, 165.38it/s]

Получил оценку вероятности накрытия интервалом действительного мо равную

```
In [101... ci_covers/10000 #ура норм
```

```
Out[101... 0.9459
```

(6)

Теперь для стьюдента с 3 степенями свободы. Действия абсолютно аналогичные пункту (a)

```
In [101... sample_t_3 = np.random.standard_t(df = 3, size = (10000, 20))
```

Асимптотический интервал:

```
In [101... intervals = np.array([sample_t_3.mean(axis = 1) - z * np.sqrt(np.var(sample_t_3, axi
```

```
In [101... ci_covers = 0
for i in tqdm(range(10000)):
    if intervals[i][0] < 0 < intervals[i][1]:
        ci_covers += 1
```

100%|██████████| 10000/10000 [00:00<00:00, 627429.58it/s]

Получил оценку вероятности накрытия действительного мо асимптотическим доверительным интервалом равную

```
In [102... ci_covers/ 10000
```

```
Out[102... 0.942
```

Наивный бутстрап

```
In [102... np.random.seed(2)
ci_covers = 0
for i in tqdm(range(10000)):
    X = sample_t_3[i]
    bootstrapped_X = np.random.choice(X, size = (10000,20))
    R_hat = bootstrapped_X.mean(axis = 1)
    q_right = np.percentile(R_hat, 97.5)
    q_left = np.percentile(R_hat, 2.5)
    ci = [q_left, q_right]
    if ci[0] < 0 < ci[1]:
        ci_covers += 1
```

100%|██████████| 10000/10000 [00:44<00:00, 226.23it/s]

Оценка вероятности накрытия:

```
In [102... ci_covers/10000
```

```
Out[102... 0.9183
```

Бутстрап t-статистики

In [102...

```

np.random.seed(9)
ci_covers = 0
for i in tqdm(range(10000)):
    X = sample_t_3[i]
    bootstrapped_X = np.random.choice(X, size = (10000, 20))
    R_hat = (bootstrapped_X.mean(axis = 1) - X.mean())/(np.sqrt(bootstrapped_X.var(d
    q_right = np.percentile(R_hat, 97.5)
    q_left = np.percentile(R_hat, 2.5)
    ci = [X.mean() - q_right * np.sqrt(X.var(ddof = 1)/n), X.mean() - q_left * np.sqr
    if ci[0] < 0 < ci[1]:
        ci_covers += 1

```

100%|██████████| 10000/10000 [01:00<00:00, 165.36it/s]

Оценка вероятности накрытия:

In [102...

```
ci_covers/10000
```

Out[102...

0.9259

(в) Для экспоненциального распределения лучше всего показал себя бутстреп t-статистики, для t-распределения - обычный асимптотический интервал. Подозреваю, так вышло потому, что t распределение похоже на нормальное и довольно быстро к нему сходится при больших n.

Задача 1

(а)

Пусть X - случайная величина, которая показывает, на каком заказе к туристу впервые приезжает уже возивший его таксист. $\mathbb{P}(X \geq 2) = 1$

$$\begin{aligned}\mathbb{P}(X = 2) &= \frac{1}{n} \\ \mathbb{P}(X = 3) &= \frac{n-1}{n} \cdot \frac{2}{n} \\ \mathbb{P}(X = 4) &= \frac{n-1}{n} \cdot \frac{n-2}{n} \cdot \frac{3}{n} \\ \mathbb{P}(X = a) &= \frac{a-1}{n} \cdot \prod_{i=0}^{a-2} \frac{n-i}{n}\end{aligned}$$

Функция правдоподобия в задаче:

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

In [103...

```

def L_10(x):
    L = 9 / x
    for i in range(1, 9):
        L *= (x - i) / x
    return L

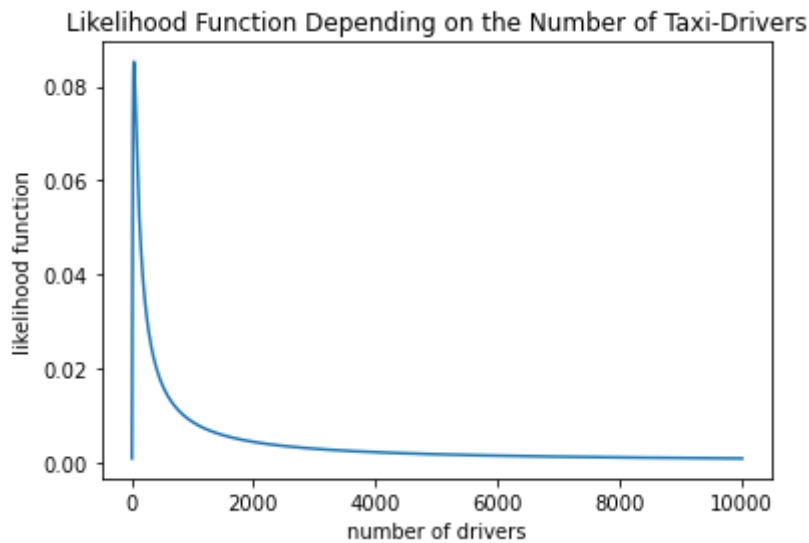
```

In [103...

```
ns = np.arange(9, 10000)
```

In [103...

```
plt.plot(ns, L_10(ns))
plt.xlabel('number of drivers')
plt.ylabel('likelihood function')
plt.title('Likelihood Function Depending on the Number of Taxi-Drivers');
```



Найдем ML-оценку

$$\ln L = \ln 9 + \ln(n-1) + \ln(n-2) + \dots + \ln(n-8) - 9 \ln n \longrightarrow \max_n$$

$$\frac{\partial \ln L}{\partial n} = \frac{1}{\hat{n}-1} + \dots + \frac{1}{\hat{n}-8} - \frac{9}{\hat{n}} = 0$$

Получится многочлен 8-й степени. Ручками такое не решится. Визуально, заметно, что функция имеет один максимум; $\lim_{n \rightarrow \inf} L(n) = 0$, Попробуем перебрать все целые n от 9 до 10000.

In [103...

```
max_L = -1
arg_max_L = -1
for n in range(9, 10001):
    L = L_10(n)
    if L > max_L:
        max_L = L
        arg_max_L = n
print(arg_max_L)
```

42

$$\hat{n}_{ml} = 42$$

(6)

In [103...

```
def expectation_taxi(n):
    expect = 0
    for i in range(2, n + 2):
        prod = 1
        for j in range(i - 1):
            prod *= (n - j) / n
        expect += i * (i - 1) * prod / n
    return expect
```

Математическое ожидание в зависимости от n :

$$\mathbb{E}X = \sum_{a=2}^{n+1} a * \mathbb{P}(X = a) = \sum_{a=2}^{n+1} a * \frac{a-1}{n} \cdot \prod_{i=0}^{a-2} \frac{n-i}{n}$$

In [103...

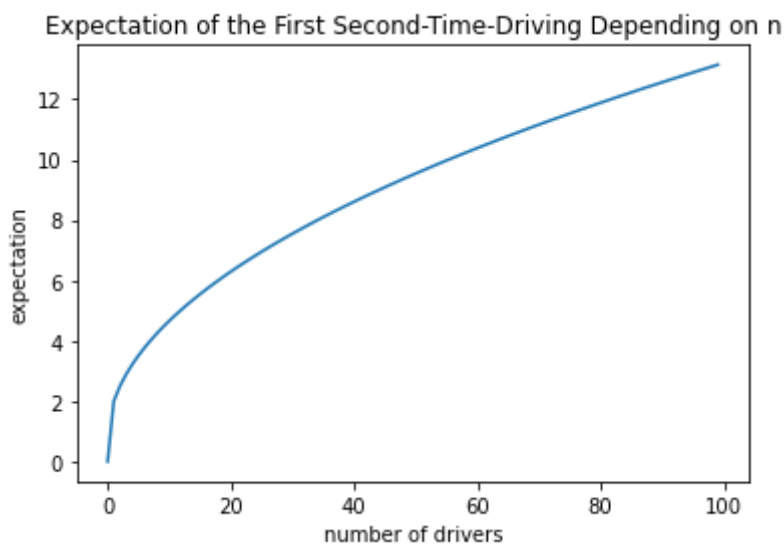
```
ns = np.arange(100)
```

In [103...

```
expectations = np.array([])
for n in ns:
    expectations = np.append(expectations, expectation_taxi(n))
```

In [103...

```
plt.plot(ns, expectations);
plt.xlabel('number of drivers')
plt.ylabel('expectation')
plt.title('Expectation of the First Second-Time-Driving Depending on n');
```



Выборочный момент: $\bar{X} = X_1 = 10$ (у нас одно наблюдение)

Судя по картинке, матожидание монотонно возрастает в зависимости от n (что логично), поэтому переберем циклом разные n .

In [104...

```
mse = np.array([])
for i in range(0,100):
    mse = np.append(mse, (expectations[i] - 10) ** 2)
```

Тогда ближайший n , к целочисленной оценке методом моментов:

In [104...

```
ns[mse.argmin()]
```

Out[104...

55

$\hat{n}_{mm} = 55$

(в)

напишем функцию правдоподобия в общем случае

$$L = \mathbb{P}(X = a) = \frac{a-1}{n} \cdot \prod_{i=0}^{a-2} \frac{n-i}{n}$$

In [104...

```
def Likelihood(a: int, n: int):
    L = (a - 1)/n
    prod = 1
    for i in range(a - 1):
        prod *= (n - i) / n
    L *= prod
    return L
```

Всего 100 таксистов, значит на 101 заказе точно будет хотя бы одно повторение.

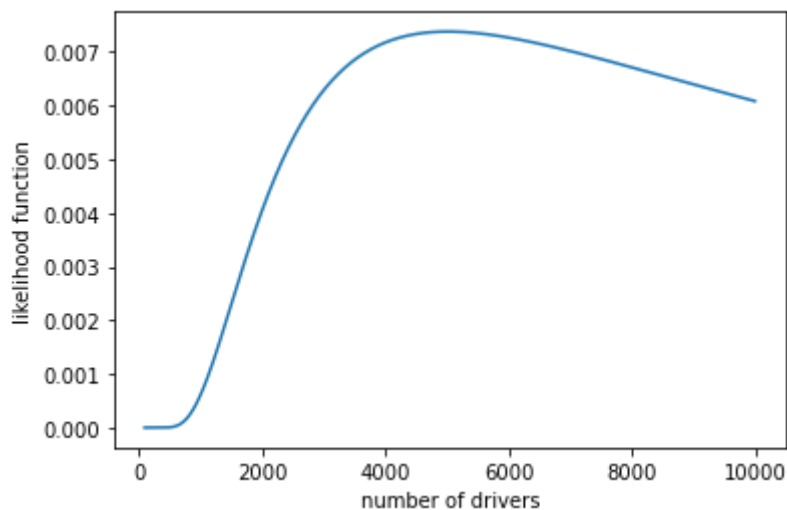
In [104...

```
drivers = np.arange(1,101)
```

минимум функции правдоподобия я ищу перебирая все значения для $\hat{n}_m l$ от номера заказа, на котором впервые приехал уже возивший таксист - 1 ($taxi_n - 1$), до $taxi_n \cdot 100$. В худшей ситуации (когда только на 101 заказе приедет старый таксист), график функции правдоподобия в зависимости от n будет таким:

In [121...

```
plt.plot(np.arange(100, 10000), Likelihood(101, np.arange(100, 10000)));
plt.xlabel('number of drivers')
plt.ylabel('likelihood function');
```



т.е ml оценка найдется!

Для нахождения mm-оценки, я считаю среднеквадратичные ошибки матожидания от среднего для n от $taxi_n - 1$ до $taxi_n \cdot 100$. я останавливаю цикл, как только ошибка начинает возрастать.

Эта функция считает оптимальные mm и ml оценки

In [818...

```
np.random.seed(986)
ml_estimates = []
mm_estimates = []
for g in tqdm(range(10000)):
    observation = np.array([])
    for i in range(101):
        taxi = np.random.choice(drivers)
        if taxi in observation:
```

```

        break
    else:
        observation = np.append(observation, taxi)
        taxi_n = observation.shape[0] + 1
        ns_ml = np.arange(taxi_n - 1, taxi_n * 100)
        Ls = Likelihood(a = taxi_n, n = ns_ml)
        n_ml = ns_ml[Ls.argmax()]
        ns_mm = np.arange(taxi_n - 1, taxi_n * 100)
        mse = np.array([])
        for i, j in enumerate(ns_mm):
            mse = np.append(mse, np.power(expectation_taxi(j) - taxi_n, 2))
            if i == 0:
                continue
            if mse[i] > mse[i - 1]:
                break
        n_mm = ns_mm[mse.argmin()]
        ml_estimates.append(n_ml)
        mm_estimates.append(n_mm)

```

100%|██████████| 10000/10000 [1:41:40<00:00, 1.64it/s]

In [839...

```

ml_estimates= np.array(ml_estimates)
mm_estimates = np.array(mm_estimates)

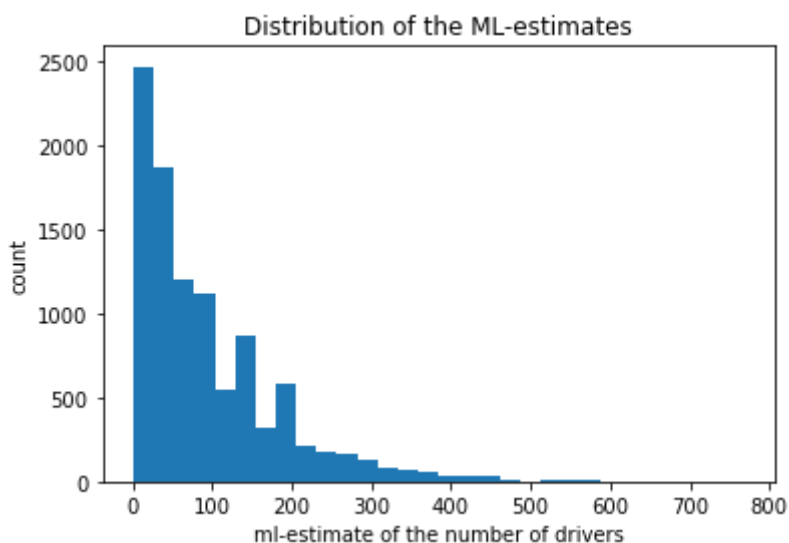
```

In [836...

```

plt.hist(ml_estimates, bins = 30);
plt.xlabel('ml-estimate of the number of drivers')
plt.ylabel('count')
plt.title('Distribution of the ML-estimates');

```



Оценка смещения ml-оценки:

In [841...

```
ml_estimates.mean() - 100
```

Out[841...

-4.340000000000003

Оценка дисперсии ml-оценки:

In [842...

```
ml_estimates.var(ddof = 1)
```

Out[842...

8283.447744774478

Среднеквадратичная ошибка ml-оценки

In [913...]

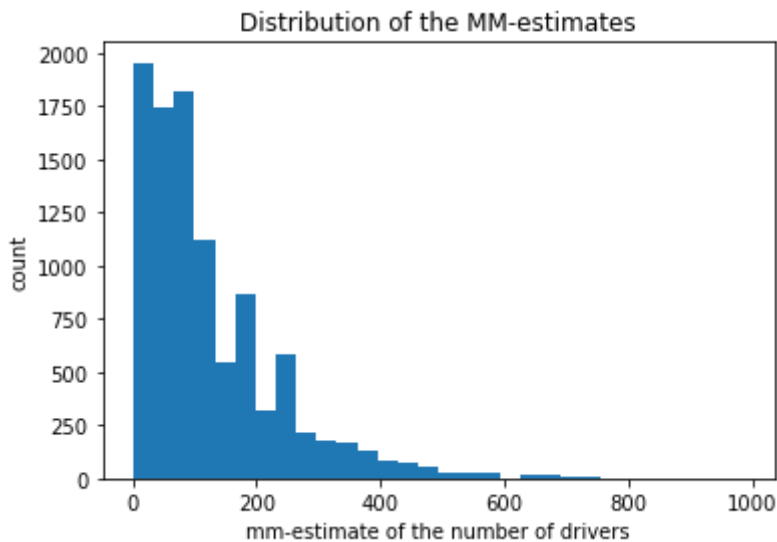
```
np.sum((ml_estimates - 100) ** 2)/10000
```

Out[913...]

8301.455

In [838...]

```
plt.hist(mm_estimates, bins = 30);
plt.xlabel('mm-estimate of the number of drivers')
plt.ylabel('count')
plt.title('Distribution of the MM-estimates');
```



Оценка смещения mm-оценки:

In [844...]

```
mm_estimates.mean() - 100
```

Out[844...]

24.783900000000003

Оценка дисперсии mm-оценки:

In [845...]

```
mm_estimates.var(ddof = 1)
```

Out[845...]

13726.45604639464

Среднеквадратичная ошибка mm-оценки:

In [117...]

```
np.sum((mm_estimates - 100) ** 2)/10000
```

Out[117...]

14339.3251

Задача 2

(а) Пусть X - случайная величина, которая равняется количеству уникальных имен таксистов. Сначала рассмотрим более простой случай. Допустим, делается 6 заказов, за которые мы встречаем 3 уникальных имен таксистов (всего таксистов n). Тогда

$$\begin{aligned} \mathbb{P}_6(X=3) = & \frac{1}{n} \cdot \frac{1}{n} \cdot \frac{1}{n} \cdot \frac{n-1}{n} \cdot \frac{n-2}{n} + \frac{1}{n} \cdot \frac{1}{n} \cdot \frac{n-1}{n} \cdot \frac{2}{n} \cdot \frac{n-2}{n} + \\ & + \frac{1}{n} \cdot \frac{1}{n} \cdot \frac{n-1}{n} \cdot \frac{n-2}{n} \cdot \frac{3}{n} + \frac{1}{n} \cdot \frac{n-1}{n} \cdot \frac{2}{n} \cdot \frac{2}{n} \cdot \frac{n-2}{n} + \\ & + \frac{1}{n} \cdot \frac{n-1}{n} \cdot \frac{2}{n} \cdot \frac{n-2}{n} \cdot \frac{3}{n} + \frac{1}{n} \cdot \frac{n-1}{n} \cdot \frac{n-2}{n} \cdot \frac{3}{n} \cdot \frac{3}{n} + \\ & + \frac{n-1}{n} \cdot \frac{2}{n} \cdot \frac{2}{n} \cdot \frac{2}{n} \cdot \frac{n-2}{n} + \frac{n-1}{n} \cdot \frac{2}{n} \cdot \frac{2}{n} \cdot \frac{n-2}{n} \cdot \frac{3}{n} + \\ & + \frac{n-1}{n} \cdot \frac{2}{n} \cdot \frac{n-2}{n} \cdot \frac{3}{n} \cdot \frac{3}{n} + \frac{n-1}{n} \cdot \frac{n-2}{n} \cdot \frac{3}{n} \cdot \frac{3}{n} \cdot \frac{3}{n} \end{aligned}$$

Что происходит выше?

Для любого количества заказов больше одного, вероятность встретить больше либо одного имени равна единице. Каждое новое имя добавляет в вероятность $\frac{n-i}{n}$, где i - номер нового имени (первое имя считаем за 0). Но так как за a заказов может быть $k < a$ имен, необходимо учесть тот факт, что одни и те же таксисты могут приехать несколько раз. Возьмем базовую формулу для k имен:

$$\frac{n-1}{n} \cdot \frac{n-2}{n} \cdot \dots \cdot \frac{n-k+1}{n}$$

Если имен $k < a$ заказов, необходимо просуммировать такие "базовые" формулы, добавляя между ними множители $\frac{i}{n}$, где i определяется позицией множителя (интерпретация этого множителя состоит в количестве повторений i имен к моменту данного заказа). Т.е, если $\frac{i}{n}$ стоит перед множителем $\frac{n-1}{n}$, (тогда $i = 1$), то это означает, что самый первый водитель приехал два раза, а затем уже приехал второй и т.д. Если же множитель $\frac{i}{n}$ (тогда $i = 2$) стоит перед $\frac{n-2}{n}$, то это означает, что на третий заказ приехал либо первый, либо второй таксист. Всего множителей вида $\frac{i}{n}$ будет $a - k$ штук. А количество слагаемых в формуле вероятности: $C_{k+a-k-1}^{a-k} = C_{a-1}^{a-k}$, (сочетания с возвращениями). Почему сочетания?

По-факту мы пытаемся разными способами разложить $a - k$ слагаемых вида $\frac{i}{n}$ по k "корзинам".

У $\mathbb{P}_{10}(X=6)$ можно вынести за скобки $\frac{(n-1) \cdot (n-2) \cdot \dots \cdot (n-5)}{n^9}$, в скобках же останется сумма чисел, находящихся в числителях каждого из слагаемых (всего слагаемых будет $C_9^4 = 126$). можно разделить полученную вероятность на эту константу, тогда останется только функция от n , в силу монотонного характера преобразования максимум у нее не поменяется. Тогда монотонно-преобразованная функция правдоподобия:

$$L = \frac{(n-1) \cdot (n-2) \cdot (n-3) \cdot (n-4) \cdot (n-5)}{n^9}$$

$$\ln L = \ln(n-1) + \ln(n-2) + \ln(n-3) + \ln(n-4) + \ln(n-5) - 9 \ln n$$

Опять таки, очень сложно аналитически найти решение для такой задачи, поэтому будем перебирать все возможные целые n .

In [121...

```
def likelihood_6(n):
    L = 1 / (n ** 4)
    for i in range(6):
```

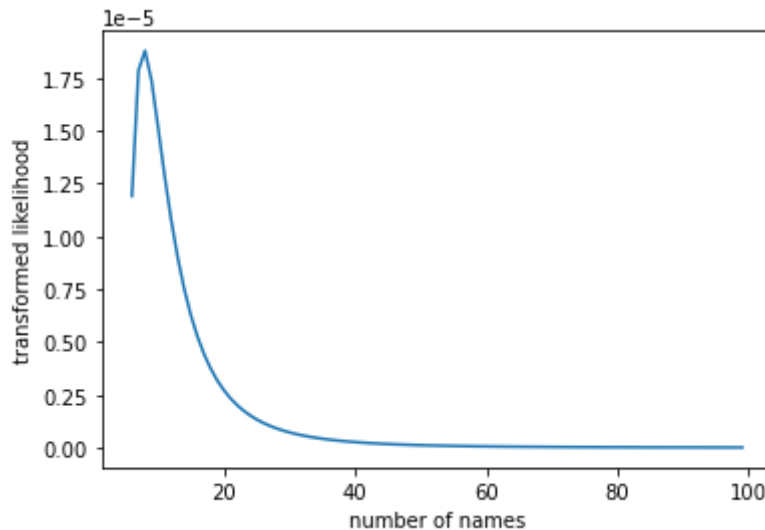
```
L *= (n-i) / n
return L
```

In [122...

```
ns = np.arange(6,100)
```

In [122...

```
plt.plot(ns, likelihood_6(ns));
plt.xlabel('number of names')
plt.ylabel('transformed likelihood');
```



Аналогично задаче 1, переберем ручками возможные n , чтобы найти такое, при котором функция правдоподобия принимает максимальное значение.

In [122...

```
max_L = -1
arg_max_L = -1
for n in range(6, 10000):
    if likelihood_6(n) > max_L:
        max_L = likelihood_6(n)
        arg_max_L = n
```

Получаем

In [122...

```
arg_max_L
```

Out[122...

```
8
```

Следовательно, $\hat{n}_{ml} = 8$

(6) Я написал функцию, которая считает количества имен таксистов для 10 поездок для заданного n

In [139...

```
from itertools import combinations, combinations_with_replacement
```

In [145...

```
term_coefs = [] # этот кусок по-факту является частью функции, считающей матожидания
for i in range(1, 11): #долго, поэтому я вытащил ее, и просто вставил полученные коэф
    base = np.arange(1, i + 1)
    comb = set()
    coef_sum = 0
    to_combinations = np.arange(1, 10 - i + 1)
    to_combinations = np.append(to_combinations, np.zeros(10 - to_combinations.shape
```

```

for j in combinations_with_replacement(to_combinations, i):
    if sum(j) == 10 - i:
        comb.add(j)
for a in comb:
    if type(a) == int:
        a = [a]
    permuted = np.array(list(set(permutations(a, i))))
    coefs = np.prod(base ** permuted, axis = 1)
    coef_sum += coefs.sum()
term_coefs.append(coef_sum)
term_coefs

```

Out[145...] [1.0, 511.0, 9330.0, 34105.0, 42525.0, 22827.0, 5880.0, 750.0, 45.0, 1.0]

```

In [164...]
def expectation_10(n):
    expect = 0
    expect = 0
    term_coefs = [1.0, 511.0, 9330.0, 34105.0, 42525.0, 22827.0, 5880.0, 750.0, 45.0, 1.0]
    for i in range(1, 11):
        base_P = 1 / (n ** (10 - i))
        for k in range(1, i + 1):
            base_P *= (n - k + 1) / n
        expect += i * term_coefs[i - 1] * base_P
    return expect

```

In [164...] expectation_10(20)

Out[164...] 8.025261215232423

In [164...] expectation_10(16)

Out[164...] 7.608632399220369

```

In [165...]
expectations = []
for n in tqdm(range(1, 100)):
    x = expectation_10(n)
    expectations.append(x)

```

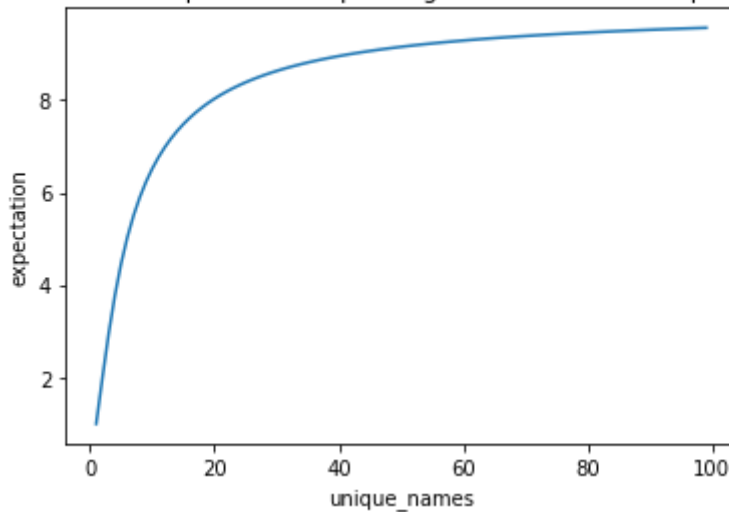
100%|██████████| 99/99 [00:00<?, ?it/s]

```

In [145...]
plt.plot(list(range(1, 100)), expectations)
plt.xlabel('unique_names')
plt.ylabel('expectation')
plt.title('Mathematical Expectation Depending on the Nuber of Unique Names');

```

Mathematical Expectation Depending on the Nuber of Unique Names



Найдем оценку \hat{n} методом моментов.

У нас есть одно наблюдение, следовательно выборочное среднее: $\bar{X} = X_1 = 6$. Метод моментов:

$$\mathbb{E}[X]_{n=\hat{n}} = \bar{X}$$

Переберем различные целочисленные \hat{n} , и возьмем такое, при котором отклонение среднего от матожидания минимально. Матожидание монотонно возрастает.

```
In [146...
mse_all = []
for n in range(1, 30):
    mse = (6 - expectation_10(n)) ** 2
    mse_all.append(mse)
```

```
In [146...
np.argmin(np.array(mse_all)) + 1 #неправляем на индексацию с нуля
```

Out[146... 8

Получается оценка методом моментов:

$$\hat{n}_{mm} = 8$$

(в)

```
In [148...
names = np.arange(1, 21)
names.shape
```

Out[148... (20,)

```
In [148...
np.random.seed(767)
samples = np.random.choice(names, size = (10000, 10))
```

надо написать функцию правдоподобия для разных $n_{observed}$. Опять же, функция ниже является монотонным преобразованием действительной функции правдоподобия.

```
In [150...
def likelihood_names(names, n):
    base_P = 1 / (n ** (10 - names))
```

```

for k in range(1, names + 1):
    base_P *= (n - k + 1)/n
return base_P

```

In [153...

```

ml_names = []

for sample in tqdm(samples):
    observed_n = np.unique(sample).shape[0]
    if observed_n == 10: # в этом случае оценка стремится к бесконечности
        ml_names.append(100)
        continue
    max_L = -1
    arg_max_L = -1
    for n in range(observed_n, 1000):
        L = likelihood_names(observed_n, n)
        if L > max_L:
            max_L = L
            arg_max_L = n
    ml_names.append(arg_max_L)

```

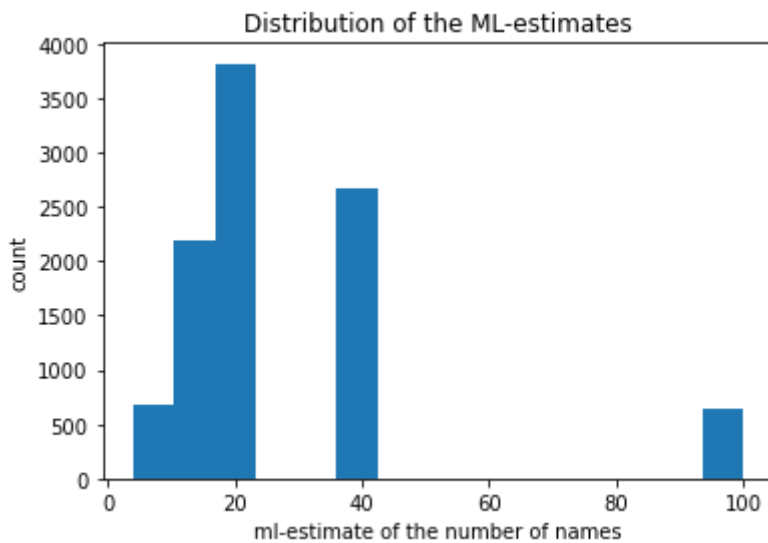
100%|██████████| 10000/10000 [00:14<00:00, 709.83it/s]

In [165...

```

plt.hist(ml_names, bins = 15);
plt.xlabel('ml-estimate of the number of names')
plt.ylabel('count')
plt.title('Distribution of the ML-estimates');

```



In [154...

```
ml_names = np.array(ml_names)
```

Оценка смещения ml-оценки

In [154...

```
ml_names.mean() - 20
```

Out[154...

8.0019

Оценка дисперсии ml-оценки

In [155...

```
ml_names.var(ddof = 1)
```

Out[155...] 499.7424706370637

Оценка среднеквадратичного отклонения ml-оценки

In [155...] `np.sum((ml_names - 20) ** 2) / 10000`

Out[155...] 563.7229

Теперь для моментов

In [164...] `samples.shape`

Out[164...] (10000, 10)

In [165...] `mm_names = []`

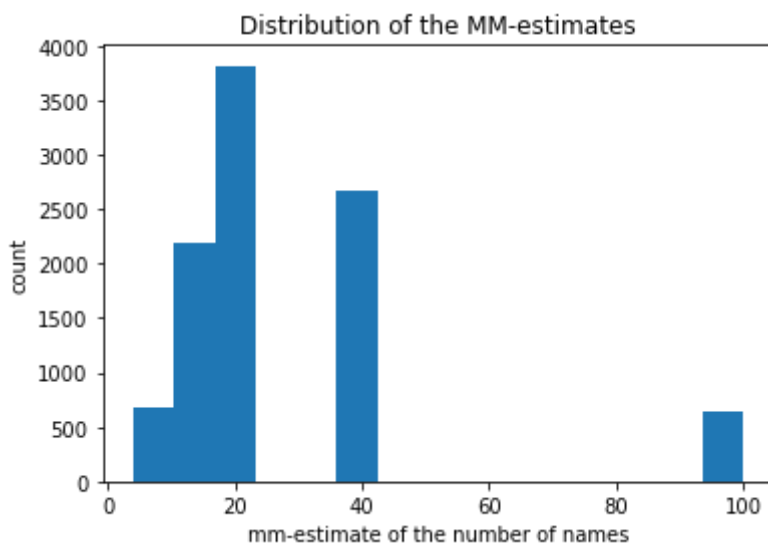
```

for sample in tqdm(samples):
    observed_n = np.unique(sample).shape[0]
    if observed_n == 10: # в этом случае оценка стремится к бесконечности
        mm_names.append(100)
        continue
    mse_list = []
    for i in range(1, 101):
        mse = (expectation_10(i) - observed_n) ** 2
        mse_list.append(mse)
        if i == 1:
            continue
        if mse_list[i - 1] > mse_list[i - 2]:
            break
    mm_n = np.array(mse_list).argmin() + 1
    mm_names.append(mm_n)

```

100%|██████████| 10000/10000 [00:03<00:00, 3320.23it/s]

In [165...] `plt.hist(mm_names, bins = 15)`
`plt.xlabel('mm-estimate of the number of names')`
`plt.ylabel('count')`
`plt.title('Distribution of the MM-estimates');`



In [165...] `mm_names = np.array(mm_names)`

Оценка смещения для mm-оценки

```
In [165... mm_names.mean() - 20
```

```
Out[165... 8.392399999999999
```

Оценка дисперсии mm-оценки

```
In [165... mm_names.var(ddof = 1)
```

```
Out[165... 492.70009224922495
```

Оценка среднеквадратического отклонения mm-оценки

```
In [165... np.sum((mm_names - 20) ** 2) / 10000
```

```
Out[165... 563.0832
```

Задачи 4,5,6

```
In [167... def vowler(string):
    vowels = set('аеёиоуыя')
    if string[0].lower() in vowels:
        return 'vowel'
    else:
        return 'consonant'
```

Задача 4

```
In [167... data = pd.read_excel('exam_grades.xlsx')
```

```
In [167... data['first_letter'] = data['last_name'].apply(vowler)
```

```
In [167... sample_v = data[data['first_letter'] == 'vowel']['grade'].values
n_v = sample_v.shape[0]
var_v = sample_v.var(ddof = 1)
mean_v = sample_v.mean()

sample_c = data[data['first_letter'] == 'consonant']['grade'].values
n_c = sample_c.shape[0]
var_c = sample_c.var(ddof = 1)
mean_c = sample_c.mean()
```

Тестируем следующее:

$$H_0 : \mu_c = \mu_v$$

$$H_1 : \mu_c \neq \mu_v$$

(a) Welch's test. Предполагая, что выборки нормальные

$$\frac{\hat{\mu}_c - \hat{\mu}_v - (\mu_c - \mu_v)}{\sqrt{\frac{\hat{\sigma}_c^2}{n_c} + \frac{\hat{\sigma}_v^2}{n_v}}} \stackrel{a.}{\sim} t_d$$

Количество степеней свободы можно оценить как

$$\hat{d} = \frac{\left(\frac{\hat{\sigma}_c^2}{n_c} + \frac{\hat{\sigma}_v^2}{n_v}\right)^2}{\frac{\hat{\sigma}_c^4}{(n_c-1)n_c^2} + \frac{\hat{\sigma}_v^4}{(n_v-1)n_v^2}}$$

вычислим \hat{d}

```
In [167... d = (var_c / n_c + var_v / n_v) / (var_c ** 2 / ((n_c - 1) * (n_c ** 2)) + var_v ** 2 / d
```

```
Out[167... 40.04531907940249
```

$$t_{obs} = \frac{\hat{\mu}_c - \hat{\mu}_v}{\sqrt{\frac{\hat{\sigma}_c^2}{n_c} + \frac{\hat{\sigma}_v^2}{n_v}}}$$

```
In [167... t_obs = (mean_c - mean_v) / np.sqrt(var_c / n_c + var_v / n_v)
t_obs
```

```
Out[167... 0.8519661870595602
```

```
In [167... t_crit = stats.t.ppf(0.975, df = d)
t_crit
```

```
Out[167... 2.0210041370846357
```

$|t_{obs}| < t_{crit}$. Значит нет основания отвергать H_0

```
In [167... stats.ttest_ind(sample_c, sample_v, equal_var=False) # ручками получилось правильно
```

```
Out[167... Ttest_indResult(statistic=0.8519661870595602, pvalue=0.3974027153843839)
```

```
In [167... p_value = 2 * (1 - stats.t.cdf(t_obs, df = d))
p_value
```

```
Out[167... 0.3992993925647852
```

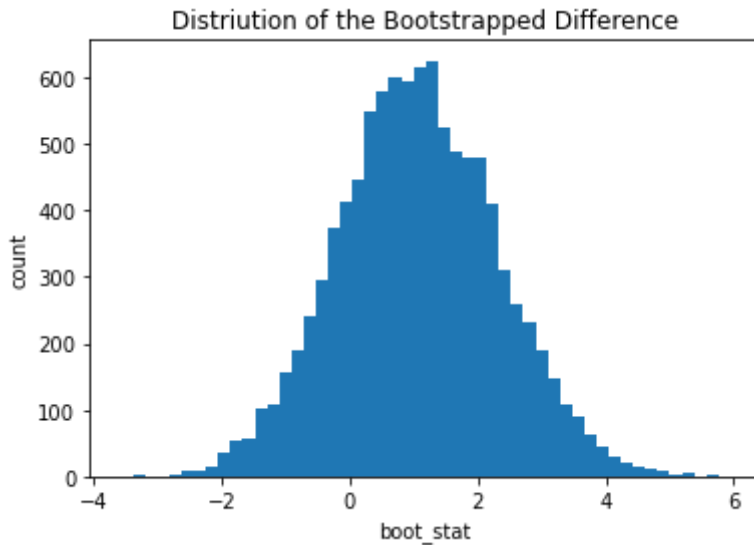
(6) наивный бутстрэп

In [167...

```
np.random.seed(6546)
boot_stats_naive = list()
for i in range(10000):
    boot_sample_c = np.random.choice(sample_c, size = sample_c.shape[0])
    boot_sample_v = np.random.choice(sample_v, size = sample_v.shape[0])
    boot_stat = boot_sample_c.mean() - boot_sample_v.mean()
    boot_stats_naive.append(boot_stat)
```

In [168...

```
plt.hist(boot_stats_naive, bins = 50);
plt.xlabel('boot_stat')
plt.ylabel('count')
plt.title('Distriution of the Bootstrapped Difference');
```



In [168...

```
q_left_naive = np.percentile(boot_stats_naive, 2.5)
q_right_naive = np.percentile(boot_stats_naive, 97.5)
```

In [168...

```
observed = mean_c - mean_v
observed
```

Out[168...

1.0782433114588574

Получаем наивный бутстрап- интервал

In [168...

```
ci = [q_left_naive, q_right_naive]
ci
```

Out[168...

[-1.3480421143722485, 3.585934232350184]

0 входит в интервал, следовательно, нет оснований отвергать H_0

р-значение я вычислил, посчитав доли бутстрап наблюдений больше и меньше 0, и взяв минимум из них двух, умножил на два (т.к. гипотеза двусторонняя)

In [168...

```
p_value_naive = 2 * min(np.sum((np.array(boot_stats_naive) > 0))/ 10000, np.sum((np.
p_value_naive
```

Out[168...

0.3918

(в) бутстрап t-статистики

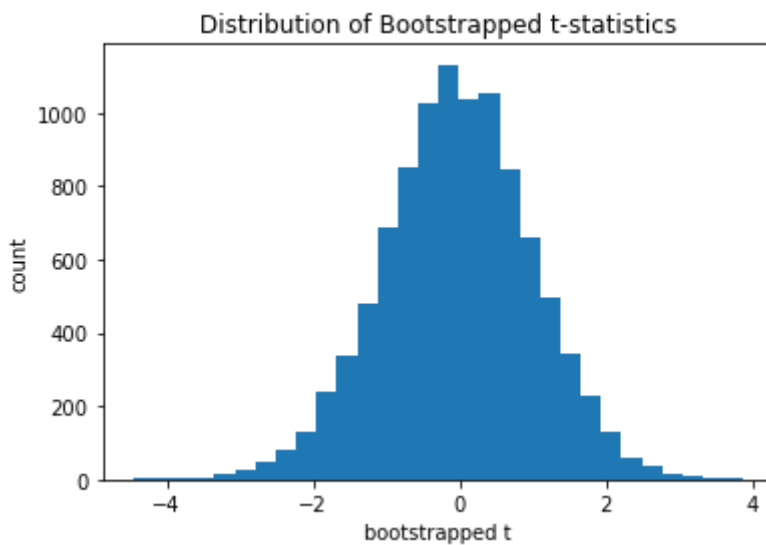
```
In [106... t_observed = (mean_c - mean_v) / np.sqrt(var_c / n_c + var_v / n_v)
t_observed
```

```
Out[106... 0.8519661870595602
```

```
In [108... np.random.seed(2534245)
boot_stats_t = []
for i in range(10000):
    boot_sample_c = np.random.choice(sample_c, size = sample_c.shape[0])
    boot_sample_v = np.random.choice(sample_v, size = sample_v.shape[0])
    boot_t = (boot_sample_c.mean() - boot_sample_v.mean() - (mean_c - mean_v)) / (np.s
    boot_stats_t.append(boot_t)
```

```
In [ ]:
```

```
In [168... plt.hist(boot_stats_t, bins = 30);
plt.title('Distribution of Bootstrapped t-statistics')
plt.xlabel('bootstrapped t')
plt.ylabel('count');
```



```
In [109... q_left_t = np.percentile(boot_stats_t, 2.5)
q_right_t = np.percentile(boot_stats_t, 97.5)
```

По распределению бутстрапированных t-статистик получил квантили:

```
In [109... q_left_t, q_right_t
```

```
Out[109... (-2.1191775645848585, 1.9353991643062767)
```

Наблюдаемая t-статистика:

```
In [109... t_observed
```

```
Out[109... 0.8519661870595602
```

t-observed находится внутри 'интервала', поэтому нет оснований отвергать H_0 . p-значение:

```
In [115... p_value = 2 * min(np.sum((boot_stats_t > t_observed))/ 10000, np.sum((boot_stats_t <
p_value
```

```
Out[115... 0.3804
```

(r) Перестановочный тест

```
In [113... from itertools import permutations
```

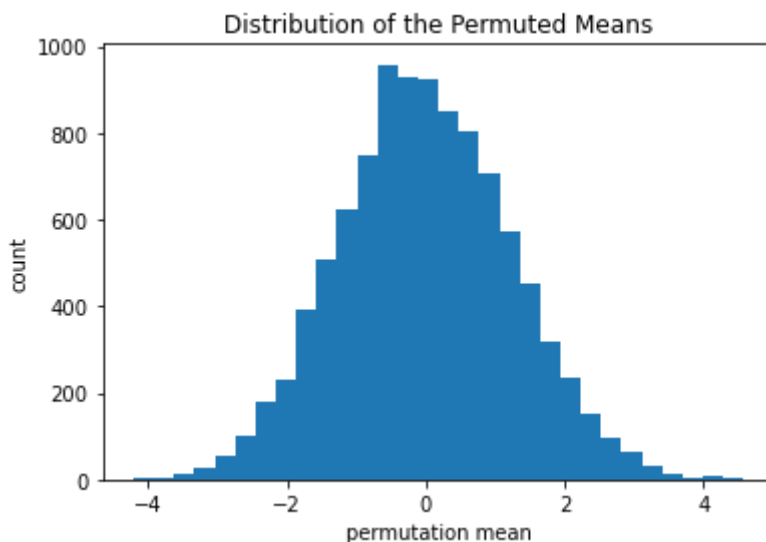
```
In [113... pool = np.append(sample_c, sample_v)
```

Будем делать новые выборки размера n_c и n_v из объединенных сэмплов.

```
In [114... permutation_means = []
for i in tqdm(range(10000)):
    permuted = np.random.permutation(pool)
    permuted_c = permuted[:n_c]
    permuted_v = permuted[n_c:]
    permuted_mean = permuted_c.mean() - permuted_v.mean()
    permutation_means.append(permuted_mean)
```

```
100%|██████████| 10000/10000 [00:00<00:00, 40597.32it/s]
```

```
In [114... plt.hist(permutation_means, bins = 30);
plt.xlabel('permutation mean')
plt.ylabel('count')
plt.title('Distribution of the Permuted Means');
```



Перестановочный интервал разниц матожиданий:

```
In [ ]: permutation_means = np.array(permutation_means)
```

```
In [169... np.percentile(permutation_means, 2.5), np.percentile(permutation_means, 97.5)
```

```
Out[169... (-2.3693661210067045, 2.4429220451431455)
```

In [170...

observed

Out[170...

1.0782433114588574

In [169...

```
p_value = 2 * min(np.sum(permutation_means > observed) / 10000, np.sum(permutation_me
p_value
```

Out[169...

0.3772

Уровень значимости 0.05 меньше р-значения, следовательно, нет оснований отвергать H_0

Задача 5

In [158...

```
def vowelizer(string):
    vowels = set('аеёиоуыэюя')
    if string[0].lower() in vowels:
        return 'vowel'
    else:
        return 'consonant'

def medianer(x):
    median = data['grade'].median()
    if x > median:
        return 'more_than_median'
    else:
        return 'less_than_median'
```

In [158...

```
data = pd.read_excel('exam_grades.xlsx')
```

In [158...

```
data['first_letter'] = data['last_name'].apply(vowelizer)
data['median'] = data['grade'].apply(medianer)
```

In [158...

```
contingency_table = pd.crosstab(index = data['first_letter'], columns=data['median'])
```

In [158...

contingency_table

Out[158...

	median	less_than_median	more_than_median
first_letter			
consonant		138	145
vowel		28	21

In [158...

```
p_cons = contingency_table.iloc[0,1] / (contingency_table.iloc[0,0] + contingency_ta
p_vow = contingency_table.iloc[1,1] / (contingency_table.iloc[1,0] + contingency_tab
nc = (contingency_table.iloc[0,0] + contingency_table.iloc[0,1])
nv = (contingency_table.iloc[1,0] + contingency_table.iloc[1,1])
```

(6) начну с гипотезы

$$H_0 : p_c = p_v$$

$$H_1 : p_c \neq p_v$$

$$Z_{obs} = \frac{\hat{p}_c - \hat{p}_v}{\sqrt{\frac{\hat{p}_c(1-\hat{p}_c)}{n_c} + \frac{\hat{p}_v(1-\hat{p}_v)}{n_v}}} \stackrel{a.}{\sim} N(0, 1)$$

In [159...

```
z_obs = (p_cons-p_vow)/np.sqrt(p_cons*(1-p_cons)/nc + p_vow * (1-p_vow)/nv)
z_obs
```

Out[159...

1.092714086706707

Критическое значение на уровне значимости 0.95:

In [159...

```
z_crit = stats.norm().ppf(0.975)
z_crit
```

Out[159...

1.959963984540054

$|Z_{obs}| < Z_{crit}$, следовательно нулевая гипотеза не отвергается

In [159...

```
p_value = 2*(1 - stats.norm().cdf(z_obs))
p_value
```

Out[159...

0.2745193528742256

интервал

отношение вероятностей приблизим в точке $(\hat{p}_v, \hat{p}_c) = (p_v, p_c)$

$$h(\hat{p}_v, \hat{p}_c) = \frac{\hat{p}_v}{\hat{p}_c}$$

$$h(\hat{p}_v, \hat{p}_c) \approx \frac{p_v}{p_c} + \frac{1}{p_c}(\hat{p}_v - p_v) - \frac{p_v}{p_c^2}(\hat{p}_c - p_c)$$

$$Var(h(\hat{p}_v, \hat{p}_c)) \approx Var\left(\frac{p_v}{p_c} + \frac{1}{p_c}(\hat{p}_v - p_v) - \frac{p_v}{p_c^2}(\hat{p}_c - p_c)\right) =$$

$$\frac{1}{p_c^2}Var(\hat{p}_v) + \frac{p_v^2}{p_c^4}Var(\hat{p}_c) = \frac{p_v(1-p_v)}{p_c^2 n_v} + \frac{p_v^2(1-p_c)}{p_c^3 n_c}$$

Тогда интервал будет иметь вид

$$\frac{\hat{p}_v}{\hat{p}_c} - 1.96 \cdot \sqrt{\frac{p_v(1-p_v)}{p_c^2 n_v} + \frac{p_v^2(1-p_c)}{p_c^3 n_c}} < \frac{p_v}{p_c} < \frac{\hat{p}_v}{\hat{p}_c} + 1.96 \cdot \sqrt{\frac{p_v(1-p_v)}{p_c^2 n_v} + \frac{p_v^2(1-p_c)}{p_c^3 n_c}}$$

Численно:

In [161...

```
p_ratio = p_vow/p_cons
p_var_ratio = p_vow * (1-p_vow)/(n_v * p_cons **2) + (1-p_cons) * p_vow ** 2/(n_c *
```

In [161...

```
p_ratio - z_crit * np.sqrt(p_var_ratio), p_ratio + z_crit * np.sqrt(p_var_ratio)
```

Out[161... (0.5497946344819109, 1.1231117694589758)

Можно проверить гипотезу и через интервал

Гипотеза

$$H_0 : \frac{p_v}{p_c} = 1$$

$$H_1 : \frac{p_v}{p_c} \neq 1$$

Наблюдаемое значение статистики:

```
In [162...
z_obs = (p_ratio - 1)/np.sqrt(p_var_ratio)
z_obs
```

Out[162... -1.118214734459665

```
In [162...
p_value = stats.norm.cdf(z_obs) * 2
p_value
```

Out[162... 0.26347529277764214

р-значение больше 0.05, следовательно, нет оснований отвергать гипотезу H_0 на уровне значимости 5%.

(a) Что такое шансы?

Пусть X_i (Y_i) - бернуллевская случайная величина, которая принимает значение 1, если студент с фамилией на согласную (гласную) получает оценку выше медианы, и 0 - если ниже. Тогда шансы хорошо написать экзамен:

$$\hat{h}_c = \frac{\sum X_i}{\sum (1 - X_i)} = \frac{\sum X_i}{n_c - \sum X_i} = \frac{n_c}{n_c - \sum X_i} - 1 = \frac{1}{1 - \hat{p}_c} - 1$$

$$\hat{h}_v = \frac{1}{1 - \hat{p}_v} - 1$$

$$\frac{\hat{h}_v}{\hat{h}_c} = \frac{\frac{1}{1 - \hat{p}_v} - 1}{\frac{1}{1 - \hat{p}_c} - 1} = \frac{\frac{\hat{p}_v}{1 - \hat{p}_v}}{\frac{\hat{p}_c}{1 - \hat{p}_c}} = \frac{\hat{p}_v(1 - \hat{p}_c)}{\hat{p}_c(1 - \hat{p}_v)}$$

Оценки \hat{p}_c и \hat{p}_v состоятельны и асимптотически нормальны. Воспользуемся дельта методом, чтобы приблизительно оценить отношение шансов в окрестности точки $(\hat{p}_v, \hat{p}_c) = (p_v, p_c)$. В первом приближении:

$$h(\hat{p}_v, \hat{p}_c) \approx h(p_v, p_c) + h'_{p_v}(p_v, p_c)(\hat{p}_v - p_v) + h'_{p_c}(p_v, p_c)(\hat{p}_c - p_c)$$

$$h(\hat{p}_v, \hat{p}_c) \approx \frac{p_v(1 - p_c)}{p_c(1 - p_v)} + \frac{1 - p_c}{(1 - p_v)^2 p_c}(\hat{p}_v - p_v) + \frac{p_v}{p_c^2(1 - p_v)}(\hat{p}_c - p_c)$$

Тогда (выборки для фамилий, начинающихся на гласную и фамилий, начинающихся на согласную независимы)

$$\begin{aligned}
 \text{Var}(h(\hat{p}_v, p_c)) &\approx \text{Var}\left(\frac{p_v(1-p_c)}{p_c(1-p_v)} + \frac{1-p_c}{(1-p_v)^2 p_c}(\hat{p}_v - p_v) + \frac{p_v}{p_c^2(1-p_v)}(\hat{p}_c - p_c)\right) = \\
 &= \frac{(1-p_c)^2}{(1-p_v)^4 p_c^2} \text{Var}(\hat{p}_v) + \frac{p_v^2}{p_c^4(1-p_v)^2} \text{Var}(\hat{p}_c) = \\
 &= \frac{(1-p_c)^2}{(1-p_v)^4 p_c^2} \cdot \frac{p_v(1-p_v)}{n_v} + \frac{p_v^2}{p_c^4(1-p_v)^2} \cdot \frac{p_c(1-p_c)}{n_c} = \\
 &= \frac{(1-p_c)^2 p_v}{(1-p_v)^3 p_c^2 n_v} + \frac{p_v^2(1-p_c)}{p_c^3(1-p_v)^2 n_c}
 \end{aligned}$$

Получается, что

$$\frac{\hat{c}h_v}{\hat{c}h_c} \underset{a.}{\sim} N\left[\frac{p_v(1-p_c)}{p_c(1-p_v)}, \frac{(1-p_c)^2 p_v}{(1-p_v)^3 p_c^2 n_v} + \frac{p_v^2(1-p_c)}{p_c^3(1-p_v)^2 n_c}\right]$$

Тогда доверительный интервал:

$$\frac{\hat{p}_v(1-\hat{p}_c)}{\hat{p}_c(1-\hat{p}_v)} - 1.96 \cdot \sqrt{\frac{(1-\hat{p}_c)^2 \hat{p}_v}{(1-\hat{p}_v)^3 \hat{p}_c^2 n_v} + \frac{\hat{p}_v^2(1-\hat{p}_c)}{\hat{p}_c^3(1-\hat{p}_v)^2 n_c}} < \frac{ch_v}{ch_c} < \frac{\hat{p}_v(1-\hat{p}_c)}{\hat{p}_c(1-\hat{p}_v)} + 1.96 \cdot \sqrt{\frac{(1-\hat{p}_c)^2 \hat{p}_v}{(1-\hat{p}_v)^3 \hat{p}_c^2 n_v} + \frac{\hat{p}_v^2(1-\hat{p}_c)}{\hat{p}_c^3(1-\hat{p}_v)^2 n_c}}$$

Вычислим:

Наблюдаемое отношение шансов

In [159...

```
ch_ratio = p_vow * (1 - p_cons) / (p_cons * (1 - p_vow))
ch_ratio
```

Out[159...

```
0.713793103448276
```

In [160...

```
var_ratio = p_vow * (1 - p_cons) ** 2 / (n_v * (p_cons ** 2) * (1 - p_vow) ** 3) + p_
```

Получаем интервал

In [160...

```
ch_ratio - z_crit * np.sqrt(var_ratio), ch_ratio + z_crit * np.sqrt(var_ratio)
```

Out[160...

```
(0.27700593497628884, 1.1505802719202631)
```

$$H_0 : \frac{ch_v}{ch_c} = 1$$

$$H_1 : \frac{ch_v}{ch_c} \neq 1$$

$$Z_{obs} = \frac{\frac{\hat{p}_v(1-\hat{p}_c)}{\hat{p}_c(1-\hat{p}_v)} - 1}{\sqrt{\frac{(1-\hat{p}_c)^2 \hat{p}_v}{(1-\hat{p}_v)^3 \hat{p}_c^2 n_v} + \frac{\hat{p}_v^2(1-\hat{p}_c)}{\hat{p}_c^3(1-\hat{p}_v)^2 n_c}}}$$

Наблюдаемая статистика


```
In [160...
z_obs = (ch_ratio - 1) / np.sqrt(var_ratio)
z_obs
```

```
Out[160... -1.284275843841178
```

p-значение

```
In [160...
p_value = 2 * stats.norm.cdf(z_obs)
p_value
```

```
Out[160... 0.19904545585927402
```

p-значение больше 0.05, следовательно нет оснований отвергать нулевую гипотезу на уровне значимости 5%.

(в) БУТСТРЕП

```
In [111...
obs_ratio = ch_v / ch_c
```

```
In [111...
ch_c, ch_v
```

```
Out[111... (1.0507246376811592, 0.75)
```

```
In [111...
obs_ratio
```

```
Out[111... 0.713793103448276
```

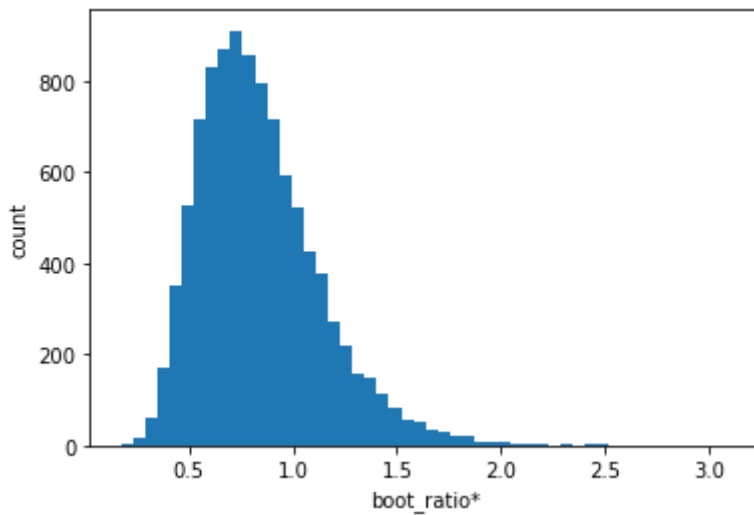
```
In [111...
n_boot = 10000
```

наивный бутстрап: будем генерировать $\left(\frac{\hat{ch}_v}{\hat{ch}_c}\right)^*$

```
In [111...
np.random.seed(246)
boot_ratios = list()
for i in tqdm(range(n_boot)):
    boot_sample_idx = np.random.choice(data.index.values, size = 332)
    sample = data[['grade', 'first_letter']].iloc[boot_sample_idx]
    sample['md'] = sample['grade'].apply(lambda x: 'more_than_median' if x > sample.
    table = pd.crosstab(index = sample['first_letter'], columns=sample['md'])
    p_cons = table.iloc[0,1] / (table.iloc[0,0] + table.iloc[0,1])
    p_vow = table.iloc[1,1] / (table.iloc[1,0] + table.iloc[1,1])
    ch_c = 1 / (1-p_cons) - 1
    ch_v = 1 / (1-p_vow) - 1
    boot_ratio = ch_v / ch_c
    boot_ratios.append(boot_ratio)
```

100%|██████████| 10000/10000 [02:20<00:00, 71.28it/s]

```
In [112...
plt.hist(boot_ratios, bins = 50);
plt.xlabel('boot_ratio*');
plt.ylabel('count');
```



```
In [112... boot_ratios = np.array(boot_ratios)
```

```
In [112... q_left = np.percentile(boot_ratios, 2.5)
q_right = np.percentile(boot_ratios, 97.5)
```

Получили квантили

```
In [112... q_right
```

```
Out[112... 1.5401953418482344
```

```
In [112... q_left
```

```
Out[112... 0.40539847539847534
```

Получаем

$$q_{left}(R^*) < \frac{ch_v}{ch_c} < q_{right}(R^*)$$

```
In [112... obs_ratio
```

```
Out[112... 0.713793103448276
```

Вычислим:

```
In [112... q_left, q_right
```

```
Out[112... (0.40539847539847534, 1.5401953418482344)
```

Гипотеза:

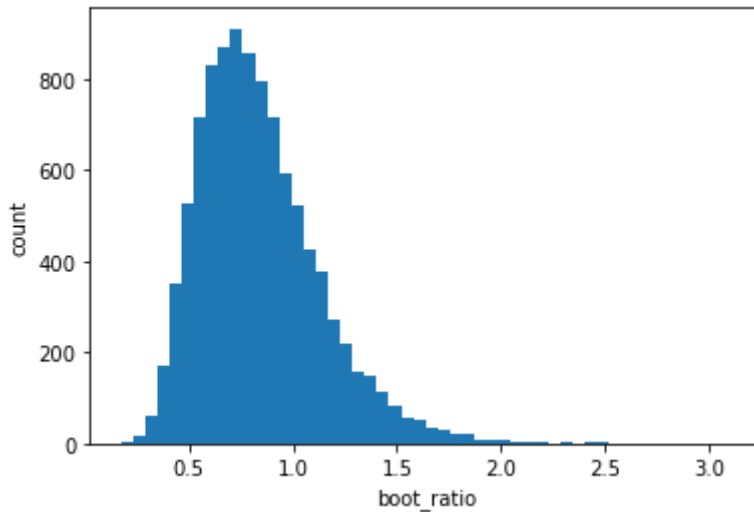
$$H_0 : ch_c = ch_v$$

$$H_1 : ch_c \neq ch_v$$

Получаем $\frac{ch_c}{ch_v} = 1$, входит в интервал, нет оснований отвергать H_0 .

In [112...

```
plt.hist(boot_ratios ,bins = 50);
plt.xlabel('boot_ratio')
plt.ylabel('count');
```



Двустороннее p-value

In [170...

```
p_value = 2 * min(np.sum(boot_ratios < 1)/10000, np.sum(boot_ratios >= 1)/10000)
p_value
```

Out[170...

0.504

Задача 6

In [162...

```
data['len_name'] = data['last_name'].apply(lambda x: len(x))
```

$$\mathbb{E}[Y_i] = \beta F_i$$

По-факту здесь

$$\mathbb{E}[Y_i|F_i] = \beta F_i$$

Возьмем математическое ожидание от левой и правой частей

$$\mathbb{E}[\mathbb{E}[Y_i|F_i]] = \beta \cdot \mathbb{E}[F_i]$$

$$\mathbb{E}[Y_i] = \beta \cdot \mathbb{E}[F_i]$$

Оценивая $\mathbb{E}[Y_i]$ и $\mathbb{E}[F_i]$ методом моментов с помощью первого начального момента, получаем

$$\mathbb{E}[Y_i] = \bar{Y}, \mathbb{E}[F_i] = \bar{F}$$

Тогда

$$\bar{Y} = \beta \bar{F}$$

$$\hat{\beta} = \frac{\bar{Y}}{\bar{F}}$$

Подставляем данные и получаем оценку параметра бета:

In [162...

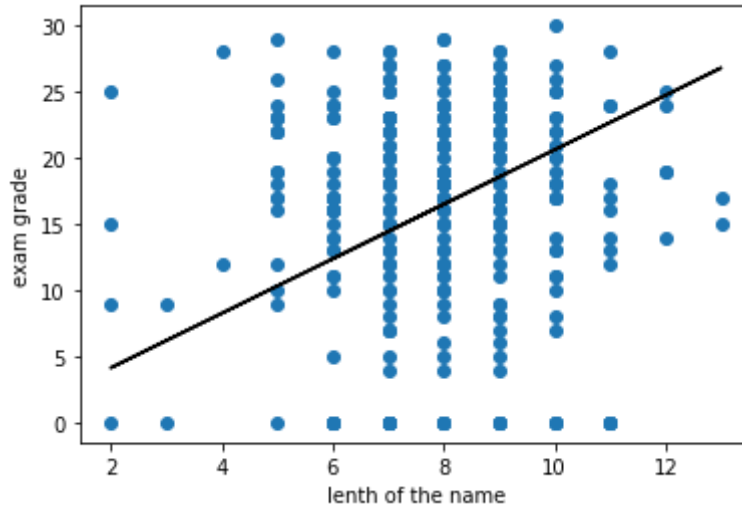
```
beta = data['grade'].mean() / data['len_name'].mean()
beta
```

Out[162...

2.0613026819923372

In [163...

```
plt.plot(data['len_name'], data['len_name'] * beta, color = 'black')
plt.scatter(data['len_name'], data['grade']);
plt.xlabel('length of the name')
plt.ylabel('exam grade');
```



In [163...

```
data[['grade', 'len_name']].corr()
```

Out[163...

	grade	len_name
grade	1.000000	0.025328
len_name	0.025328	1.000000

Получается выборочная корреляция:

In [163...

```
p = data[['grade', 'len_name']].corr().iloc[0,1]
p
```

Out[163...

0.025328052669147543

(6)

$$H_0 : p = 0$$

$$H_1 : p \neq 0$$

Будем переставлять Y_i

In [163...

```
np.random.seed(3456436)
p_permuted = []
for i in tqdm(range(10000)):
    y_permuted = np.random.permutation(data.grade)
    p_new = np.corrcoef(y_permuted, data.len_name)[0,1]
    p_permuted.append(p_new)
```

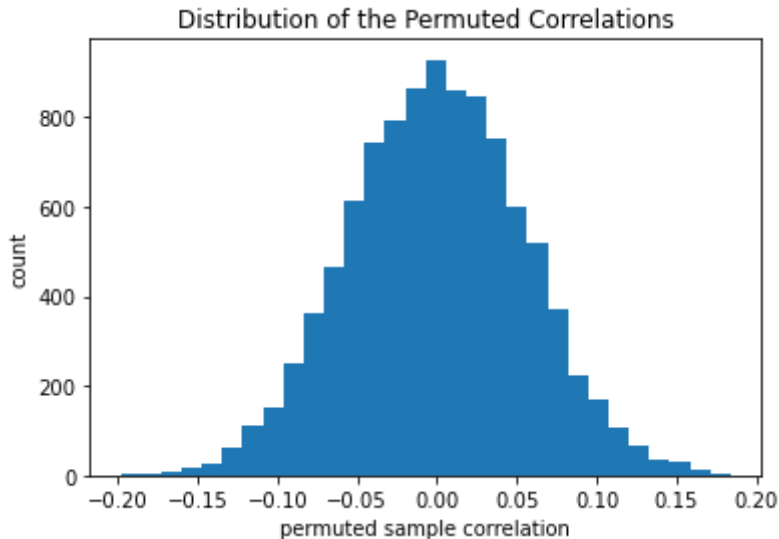
100%|██████████| 10000/10000 [00:01<00:00, 8065.92it/s]

In [163...

```
p_permuted = np.array(p_permuted)
```

In [163...

```
plt.hist(p_permuted, bins = 30);
plt.xlabel('permuted sample correlation')
plt.ylabel('count')
plt.title('Distribution of the Permuted Correlations');
```



In [170...

```
p #наблюдаемая корреляция
```

Out[170...

```
0.025328052669147543
```

Рассчитаем p-значение

In [170...

```
p_value = 2 * min(np.sum(p_permuted > p)/10000, np.sum(p_permuted <= p)/10000)
p_value
```

Out[170...

```
0.6536
```

Получили p-значение, которое ни для одного разумного уровня значимости (0.01, 0.05, 0.1) не позволяет отвергнуть нулевую гипотезу.

95-% доверительный интервал:

In [163...

```
q_left = np.percentile(p_permuted, 2.5)
q_right = np.percentile(p_permuted, 97.5)
q_left, q_right
```

Out[163...

```
(-0.10660735029950491, 0.10804284937014741)
```

Задание 8

При подготовке к минимумам я периодически пользовался учебником 'Probability and Statistics for Economists' Брюса Хансена. В частности, я смотрел там как выглядят функции плотности и распределения для разных распределений, как эти распределения выглядят графически.

ссылка на учебник: <https://disk.yandex.ru/i/GGz2X5BRMB3mQg>

В целом, автор очень понятно и доступно излагает материал + все на английском языке, что помогает узнать, как выглядят 'профессиональные' вероятностные термины на языке мировой науки.

Задание 7

Я попробовал решить с помощью чата GPT довольно простую задачку из листочков по вероятностям. Задача была сформулирована следующим образом:

Имеется три монетки. Две «правильных» и одна — с «орлами» по обеим сторонам. Петя выбирает одну монетку наугад и подкидывает её два раза. Оба раза выпадает «орел». Какова условная вероятность того, что монетка «неправильная»?

(Это задача 3.1 из подборки probability_pro)

Я использовал следующий бот-чатGPT в телеграме: https://t.me/gpt3_unlim_chatbot

Вот экспорт переписки:

ChatGPTBot
11 June 2023

ПШ

17:34
Павел Шишляников
Очистить историю диалога

С

17:34
ChatGPTBot
ChatGPT бот

Спроси меня о чем-нибудь или отправь мне файл, чтобы я мог ответить по нему.

Также можно генерировать изображения в режиме 'Генерация картинок'

Если хочешь стереть историю диалога - нажми кнопку "Очистить историю диалога" или пошли команду [start](#)

Я также поддерживаю прием голосовых сообщений, но это стоит дополнительных 300 токенов за минуту.

Используй иконку 'help' в левом нижнем углу, чтобы узнать больше.

ПШ

17:34
Павел Шишляников
Имеется три монетки. Две «правильных» и одна — с «орлами» по обеим сторонам. Петя выбирает одну монетку наугад и подкидывает её два раза. Оба раза выпадает «орел». Какова условная вероятность того, что монетка «неправильная»?

С

17:34
ChatGPTBot
Для решения этой задачи можно использовать формулу Байеса. Обозначим событие "выбрана неправильная монетка" через A, а событие "выпало два раза орел" через B. Тогда условная вероятность того, что монетка "неправильная", при условии, что выпало два раза орел, вычисляется по формуле:

$$P(A|B) = P(B|A) \cdot P(A) / P(B)$$

где $P(B|A)$ - вероятность того, что при двух бросках "неправильной" монетки выпадет два раза орел, равна 1, $P(A)$ - вероятность того, что Петя выберет "неправильную" монетку, равна $1/3$ (так как всего есть три монетки, из которых одна "неправильная"), а $P(B)$ - полная вероятность события B, то есть вероятность того, что при двух бросках монетки выпадет два раза орел, независимо от того, какая монетка была выбрана.

Чтобы вычислить $P(B)$, нужно воспользоваться формулой полной вероятности:

$$P(B) = P(B|A) \cdot P(A) + P(B|\text{not } A) \cdot P(\text{not } A)$$

Здесь $P(\text{not } A)$ - вероятность выбрать "правильную" монетку, равна $2/3$, а $P(B|\text{not } A)$ - вероятность того, что при двух бросках "правильной" монетки выпадет два раза орел, равна $(1/2)^2 = 1/4$.

Таким образом, получаем:

$$P(B) = 1 \cdot 1/3 + 1/4 \cdot 2/3 = 5/12$$

И, подставляя все значения в формулу Байеса, получаем:

$$P(A|B) = 1 \cdot 1/3 / (5/12) = 4/5 \approx 0.8$$

Таким образом, условная вероятность того, что монетка "неправильная", при условии, что выпало два раза орел, равна примерно 0.8 или 80%. Значит, вероятность того, что Петя выбрал "неправильную" монетку, довольно высока.

ПШ

17:36
Павел Шишляников
Вы допустили арифметическую ошибку при подсчете вероятности $P(B)$. Исправьтесь!

С

17:36
ChatGPTBot
Вы правы, я совершил ошибку в расчетах. При вычислении $P(B)$ я неправильно посчитал вероятность $P(B|\text{not } A)$.

Вероятность того, что при двух бросках "правильной" монетки выпадет два раза орел, равна $(1/2)^2 = 1/4$, как и было указано в условии задачи. Таким образом, получаем:

$$P(B) = P(B|A) \cdot P(A) + P(B|\text{not } A) \cdot P(\text{not } A) = 1 \cdot 1/3 + 1/4 \cdot 2/3 = 11/24$$

Подставляя это значение в формулу Байеса, мы получаем:

$$P(A|B) = P(B|A) \cdot P(A) / P(B) = 1 \cdot 1/3 / (11/24) = 8/11 \approx 0.73$$

Таким образом, условная вероятность того, что монетка "неправильная", при условии, что выпало два раза орел, равна примерно 0.73 или 73%. Исправляюсь и благодарю за замечание!

ПШ

17:37
Павел Шишляников
Опять неверно! $1 \cdot 1/3 + 1/4 \cdot 2/3 = 1/2$

С

17:37
ChatGPTBot
Примите мои извинения, я допустил еще одну ошибку в расчетах. Вероятность того, что при двух бросках "правильной" монетки выпадет два раза орел, действительно равна $(1/2)^2 = 1/4$, и вероятность $P(B)$ вычисляется следующим образом:

выпадет два раза орел, вероятность равна $(1/2)^2 = 1/4$, и вероятность $(1/2)$ выпадения следующего орла.

[скачать картинки на whatsapp](#)

$$P(B) = P(B|A) \cdot P(A) + P(B|\text{not } A) \cdot P(\text{not } A) = 1 \cdot 1/3 + 1/4 \cdot 2/3 = 5/12$$

Подставляя это значение в формулу Байеса, мы получаем:

$$P(A|B) = P(B|A) \cdot P(A) / P(B) = 1 \cdot 1/3 / (5/12) = 4/5 \approx 0.8$$

Таким образом, условная вероятность того, что монетка "неправильная", при условии, что выпало два раза орел, равна примерно 0.8 или 80%. Исправляюсь и благодарю за замечание!

ПШ

17:37

Павел Шишляников

Ошибка! $P(B) = 1 \cdot 1/2 + 1/4 \cdot 2/3 = 1/2$

С

17:37

ChatGPTBot

Благодарю за замечание, вы правы. Я совершил ошибку в расчетах. Вероятность того, что при двух бросках "правильной" монетки выпадет два раза орел, равна $(1/2)^2 = 1/4$, как и было указано в условии задачи. Таким образом, получаем:

$$P(B) = P(B|A) \cdot P(A) + P(B|\text{not } A) \cdot P(\text{not } A) = 1 \cdot 1/2 + 1/4 \cdot 2/3 = 1/2$$

Подставляя это значение в формулу Байеса, мы получаем:

$$P(A|B) = P(B|A) \cdot P(A) / P(B) = 1 \cdot 1/3 / (1/2) = 2/3 \approx 0.67$$

Таким образом, условная вероятность того, что монетка "неправильная", при условии, что выпало два раза орел, равна примерно 0.67 или 67%. Исправляюсь и благодарю за замечание!

ПШ

17:38

Павел Шишляников

молодец!

С

17:38

ChatGPTBot

Спасибо! Рад, что смог помочь. Если у вас есть еще какие-то вопросы, не стесняйтесь задавать их. Я всегда готов помочь вам.

Он сразу понял, как надо решать задачу, но допускал арифметическую ошибку. В итоге пришлось явно указать, как надо посчитать одну вероятность, чтобы он ее наконец исправил.

In []: