Теория вероятностей и статистика, НИУ ВШЭ

Домашнее задание

ФИО: Репенкова Полина Александровна, Группа: БЭК212

Ввод [12]:

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

from tqdm import tqdm

import warnings
warnings.filterwarnings('ignore')
```

Задача 1. Таксисты в Самарканде.

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 равновероятно.

Пункт а.

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

Ввод [13]:

```
def prob(n): # вероятность встретить на первом заказе не приезжавшего ранее таксиста - n
    return  ((n - 1) * (n - 2) * (n - 3) * (n - 4) * (n - 5) * (n - 6) * (n - 7) * (n -
```

Ввод [14]:

```
L = []
n = 0

for i in range(9, 201):
    L.append(prob(i))
    if prob(i) > n:
        n = prob(i)
        n_ind = i

print('Оценка числа n методом макимального правдоподобия, n =', n_ind)
```

```
Оценка числа n методом макимального правдоподобия, n = 42
```

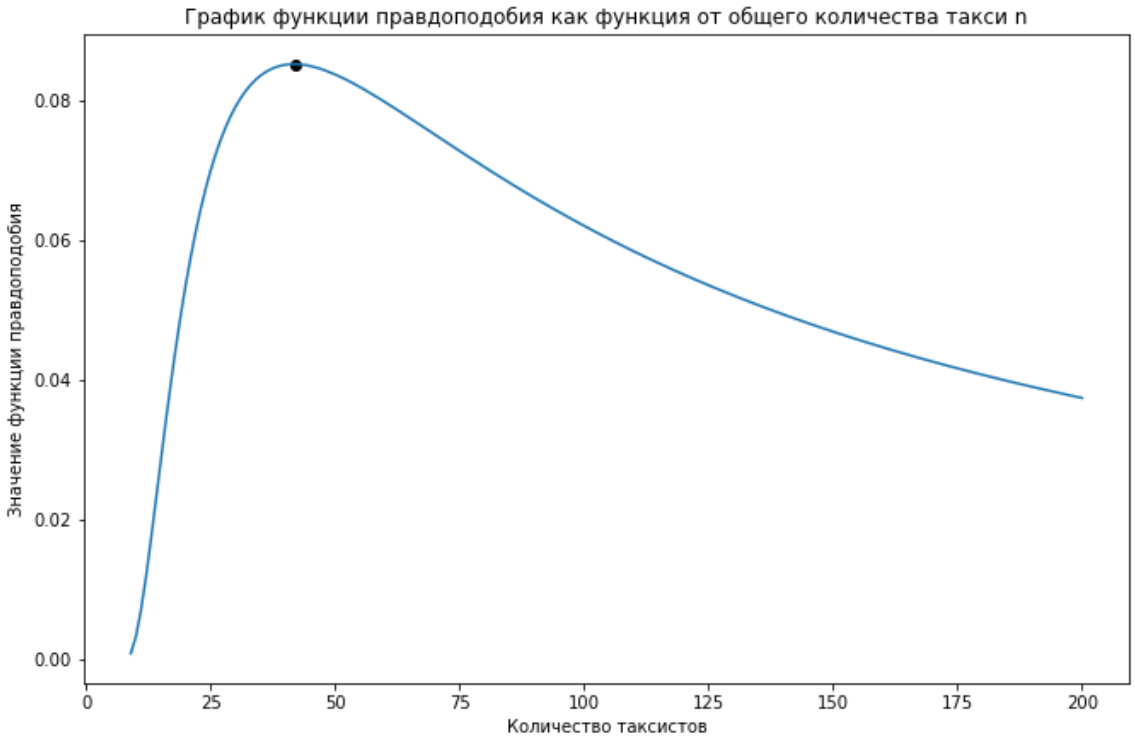
Ввод [15]:

```
plt.figure(figsize = (11, 7))

plt.plot(np.arange(9, 201), L)
plt.scatter(n_ind, n, color = 'black')

plt.title('График функции правдоподобия как функция от общего количества такси n')
plt.xlabel('Количество таксистов')
plt.ylabel('Значение функции правдоподобия')

plt.show()
```



Пункт 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.

Ввод [16]:

```
def wait(x):
    mat = 0
    for i in range(2, x + 2):
        prod = 1

        for j in range(i - 1):
            prod *= (x - j) / x
        mat += i * (i - 1) * prod / x

    return mat
```

Ввод [17]:

```
a = np.arange(100)
mu = np.array([])

for n in a:
    mu = np.append(mu, wait(n))

mse = np.array([])
for i in range(0,100):
    mse = np.append(mse, (mu[i] - 10) ** 2) # выборочный момент равен 10, имеем одно наб

print('Оценка числа n методом моментов, n =', mse.argmin())
```

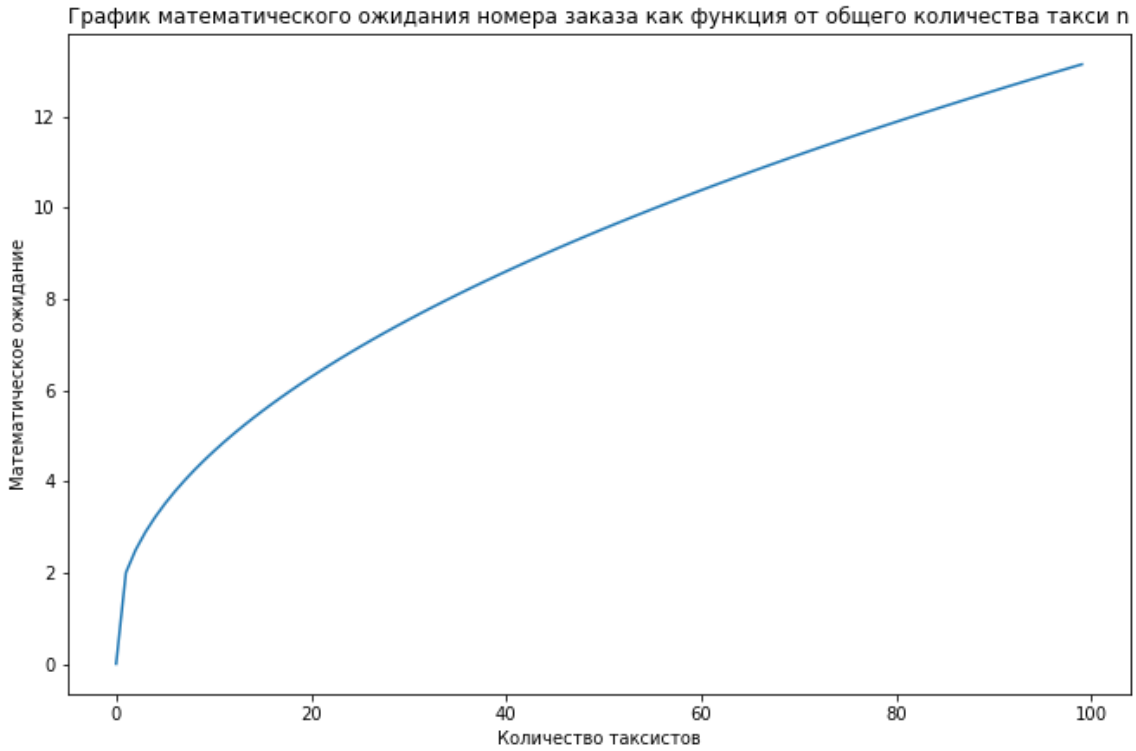
```
Оценка числа n методом моментов, n = 55
```

Ввод [18]:

```
plt.figure(figsize = (11, 7))
plt.plot(a, mu)

plt.title('График математического ожидания номера заказа как функция от общего количеств
plt.xlabel('Количество таксистов')
plt.ylabel('Математическое ожидание')

plt.show()
```



Задача 3. Иннокентий.

Пункт а

Ввод [22]:

```
np.random.seed(5)

rv = sts.expon(scale = 1)

samples = rv.rvs((10000, 20))
sr_real = rv.mean()
```

Ввод [23]:

```
a = 0.95
v = 20
h = 10000
b = 0.5
c1 = 0
c2 = 0
c3 = 0
```

Ввод [24]:

```
np.random.seed(5)
for i in tqdm(samples):

    sr = np.mean(i)
    var = np.var(i)

    interv = sts.norm.interval(alpha = a, loc = sr, scale = (var / v) ** b)

    c1 = c1 + (interv[0] < sr_real < interv[1])
    p1 = c1 / h

    boo = np.random.choice(i, size = (h, v), replace = True)
    boo_sr = np.mean(boo, axis = 1)
    boo_int = [np.quantile(boo_sr, 0.025), np.quantile(boo_sr, 0.975)]

    c2 = c2 + (boo_int[0] < sr_real < boo_int[1])
    p2 = c2 / h

    t_boo = np.random.choice(i, size = (h, v), replace = True)
    t_boo_var = (np.var(t_boo, axis = 1) / v ) ** b
    t_boo_sr = np.mean(t_boo, axis = 1)

    q1 = np.quantile((t_boo_sr - sr)/ t_boo_var, 0.025)
    q2 = np.quantile((t_boo_sr - sr)/ t_boo_var, 0.975)
    t_boo_int = [sr - q2 * ((var / v) ** b), sr - q1 * ((var / v) ** b)]

    c3 = c3 + (t_boo_int[0] < sr_real < t_boo_int[1])
    p3 = c3 / h
```

```
100%|██████████████████████████████████████████████████████████████
██████| 10000/10000 [01:22<00:00, 121.95it/s]
```

Ввод [25]:

```
print('Вероятность накрытия классическим асимптотическим нормальным интервалом: ', p1)
print('Вероятность накрытия наивным бутстрэпом', p2)
print('Вероятность накрытия бутстрэпом t-статистики', p3)
```

```
Вероятность накрытия классическим асимптотическим нормальным интервалом:
0.9012
Вероятность накрытия наивным бутстрэпом 0.9076
Вероятность накрытия бутстрэпом t-статистики 0.9488
```

Пункт б

Ввод [26]:

```
np.random.seed(5)

rv = sts.t(df = 3)

samples = rv.rvs((10000, 20))
sr_real = rv.mean()
```

Ввод [27]:

```
a = 0.95
v = 20
h = 10000
b = 0.5
c1 = 0
c2 = 0
c3 = 0
```

Ввод [28]:

```
np.random.seed(5)
for i in tqdm(samples):

    sr = np.mean(i)
    var = np.var(i)

    interv = sts.norm.interval(alpha = a, loc = sr, scale = (var / v) ** b)
    c1 = c1 + (interv[0] < sr_real < interv[1])
    p1 = c1 / h

    boo = np.random.choice(i, size = (h, v), replace = True)
    boo_sr = np.mean(boo, axis = 1)
    boo_int = [np.quantile(boo_sr, 0.025), np.quantile(boo_sr, 0.975)]
    c2 = c2 + (boo_int[0] < sr_real < boo_int[1])
    p2 = c2 / h

    t_boo = np.random.choice(i, size=(h, v), replace = True)
    t_boo_sr = np.mean(t_boo, axis = 1)
    t_boo_var = (np.var(t_boo, axis = 1) / v) ** b

    q1 = np.quantile((t_boo_sr-sr)/t_boo_var, 0.025)
    q2 = np.quantile((t_boo_sr-sr)/t_boo_var, 0.975)
    t_boo_int = [sr - q2 * ((var / v) ** b), sr - q1 * ((var / v) ** b)]

    c3 = c3 + (t_boo_int[0] < sr_real < t_boo_int[1])
    p3 = c3 / h
```

```
100%|██████████████████████████████████████████████████████████████
██████| 10000/10000 [01:23<00:00, 119.79it/s]
```

Ввод [29]:

```
print('Вероятность накрытия классическим асимптотическим нормальным интервалом', p1)
print('Вероятность накрытия наивным бутстрэпом', p2)
print('Вероятность накрытия бутстрэпом t-статистики', p3)
```

```
Вероятность накрытия классическим асимптотическим нормальным интервалом 0.
9318
Вероятность накрытия наивным бутстрэпом 0.9151
Вероятность накрытия бутстрэпом t-статистики 0.9209
```

Пункт в

Способ из пункта б оказался лучше

Задача 4. Экзамен 1.0

Пункт а

Ввод [30]:

```
df = pd.read_excel('exam.xlsx')[['Last Name',  'Score']]
```

Ввод [31]:

```
df
```

Out[31]:

	Last Name	Score
0	Репенкова	16
1	Ролдугина	0
2	Сафина	19
3	Сидоров	26
4	Солоухин	21
...	...	...
327	Сенников	19
328	Ся	0
329	Сятова	0
330	Темиркулов	0

Ввод [32]:

```
vowels = ['Э','Ю','Я','А','И','У','О','Е','Ё','Ы']
```

Ввод [33]:

```
not_vowels = ['Б', 'В', 'Г', 'Д', 'Ж', 'З', 'Й', 'К', 'Л', 'М', 'Н', 'П', 'Р', 'С', 'Т',
```

Ввод [34]:

```
df_vow = df.loc[df['Last Name'].astype(str).str[0].isin(vowels)]
```

Ввод [35]:

```
df_vow
```

Out[35]:

	Last Name	Score
17	Адилхан	25
18	Алексанян	26
32	Охотин	25
45	Аврамчук	29
46	Авсеенко	26
47	Адамокова	20
48	Адамцева	19
49	Азаров	24
50	Алексеева	25
51	Афанасьев	28
67	Иванов	16
68	Иванова	16
85	Осенева	15
113	Абдугаффорова	17
114	Амреева	4
126	Ермишова	23
143	Овчарова	21
144	Осиновскова	13
158	Ускова	16
166	Ягжов	19
167	Яковлева	6
168	Ян	15
169	Янковская	8
170	Агамалов	13
171	Акимов	23
172	Амбросимов	11
173	Асаналиева	7
174	Асонкова	12
184	Есауленко	20
188	Иванов	20
189	Исмаилов	17
220	Уначева	18
221	Ушатова	20
227	Яковлева	17
228	Ямкова	11
229	Адмайкин	21
230	Алиева	5

	Last Name	Score
240	Ермошин	13
270	Янышен	13
271	Яхьяева	13
272	Абдулаева	21
273	Аврамидис	0
274	Авутов	0
275	Алина	0
276	Асатрян	0
277	Афанасьев	0
299	Евлоева	0
300	Ермаков	22
331	Эшмеев	16

Ввод [36]:

```
df_notvow = df.loc[df['Last Name'].astype(str).str[0].isin(not_vowels)]
```

Ввод [37]:

```
df_notvow
```

Out[37]:

	Last Name	Score
0	Репенкова	16
1	Ролдугина	0
7	Торова	17
8	Трофимова	20
9	Федгинкель	21
...	...	...
322	Пейсахов	15
323	Петрова	10
324	Полищук	25
325	Рахимов	0

Ввод [38]:

```
vow = df_vow['Score']
```

Ввод [39]:

```
not_vow = df_notvow['Score']
```

Ввод [40]:

```
t_statistic, p_value = sts.ttest_ind(vow, not_vow, equal_var = True)

print("t-тест:", t_statistic)
print("p-значение:", p_value)

if p_value < 0.05:
    print("Отвергаем нулевую гипотезу.")
else:
    print("Нет оснований отвергать гипотезу. Ожидаемые результаты экзамена в двух группа
```

```
t-тест: -0.960824693001346
p-значение: 0.337424525218481
Нет оснований отвергать гипотезу. Ожидаемые результаты экзамена в двух гру
ппах равны
```

Пункт б

Ввод [41]:

```
np.random.seed(43)

vow_boo = np.random.choice(vow, size = (10000, len(vow)))
not_vow_boo = np.random.choice(not_vow, size = (10000, len(not_vow)))

real_diff = np.mean(vow) - np.mean(not_vow)
boo_diff = np.mean(vow_boo, axis = 1) - np.mean(not_vow_boo, axis = 1)

q1 = np.quantile(boo_diff, 0.025)
q2 = np.quantile(boo_diff, 0.975)

print([real_diff - q2, real_diff - q1])

if ((real_diff - q2) < 0 < (real_diff - q1)):
    print('Не отвергаем гипотезу, так как ноль входит в интервал')
else:
    print('Отвергаем гипотезу')
```

```
[-2.463191541676909, 2.431829358249325]
Не отвергаем гипотезу, так как ноль входит в интервал
```

Пункт в

Ввод [42]:

```
np.random.seed(43)

vow_boo = np.random.choice(vow, size = (10000, len(vow)))
not_vow_boo = np.random.choice(not_vow, size=(10000, len(not_vow)))

real_diff = np.mean(vow) - np.mean(not_vow)
boo_diff = np.mean(vow_boo, axis = 1) - np.mean(not_vow_boo, axis = 1)

real_std = (np.var(vow) / len(vow))** 0.5 + (np.var(not_vow) / len(not_vow)) ** 0.5
boo_std = (np.var(vow_boo, axis = 1) / len(vow_boo)) ** 0.5 + (np.var(not_vow_boo, axis

q1 = np.quantile((boo_diff - real_diff) / boo_std, 0.025)
q2 = np.quantile((boo_diff - real_diff) / boo_std, 0.975)
```

Ввод [43]:

```
print([real_diff - q2 * real_std, real_diff - q1 * real_std])

if ((real_diff - q2 * real_std - q2) < 0 < (real_diff - q1 * real_std)):
    print('Не отвергаем гипотезу, так как ноль входит в интервал')
else:
    print('Отвергаем гипотезу')
```

```
[-27.683643901654484, 24.190301975696656]
Не отвергаем гипотезу, так как ноль входит в интервал
```

Задача 5. Экзамен 2.0

Пункт а

Ввод [44]:

```
median = df['Score'].median()

table = pd.DataFrame({'Меньше': [np.sum(df_vow['Score'] < median), np.sum(df_notvow['Sco
                      'Больше': [np.sum(df_vow['Score'] > median), np.sum(df_notvow['Sco

table
```

Out[44]:

	Меньше	Больше
Not vow	28	21
Vow	124	124

Ввод [45]:

```
a = 0.05

x = (21 / 28) / (124 / 124)
z = sts.norm.ppf(1 - a / 2)
se = np.sqrt(1 / 124 + 1 / 124 + 1 / 21 + 1 / 28)

p_value = 2 * min(sts.norm.cdf(np.log(x) / se), 1 - sts.norm.cdf(-np.log(x) / se))
print(p_value)

if p_value > a:
    print('Нет оснований отвергать гипотезу')
else:
    print('Отвергаем нулевую гипотезу')
```

```
0.3616708079409404
Нет оснований отвергать гипотезу
```

Пункт б

Ввод [46]:

```
x = (21 / (28 + 21)) / (124 / (124 + 124))
se = np.sqrt(1 / 124 + 1/21 - 1 / (124 + 124) - 1 / (21 + 28))
z = sts.norm.ppf(1 - a / 2)

p_value = 2 * min(sts.norm.cdf(np.log(x) / se), 1 - sts.norm.cdf(-np.log(x) /se))

if p_value > a:
    print('Нет оснований отвергать гипотезу')
else:
    print('Отвергаем нулевую гипотезу')
```

```
Нет оснований отвергать гипотезу
```

Задача 6. Иннокентий и экзамен

Пункт а

Ввод [51]:

```
lens = [len(s) for s in df['Last Name']]
score = list(df['Score'])

b = np.mean(score) / np.mean(len)

corr = sts.pearsonr(lens, score)[0]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call las
t)
C:\Users\4880~1\AppData\Local\Temp/ipykernel_20140/1865940027.py in <modul
e>
      2 score = list(df['Score'])
      3
----> 4 b = np.mean(score) / np.mean(len)
      5
      6 corr = sts.pearsonr(lens, score)[0]

<__array_function__ internals> in mean(*args, **kwargs)

C:\ProgramData\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py in me
an(a, axis, dtype, out, keepdims, where)
   3417             return mean(axis=axis, dtype=dtype, out=out, **kwargs)
   3418
-> 3419     return _methods._mean(a, axis=axis, dtype=dtype,
   3420                           out=out, **kwargs)
   3421

C:\ProgramData\Anaconda3\lib\site-packages\numpy\core\_methods.py in _mean
(a, axis, dtype, out, keepdims, where)
    188             ret = ret.dtype.type(ret / rcount)
    189     else:
--> 190         ret = ret / rcount
    191
    192     return ret

TypeError: unsupported operand type(s) for /: 'builtin_function_or_method'
and 'int'
```

Ввод [48]:

```
print(b)
print(corr)
```

```
0.5

---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
C:\Users\4880~1\AppData\Local\Temp/ipykernel_20140/378246415.py in <module
>
      1 print(b)
----> 2 print(corr)

NameError: name 'corr' is not defined
```

Пункт б

Ввод [49]:

```
np.random.seed(42)
corr_perm = []

for i in tqdm(range(0, 10000)):
    scoressf = np.random.permutation(score)
    corr_sim = sts.pearsonr(lens, scoressf)[0]
    corr_perm.append(corr_sim)

q1 = np.percentile(corr_perm, 0.25)
q2 = np.percentile(corr_perm, 0.975)

p_value = 2 * np.mean(np.array(corr_perm) > corr)
```

```
  0%|
| 0/10000 [00:00<?, ?it/s]

---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
C:\Users\4880~1\AppData\Local\Temp/ipykernel_20140/355038307.py in <module
>
      4 for i in tqdm(range(0, 10000)):
      5     scoressf = np.random.permutation(score)
----> 6     corr_sim = sts.pearsonr(len, scoressf)[0]
      7     corr_perm.append(corr_sim)
      8

C:\ProgramData\Anaconda3\lib\site-packages\scipy\stats\stats.py in pearson
r(x, y)
   4009
   4010     """
-> 4011     n = len(x)
   4012     if n != len(y):
   4013         raise ValueError('x and y must have the same length.')

TypeError: object of type 'builtin_function_or_method' has no len()
```

Ввод [50]:

```
print(p_value)

if p_value > a:
    print('Нет оснований отвергать гипотезу')
else:
    print('Отвергаем нулевую гипотезу')
```

```
0.38315185048228795
Нет оснований отвергать гипотезу
```

Задача 7. Общаемся с искуственным интеллектом

Я решила поболтать с чатикомGPT, чтобы он помог мне решить 1 задачу из задачного минимума 3 контрольной. Скрины с полным диалогом и наводками я приложила (на репозитории, тут вроде как фотки вставлять нельзя).

Задача 8. Источник

Я очень хочу упомянуть в этом месте видео от команды прекрасных ребят - https://youtu.be/zkBWjZ2tZ44 (https://youtu.be/zkBWjZ2tZ44). В своё время, когда я только начинала готовиться к третьей контрольной, очень помогло систематизировать знания по оценке максимального правдоподобия и легко запомнить правильный порядок действий с ней. Тем более, что сам видос классно смонтирован, а ребята очень классно поют)
