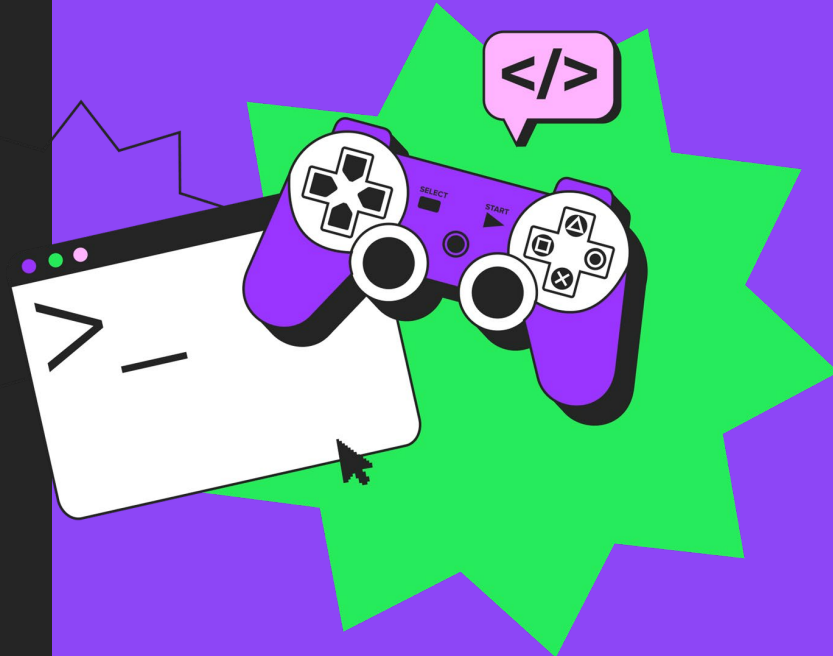




# Регрессионный анализ

Линейная регрессия. Логистическая регрессия.





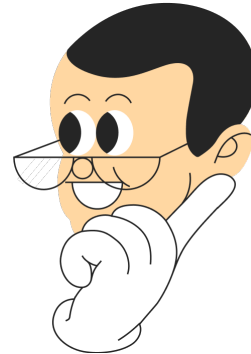
# План курса





## Что будет на уроке сегодня

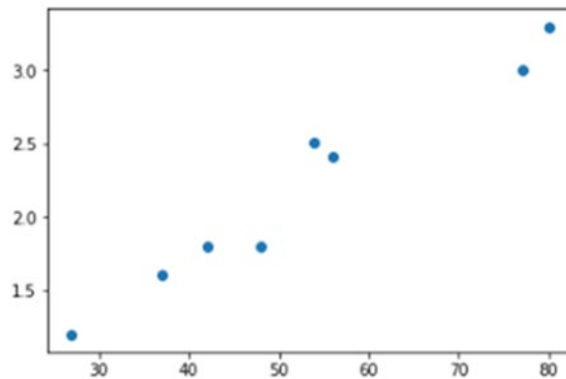
- ✚ Методы нахождения коэффициентов для уравнения линейной регрессии
- ✚ Условия применимости линейной регрессии
- ✚ Оценка полученной модели линейной регрессии
- ✚ Логистическая регрессия





# Линейная регрессия

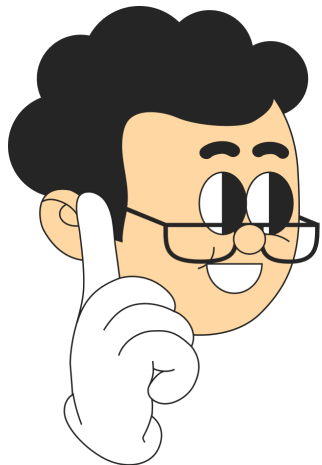
Площадь	Цена
27	1.2
37	1.6
42	1.8
48	1.8
56	2.6
57	2.5
77	3
80	3.3



*двумерные данные*



Многомерный статистический анализ – раздел статистики, который посвящен исследованиям экспериментов с многомерными наблюдениями.





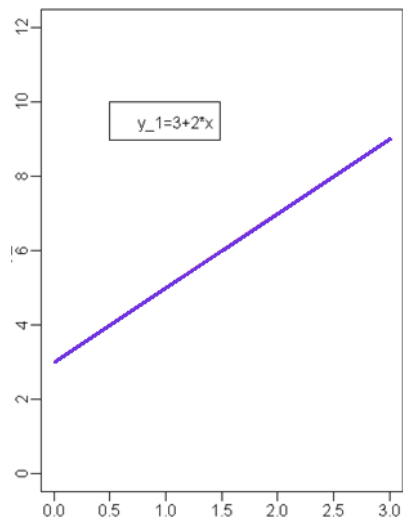
## Парная линейная регрессия

Если признак один, то такая линейная регрессия называется парной. Она описывает связь признака  $x$  с результирующим признаком  $y$ .

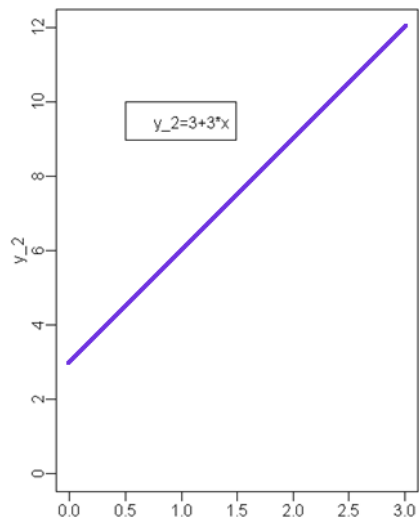
$$y = a + bx$$



## Коэффициенты уравнения линейной регрессии



**А**



**Б**

X	Y
0	3
1	5
2	7
3	9

$$y = 3 + 2 * x$$

**А**

X	Y
0	3
1	6
2	9
3	12

$$y = 3 + 3 * x$$

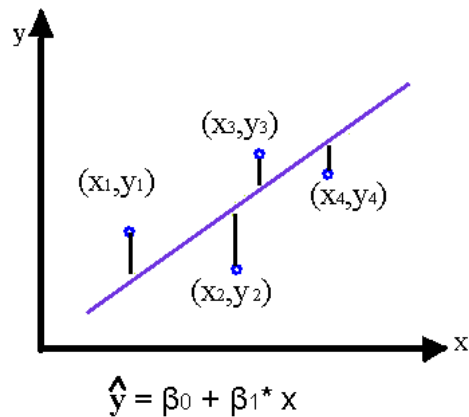
**Б**



## Как образуется прямая линейной регрессии?

1. Определяемся с моделью ( $\hat{y} = \beta_0 + \beta_1 * x$ )
2. Находим коэффициенты  $\beta$  для уравнения линейной регрессии
3. Подставляем в это уравнение значения признака  $x$  и рассчитываем оценочные значения  $y$

$$(y_1 - \hat{y}_1)^2 + \dots + (y_4 - \hat{y}_4)^2 = \min$$







# Условия применимости



## Условия применимости

- 1 Наличие линейной зависимости между независимой переменной  $x$  и зависимой  $y$
- 2 Независимость остатков
- 3 Для любого значения  $x$  значение зависимой переменной  $y$  распределено нормально
- 4 Гомоскедастичность



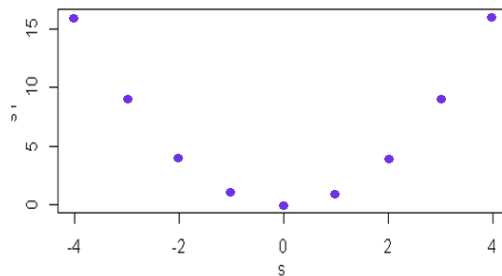
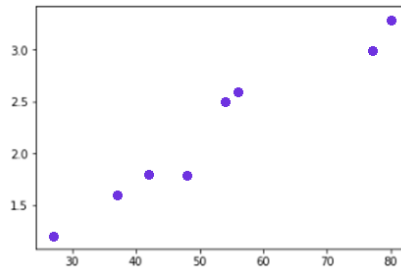
# 1. Наличие линейной зависимости между независимой переменной $x$ и зависимой $y$

Проверка условия: график

Как решить задачу нелинейности?

Нелинейная трансформация  $x$  или/и  $y$  :

- ✓ логарифм,
- ✓ квадратный корень
- ✓ умножение на обратное число

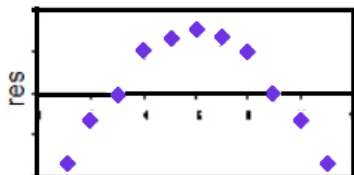
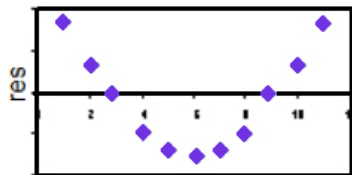
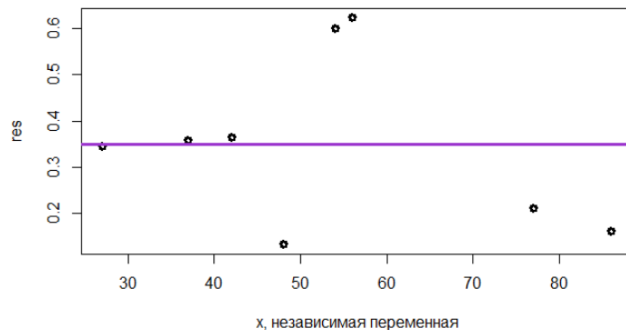




## 2 Независимость остатков

Мы не должны видеть какого-то шаблона в поведении остатков с течением времени. Например, остатки не должны постоянно с течением времени расти.

Если точки на графике остатков разбросаны случайно вокруг горизонтальной линии, то данные подходят для модели линейной регрессии.



Эти два графика показывают, что для данных подойдет лучше нелинейная модель. Решить проблему можно также путем трансформации данных.



## Нормальное распределение остатков

Распределение остатков следует нормальному распределению.

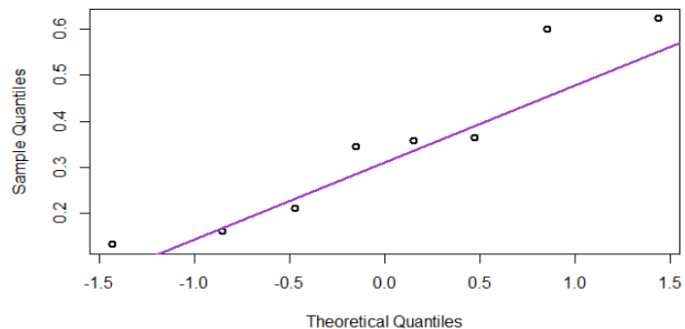
Методы проверки этого условия:

- ✓ QQ-plot
- ✓ Тест Шапиро-Уилка

```
resid = p-y_pred

resid
array([-0.01760804, -0.00506654,  0.00120421, -0.23127089,  0.12001646,
        0.25876231, -0.15490054,  0.02886191])

stats.shapiro(resid)
(0.9600725769996643, 0.8107908368110657)
```



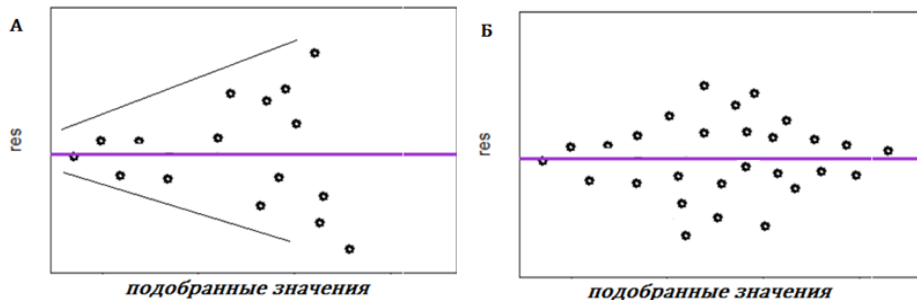
## Гомоскедастичность

Под этим свойством понимается постоянство дисперсии при всех значениях  $x$ . Т.е. стандартное отклонение  $\sigma$  одинаково при всех значениях  $x$ , если разброс остатков неравномерный, то говорят о присутствии **гетероскедастичности**.

Проверить на наличие гетероскедастичности можно с помощью графика.

Решить эту проблему:

- ✓ log- трансформация  $y$
- ✓ переопределить зависимую переменную



Примеры гетероскедастичности



## Трансформация данных

Трансформация	Уравнение регрессии	Оценочное значение $\hat{y}$
$\log(y)$	$\log(y) = b_0 + b_1 x$	$\hat{y} = 10^{b_0 + b_1 x}$
$1/y$	$1/y = b_0 + b_1 x$	$\hat{y} = \frac{1}{b_0 + b_1 x}$
$\sqrt{y}$	$\sqrt{y} = b_0 + b_1 x$	$\hat{y} = (b_0 + b_1 x)^2$
$\log(x)$	$y = b_0 + b_1 \log(x)$	$\hat{y} = b_0 + b_1 \log(x)$
$\log(x)$ и $\log(y)$	$\log(y) = b_0 + b_1 \log(x)$	$\hat{y} = 10^{b_0 + b_1 \log(x)}$



## 3 метода построения линейной регрессии

- Математические формулы
- Матричный метод
- Метод градиентного спуска





## Расчет коэффициентов по формулам

$$b = \frac{n \sum_{i=1}^n x_i y_i - (\sum_{i=1}^n x_i)(\sum_{i=1}^n y_i)}{n \sum_{i=1}^n x_i^2 - (\sum_{i=1}^n x_i)^2}$$

$$a = \bar{y} - b\bar{x}$$

$$\bar{y} = \frac{\sum_{i=1}^n y_i}{n}$$

$$\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$$

Получим модель вида  $\hat{y} = 0.1715 + 0.0387 * x$  для случайных величин цена и площадь



```
import numpy as np

s=np.array([27, 37, 42, 48, 57, 56, 77, 80])

p = np.array([1.2, 1.6, 1.8, 1.8, 2.5, 2.6, 3, 3.3])

n=8

b1 = (n*np.sum(p*s)- np. sum(s) * np.sum(p))/(n* np.sum(s**2)-
np.sum(s)**2)

b1
0.03874584717607981
```

или 2й способ:

```
b1= (np.mean(s*p)-np.mean(s)*np.mean(p))/(np.mean(s**2)-np.mean(s)**2)
b1
0.03874584717607981

b0 = np.mean(p)-b1*np.mean(s)
b0
0.17147009966776983

y_pred=0.17147009 +0.03874585 * s

y_pred
array([1.21760804, 1.60506654, 1.79879579, 2.03127089, 2.37998354,
       2.34123769, 3.15490054, 3.27113809])
```



## Функция потерь

Функция потерь *mse* - мера измерения ошибок, которые функция делает на нашем наборе данных

$$mse = \frac{\sum (y - y_{pred})^2}{n}$$

$n$  – число измерений



```
mse = ((p-y_pred)**2).sum() / n  
mse  
0.02000155730897011
```



## Матричный метод расчета коэффициентов линейной регрессии.

$$\hat{y} = b_0 + b_1 x$$

$$Y = X * B$$

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix}$$

$$\hat{B} = (X^T * X)^{-1} * X^T * Y$$

```
s=np.array([27, 37, 42, 48, 57, 56, 77, 80])  
p = np.array([1.2, 1.6, 1.8, 1.8, 2.5, 2.6, 3, 3.3])  
x = s.reshape((8,1))  
x  
array([[27],  
       [37],  
       [42],  
       [48],  
       [57],  
       [56],  
       [77],  
       [80]])  
  
y= p.reshape((8,1))  
y  
array([[1.2],  
       [1.6],  
       [1.8],  
       [1.8],  
       [2.5],  
       [2.6],  
       [3. ],  
       [3.3]])
```

```
X = np.hstack([np.ones((8,1)),x])  
X  
array([[ 1., 27.],  
       [ 1., 37.],  
       [ 1., 42.],  
       [ 1., 48.],  
       [ 1., 57.],  
       [ 1., 56.],  
       [ 1., 77.],  
       [ 1., 80.]])  
  
B = np.dot(np.linalg.inv(np.dot(X.T,X)), X.T @ y)  
B  
array([[0.1714701 ],  
       [0.03874585]])
```



## Расчет коэффициентов методом градиентного спуска для $\hat{y} = \beta_1 x$

```
# Градиентный спуск

x=np.array([27, 37, 42, 48, 57, 56, 77, 80])

y = np.array([1.2, 1.6, 1.8, 1.8, 2.5, 2.6, 3, 3.3])

def mse_(B1, y = y, x = x, n = 8):
    return np.sum((B1*x - y)**2)/ n

alpha = 1e-6

#mse= 1/n * np.sum((B1*x - y)**2)
#mse = (2/n) * np.sum((B1*x - y) * x)

B1 = 0.1

n = 8
```

```
for i in range(10):
    B1 -= alpha * (2/n) * np.sum((B1 * x - y) * x)
    print('B1 = {}'.format(B1))

B1 = 0.09963717500000001
B1 = 0.0992766067715
B1 = 0.09891828127738128
B1 = 0.09856218456783597
B1 = 0.09820830277982404
B1 = 0.09785662213653352
B1 = 0.09750712894684428
B1 = 0.09715980960479491
B1 = 0.09681465058905309
B1 = 0.09647163846238918
```



```
for i in range (3000):
    B1 -= alpha * (2/n) * np.sum ((B1 * x - y) * x)
    if i % 500 == 0:
        print ('Iteranion = {i}, B1 = {B1}, mse ={mse}'.format(i= i, B1 = B1, mse = mse_(B1)))

Iteranion = 0, B1 = 0.04166800663447113, mse =0.022847216639871513
Iteranion = 500, B1 = 0.04166800643986129, mse =0.022847216639871402
Iteranion = 1000, B1 = 0.0416680064312654, mse =0.022847216639871392
Iteranion = 1500, B1 = 0.04166800643088573, mse =0.022847216639871406
Iteranion = 2000, B1 = 0.041668006430868966, mse =0.022847216639871406
Iteranion = 2500, B1 = 0.04166800643086872, mse =0.022847216639871392

# а теперь посчитаем mse  через записанную ранее функцию и убедимся, что они одинаковы

mse_(0.041668)
0.022847216640000008
```



## Функции в Python для построения линейной регрессии

```
import numpy as np
from sklearn.linear_model import LinearRegression

model = LinearRegression() # зададим модель линейной регрессии

# делаем массив s двумерным атрибутом reshape(-1,1)
s=s.reshape(-1,1)
array([[27],
       [37],
       [42],
       [48],
       [57],
       [56],
       [77],
       [80]])

regres = model.fit(s,p) # подбираем коэффициенты

print(regres.intercept_) # выводим интерсепт
0.1714700996677747

print(regres.coef_) # выводим коэффициент
[0.03874585]
```

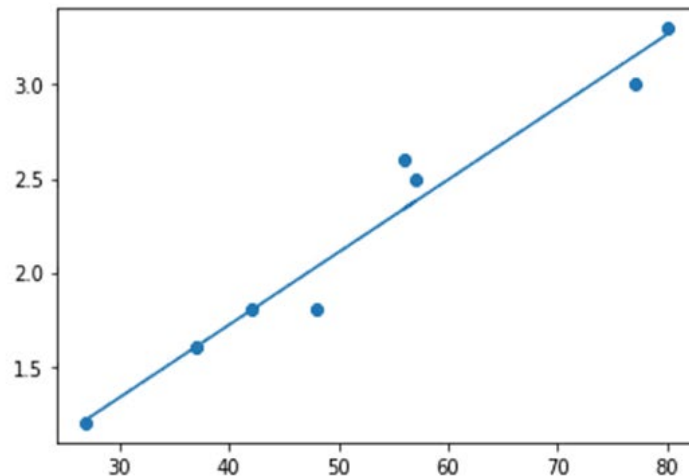
## Функция predict()

```
y_pred = model.predict(s) # подставим площадь в модель и посчитаем предиктовые значения цены квартиры
```

```
y_pred  
array([1.21760797, 1.60506645, 1.79879568, 2.03127076, 2.37998339,  
       2.34123754, 3.15490033, 3.27113787])
```

```
df = pd.DataFrame({'реальные': p, 'предсказанные':  
y_pred})
```

```
df  
реальные  предсказанные  
0    1.2      1.217608  
1    1.6      1.605066  
2    1.8      1.798796  
3    1.8      2.031271  
4    2.5      2.379983  
5    2.6      2.341238  
6    3.0      3.154900  
7    3.3      3.271138
```







## Коэффициент детерминации $R^2$

Коэффициент детерминации показывает, какую долю изменчивости у описала подобранная математическая модель. Коэффициент детерминации равен квадрату коэффициента корреляции и обозначается  $R^2 = r^2$ .

```

r = np.corrcoef(s,p) [1,0]
r
0.9785768205829909

r**2
0.9576125937823151

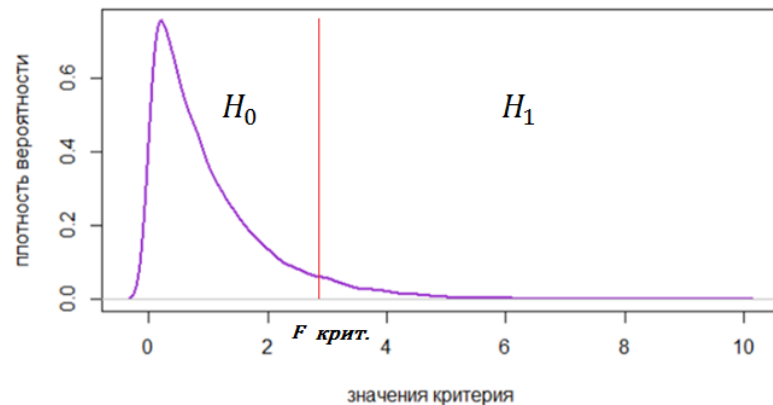
# или получим из математической модели
regres.score(s, p)
0.957612593782315
```

## Оценка значимости математической модели. Критерий Фишера

F-критерий Фишера позволяет оценить значимость модели линейной регрессии  
*Установим уровень значимости  $\alpha = 0,05$ .*

```
stats.f.ppf(1-0.05, 1, 6)
5.987377607273699
```

k1	1	2	3	4	5	6	8	12	24	$\infty$
k2										
1	161,45	199,50	215,72	224,57	230,17	233,97	238,89	243,91	249,04	254,32
2	18,51	19,00	19,16	19,25	19,30	19,33	19,37	19,41	19,45	19,50
3	10,13	9,55	9,28	9,12	9,01	8,94	8,84	8,74	8,64	8,53
4	7,71	6,94	6,59	6,39	6,26	6,16	6,04	5,91	5,77	5,63
5	6,61	5,79	5,41	5,19	5,05	4,95	4,82	4,68	4,53	4,36
6	5,99	5,14	4,76	4,53	4,39	4,28	4,15	4,00	3,84	3,67
7	5,59	4,74	4,35	4,12	3,97	3,87	3,73	3,57	3,41	3,23
8	5,32	4,46	4,07	3,84	3,69	3,58	3,44	3,28	3,12	2,93





## Оценка значимости математической модели. Критерий Фишера

$$Fp = MS_{\phi} / MS_o$$

```
# критерий Фишера F = Msf / Mso
# в свою очередь Msf (фактическая сумма квадратных отклонений на одну степень свободы)
# Msf = SSf/ df1

# остаточная сумма квадратных отклонений на 1 степень свободы
# Mso = SSo / df2

# df1 - степени свободы числителя df1 = p - 1 , где p - число параметров (у нас площадь и цена, т.е. 2)
# df2 - степень свободы знаменателя df2 = n - p, где n - число парных измерений ( у нас n =8 )

# SSf - сумма квадратных отклонений фактическая
# SSo - сумма квадратных отклонений остаточная

df1 = 2 - 1
df2 = 8 - 2
```



## Оценка значимости математической модели. Критерий Фишера

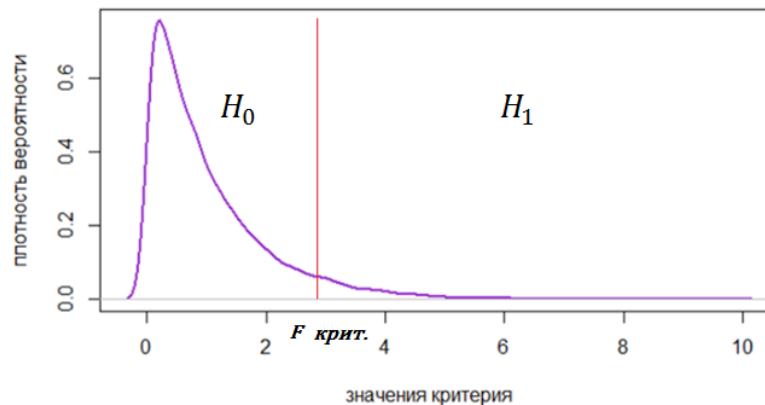
```
SSf = sum((y_pred - np.mean(p))**2)
SSf
3.614987541528235

SSo = np.sum((p - y_pred)**2)
SSo
0.16001245847176088

Msf = SSf / df1
Msf
3.614987541528235

Mso = SSo / df2
Mso
0.02666874307862681

F = Msf / Mso
F
135.55147803067638
```



## Оценка значимости отдельных коэффициентов. Критерий Стьюдента

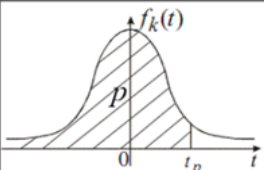
t-критерий Стьюдента позволяет оценить значимость отдельных коэффициентов модели линейной регрессии

Установим уровень значимости  $\alpha = 0,05$ .



```
import scipy.stats as stats

stats.t.ppf(1-0.025, 6)
2.4469118487916806
```



Число степеней свободы $k$	Вероятность, $p$					
	0,9	0,95	0,975	0,99	0,995	0,9995
1	2	3	4	5	6	7
1	3,078	6,314	12,706	31,821	63,657	636,619
2	1,886	2,920	4,303	6,965	9,925	31,598
3	1,638	2,353	3,182	4,541	5,841	12,941
4	1,533	2,132	2,776	3,747	4,604	8,610
5	1,476	2,015	2,571	3,365	4,032	6,869
6	1,440	1,943	2,447	3,143	3,707	5,959
7	1,415	1,895	2,365	2,998	3,499	5,405
8	1,397	1,860	2,306	2,896	3,355	5,041
9	1,383	1,833	2,262	2,821	3,250	4,781
10	1,372	1,812	2,228	2,764	3,169	4,587



## Оценка значимости отдельных коэффициентов. Критерий Стьюдента

```
tb = b1 / sb # критерий Стьюдента для коэффициента b1
tb

t0 = b0 / s0 # критерий Стьюдента для коэффициента b0

# sb и s0 - стандартные ошибки коэффициентов

sb = np.sqrt( Mso / np.sum((s - np.mean(s))**2))
sb
0.0033279211856704766

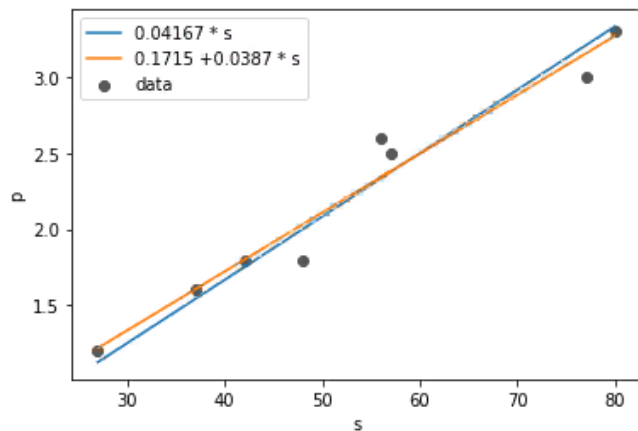
s0 = np.sqrt( (Mso * np.sum( s **2) ) / ( n * sum ( (s - np.mean(s))**2)))
s0
0.18558943

t0 = 0.17147009966776983 / 0.18558943
t0
0.92392171

tb = 0.03874584717607981 / 0.0033279211856704766
tb
11.642657687601963
```



## Графические модели линейной регрессии



Окончательная модель имеет вид  $\hat{y} = 0.1715 + 0.0387 * x$



# Логистическая регрессия

Логистическая регрессия применяется, когда  $y$  является бинарной переменной (0 или 1). Т.е. с помощью этого метода мы можем решить задачу бинарной классификации.

	x1 zp	x2 prod	x3 poezdki	y vozvrat
1	100	30	1	1
2	40	20	0	0
3	50	20	1	1
4	70	40	2	1
5	50	30	0	0
6	80	70	3	1
7	75	25	4	1

$$\text{modl} = -0.18839 + 0.01115 * x1 - 0.00279 * x2 + 0.16286 * x3$$

клинет 1

x1 = 68  
x2 = 27  
x3 = 2

modl  
0.8202

sigmoid = 1 / (1+ e<sup>(-modl)</sup>)  
sigmoid  
0.6937

клинет 2

x1 = 72  
x2 = 40  
x3 = 0

modl  
0.5028

sigmoid = 1 / (1+ e<sup>(-modl)</sup>)  
sigmoid  
0.6228

клинет 3

x1 = 120  
x2 = 40  
x3 = 1

modl  
1.2009

sigmoid = 1 / (1+ e<sup>(-modl)</sup>)  
sigmoid  
0.7680

$$\text{sigmoid} = \frac{1}{1 + e^{-\text{modl}}}$$





# Конец