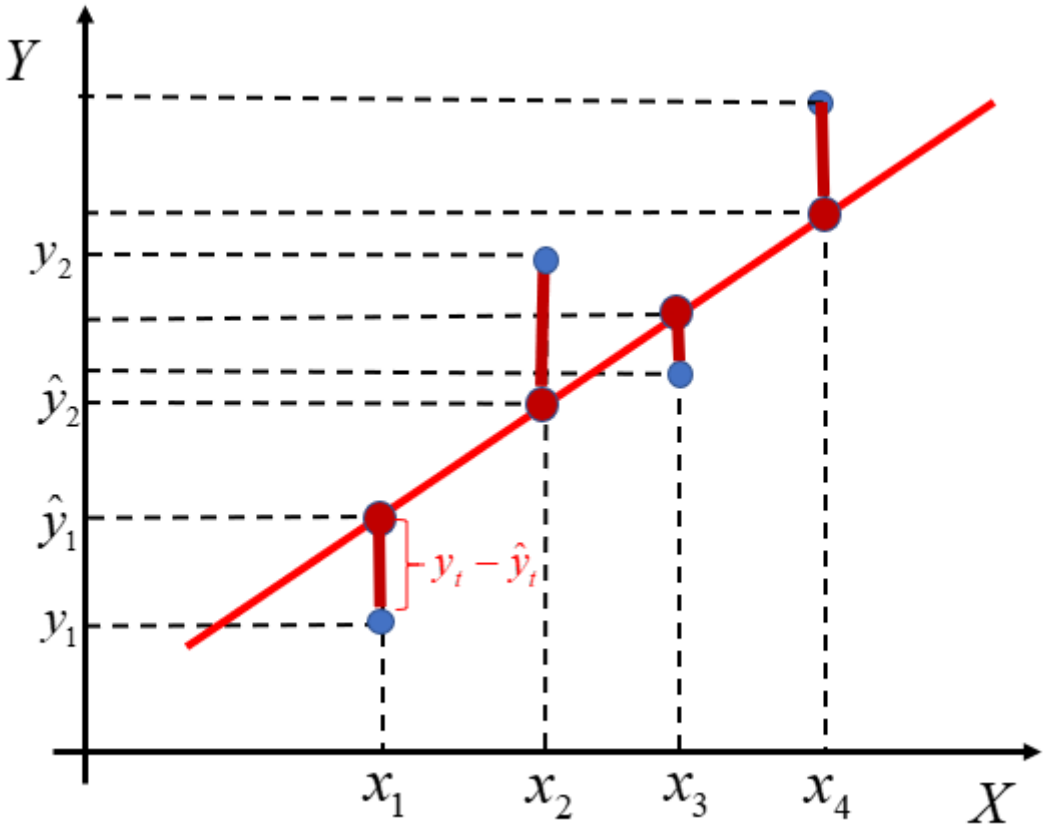
Metoda najmniejszych kwadratów

Teoria

Metoda najmniejszych kwadratów – standardowa metoda przybliżania rozwiązań układów nadokreślonych, tzn. zestawu równań, w którym jest ich więcej niż zmiennych. Nazwa „najmniejsze kwadraty” oznacza, że końcowe rozwiązanie tą metodą minimalizuje sumę kwadratów błędów przy rozwiązywaniu każdego z równań.

W statystyce wykorzystuje się ją do estymacji i wyznaczania linii trendu na podstawie zbioru danych w postaci par liczb. Najczęściej jest stosowana przy regresji liniowej, ale może też być stosowana do statystycznego wyznaczania parametrów nieliniowych linii trendu.

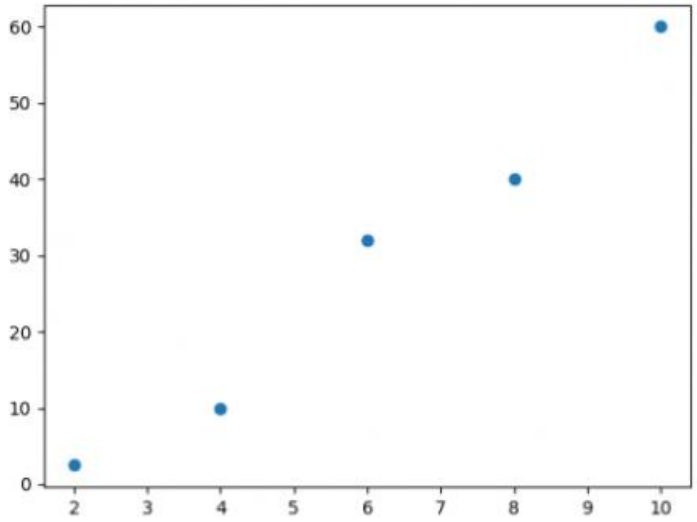


Wzór ogólny

$$S(a,b) = \sum_{i=1}^{n} [y_i - y(x_i)]^2 = \sum_{i=1}^{n} [y_i - (ax_i + b)]^2$$

Przykład zastosowania wzoru

index	x	y
1	2	2.5
2	4	10
3	6	32
4	8	40
5	10	60



Podstawienie danych z powyższego przykładu do wzoru ogólnego

$S(a,b)=[2.5-(2a+b)]^2+[10-(4a+b)]^2+[32-(6a+b)]^2+[40-(8a+b)]^2+[60-(10a+b)]^2$

Następnie szukamy minimum funkcji za pomocą pochodnych cząstkowych i warunku istnienia ekstremum.

Otrzymujemy:

$2[2.5-2a-b)](-2)+2[10-4a-b)](-4)+2[32-6a-b)](-6)+2[40-8a-b)](-8)+2[60-10a-b)](-10)=0$

oraz:

$2[2.5-2a-b)](-1)+2[10-4a-b)](-1)+2[32-6a-b)](-1)+2[40-8a-b)](-1)+2[60-10a-b)](-1)=0$

Po uproszczeniu:

440a + 60b = 2314

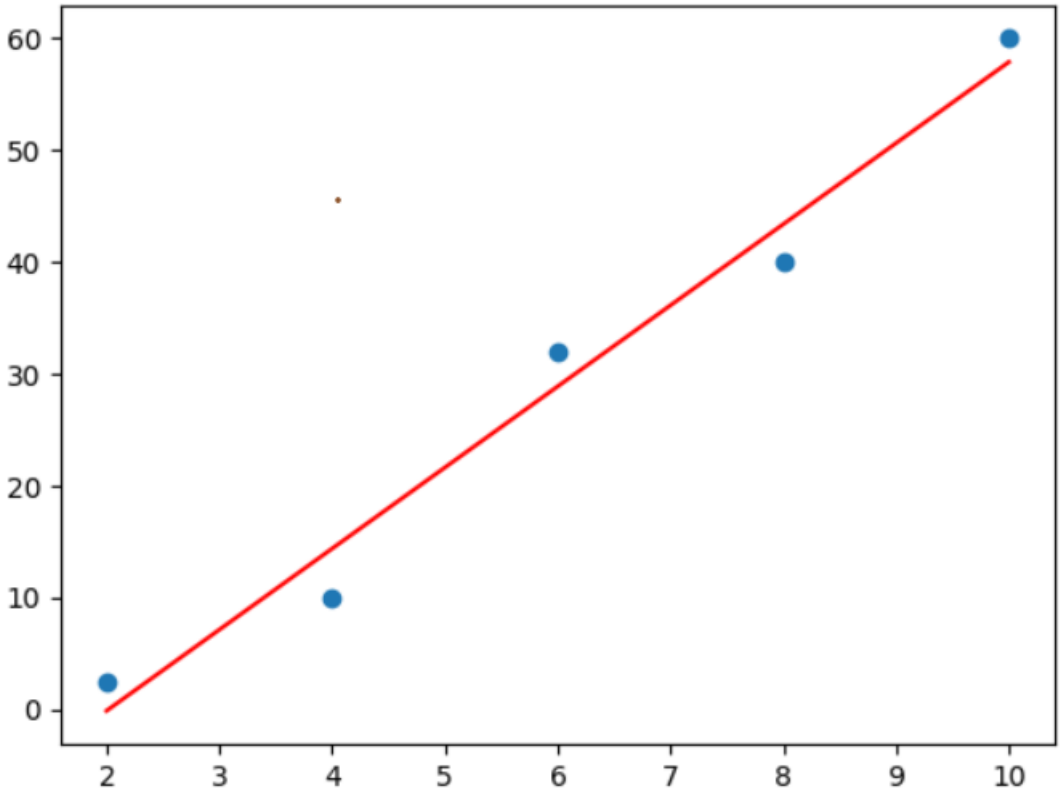
60a +10b = 289

Rozwiązanie układu równań:

a = 7.25

b = – 14.6

Na wykresie przedstawiono linię prostą określoną przez równanie y = 7.25 * x – 14.6



Regresja liniowa w python

Definicja

Regresja liniowa to metoda **przewidywania wartości** szukanej zmiennej *y* (zwanej zmienną zależną lub objaśnianą) przy znanych wartościach innych zmiennych *x* (zwanych niezależnymi lub objaśniającymi).

Metoda ta zakłada, że zależność między zmienną objaśniającą a objaśnianą jest **zależnością liniową**, dlatego funkcja regresji liniowej przyjmuje postać **funkcji liniowej**:

$$y=bx+a$$

Analiza regresji liniowej ma na celu wyliczenie takich współczynników **b (ang. slope) i a (ang. intercept)**, aby funkcja **najlepiej przewidywała zmienną objaśnianą**. Linia będąca wynikiem tej funkcji nazywa się **linią najlepiej pasującą** (ang. best-fitting line) i jest **modelem regresji liniowej**.

Pandas

Pandas to biblioteka oprogramowania napisana dla języka programowania Python do manipulacji i analizy danych . W szczególności oferuje struktury danych i operacje do manipulowania tabelami numerycznymi i szeregami czasowymi.

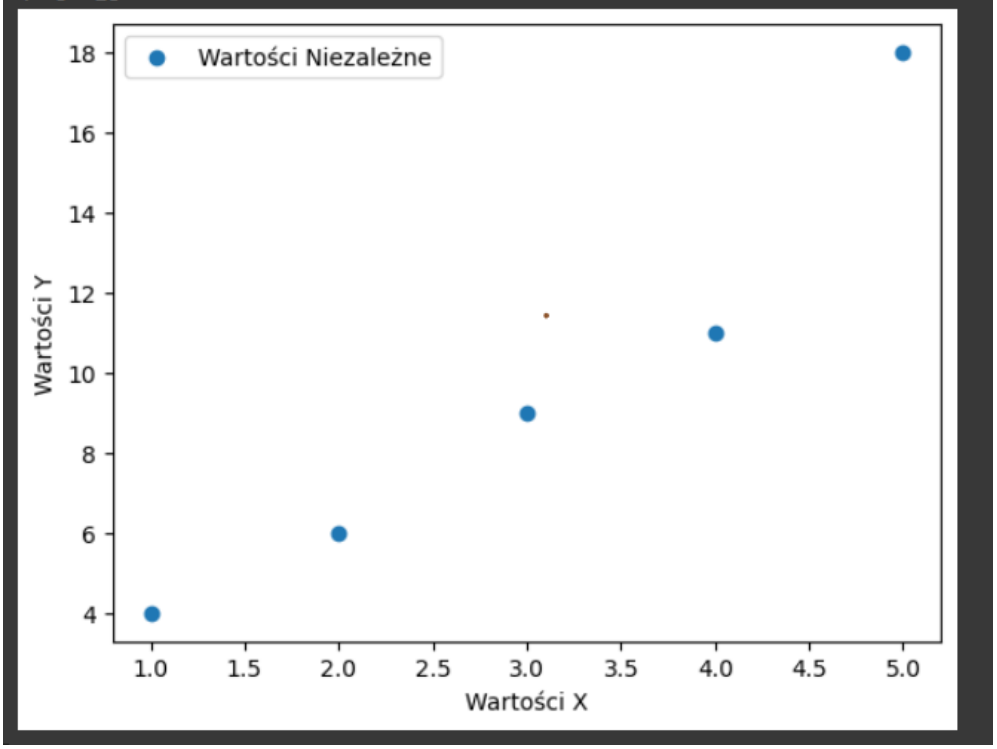
Pierwsze kroki z biblioteką Pandas

Załóżmy że mamy zbiory X={1,2,3,4,5} i Y={4,6,9,11,18}, z których chcemy stworzyć model regresji. Pierwszym krokiem jest przekształcenie danych na DataFrame (dwuwymiarowa struktura danych z oznaczonymi rzędami i kolumnami– forma tabeli), na którym będzie nam łatwiej i przyjemniej pracować.

```
%matplotlib inline
import pandas as pd
import matplotlib.pyplot as plt

df = pd.DataFrame()
df['X'] = [1, 2, 3, 4, 5]
df['Y'] = [4, 6, 9, 11, 18]
print(df)
plt.scatter(df['X'], df['Y'], label='Wartości Niezależne')
plt.xlabel('Wartości X')
plt.ylabel('Wartości Y')
plt.legend()
plt.show()
```

```
   X   Y
0  1   4
1  2   6
2  3   9
3  4  11
4  5  18
```



Aby wyznaczyć najlepiej pasującą linię, musimy obliczyć współczynniki b i a. Jednak zanim to zrobimy musimy przygotować sobie parę zmiennych:

- **Średnie zbiorów X i Y** (Mx, My)
- **Odchylenie standardowe (z próby) zbiorów X i Y** (Sx, Sy)
- **Współczynnik korelacji (Pearsona) zbiorów X i Y** (r)

Obliczanie średniej arytmetycznej

Średnia arytmetyczna zbioru to suma wszystkich jego elementów podzielona przez ich ilość. Określa **średnią wartość elementów danego zbioru**. Opisuje ją wzór:

$$M_A = \frac{a_1 + a_2 + \ldots + a_n}{n}$$

Więc średnie zbiorów X i Y możemy obliczyć tak:

$$Mx = \frac{1+2+3+4+5}{5} = \frac{15}{5} = 3$$
$$My = \frac{4,6,9,11,18}{5} = \frac{48}{5} = 9.6$$

Funkcję średniej aplikujemy na Zbiór X i Y.

```
def srednia(      ):
    [redacted]

Mean_x = srednia(df['X'])
Mean_y = srednia(df['Y'])

print("Mean x: ", Mean_x)
print("Mean y: ", Mean_y)
```

```
Mean x:  3.0
Mean y:  9.6
```

Sprawdzenie wartości średniej:

```
import numpy as np

srednia_x = np.mean(df['X'])
srednia_y = np.mean(df['Y'])
```

Obliczanie odchylenia standardowego z próby

Odchylenie standardowe to jedna z miar zmienności zbioru. **Służy ono do estymowania o ile średnio elementy zbioru różnią się od średniej tego zbioru**. Obliczamy je za pomocą poniższego wzoru:

$$s_A = \sqrt{\frac{(a_1 - M_A)^2 + (a_2 - M_A)^2 + \ldots + (a_n - M_A)^2}{n-1}}$$

Powyższy wzór jest skomplikowany, ale bardzo łatwy do zaimplementowania, gdy znamy już **średnią zbioru**. Odchylenie standardowe policzymy tak:

$$s_X = \sqrt{\frac{(1-3)^2 + (2-3)^2 + (3-3)^2 + (4-3)^2 + (5-3)^2}{5-1}} = \sqrt{\frac{10}{4}} \approx 1.58$$

$$s_Y = \sqrt{\frac{(4-9.6)^2 + (6-9.6)^2 + (9-9.6)^2 + (11-9.6)^2 + (18-9.6)^2}{5-1}} = \sqrt{\frac{117.2}{4}} \approx 5.41$$

Powyższą funkcje odchylenia standardowego aplikujemy na zbiór X i Y.

```
from math import sqrt

def odchylenie([redacted]):
    [redacted]
    for [redacted]
        [redacted] += ([redacted])
    return sqrt([redacted])

Sx = odchylenie([redacted])
Sy = odchylenie([redacted])

print("Standard deviation x: ", Sx)
print("Standard deviation y: ", Sy)

Standard deviation x:  1.5811388300841898
Standard deviation y:  5.412947441089743
```

```
import numpy as np

Sx = np.std(df['X'])
Sy = np.std(df['Y'])

print("Standard deviation of x and y: ", Sx,Sy)

Standard deviation of x and y:  1.4142135623730951 4.841487374764082
```

Wyniki odchylenia z funkcji wbudowanej a wyliczonej przez nas mogą się nieznacznie różnić.

Obliczanie współczynnika korelacji Pearsona

Współczynnik korelacji liniowej Pearsona **służy do określenia związku liczb z dwóch zbiorów**. Może przyjmować wartości od -1 do 1. Jeśli:

Współczynnik korelacji jest **większy od 0**, to znaczy, że między elementami danych zbiorów istnieje **korelacja dodatnia**. Oznacza to, że wraz ze wzrostem wartości elementów pierwszego zbioru, rosną wartości elementów drugiego zbioru. Korelacja ta jest silniejsza, gdy współczynnik ma wartość bliższą **1**.

Współczynnik korelacji jest **mniejszy od 0**, to znaczy, że między elementami danych zbiorów istnieje **korelacja ujemna**. Oznacza to, że wraz ze wzrostem wartości elementów pierwszego zbioru, maleją wartości elementów drugiego zbioru. Korelacja ta jest silniejsza, gdy współczynnik ma wartość bliższą **-1**.

Współczynnik korelacji jest **równy 0**, to znaczy, że między elementami danych zbiorów **nie ma korelacji**.

W skrócie:

- r > 0 - korelacja dodatnia
- r < 0 - korelacja ujemna
- r = 0 - brak korelacji
-

Współczynnik korelacji liniowej Pearsona opisuje wzór:

$$r_{XY} = \frac{n \times (x_1y_1 + x_2y_2 + \ldots + x_ny_n) - (x_0 + x_1 + \ldots + x_n) \times (y_0 + y_1 + \ldots + y_n)}{\sqrt{[n \times (x_0^2 + x_1^2 + \ldots + x_n^2) - (x_0 + x_1 + \ldots + x_n)^2] \times [n \times (y_0^2 + y_1^2 + \ldots + y_n^2) - (y_0 + y_1 + \ldots + y_n)^2]}}$$

Sam wzór jeśli chodzi o zapis jest dość skomplikowany, jednak okazuje się, że obliczanie przy prawidłowym rozszerzeniu naszego DataFrame nie przysporzy nam większych kłopotów.

Dodajemy nowe kolumny, w których umieszczamy wyliczone dane:

1. Iloczyn każdych dwóch odpowiadających elementów zbioru X i Y (xy)
2. Kwadrat każdego elementu ze zbioru X (x2)
3. Kwadrat każdego elementu ze zbioru Y (y2)
4. Sumy wszystkich elementów i wyżej wymienionych wartości dla każdego zbioru (Σ)
5. Ilość elementów w zbiorach (n)

```
n = len(df['X'])

pearson = pd.DataFrame(df[:]) #creating new data frame based on first DF
pearson['y2'] = df['Y'] * df['Y'] # add columns as in pdf points
pearson['xy'] = df['X'] * df['Y']
pearson['x2'] = df['X'] * df['X']
pearson['y2'] = df['Y'] * df['Y']
pearson.loc['sum'] = pearson.sum()  # add row conaining sum of each column

print("n = ", n)
print()
print(pearson)
```

```
n =  5

      X   Y   y2   xy  x2
0     1   4   16    4   1
1     2   6   36   12   4
2     3   9   81   27   9
3     4  11  121   44  16
4     5  18  324   90  25
sum  15  48  578  177  55
```

Posiadając potrzebne wartości, możemy uprościć nasz wzór:

$$r_{XY} = \frac{n \times \sum x_i y_i - \sum x_i \times \sum y_i}{\sqrt{[n \times \sum x_i^2 - (\sum x_i)^2] \times [n \times \sum y_i^2 - (\sum y_i)^2]}}$$

Po podstawieniu wartości do wzoru:

$$r_{XY} = \frac{5 \times 177 - 15 \times 48}{\sqrt{[5 \times 55 - (15)^2] \times [5 \times 578 - (48)^2]}} = \frac{165}{\sqrt{29300}} \approx 0.963...$$

```
def wsp_korelacji_pearsona(                                    ):
    return ( (
             (                                                  )

pearson_result = wsp_korelacji_pearsona
print("r = ", pearson_result)
```

```
r =  0.963940292431027
```

```
import scipy

pearson_scipy = scipy.stats.pearsonr(df['X'],df['Y'])
print(pearson_scipy[0])
```

```
0.963940292431027
```

Podsumowanie otrzymanych wyników

```
print("Mean x: ", Mean_x)
print("Mean y: ", Mean_y, "\n")
print("Standard deviation x: ", Sx)
print("Standard deviation y: ", Sy, "\n")
print("Pearson correlation coefficient = ", pearson_result)
```

```
Mean x:  3.0
Mean y:  9.6

Standard deviation x:  1.4142135623730951
Standard deviation y:  4.841487374764082

Pearson correlation coefficient =  0.963940292431027
```

Obliczanie najlepiej pasującej linii

Wzór funkcji liniowej: $y=bx+a$

Obliczamy współczynniki a i b.

$$b = r \times \frac{Sy}{Sx}$$
$$a = My - b \times Mx$$

Po podstawieniu wartości:

$$b = 0.96 \times \frac{5.41}{1.58} = 3.28$$
$$a = 9.6 - 3.28 \times 3 = -0,24$$

Gdzie:

- M x/y – wyliczona wartość średnia dla x lub y
- S x/y – wyliczone odchylenie standardowe dla x lub y
- r – wyliczony współczynnik korelacji Pearsna

Otrzymana funkcja liniowa :

$$y = 3.28 * x - 0.24$$

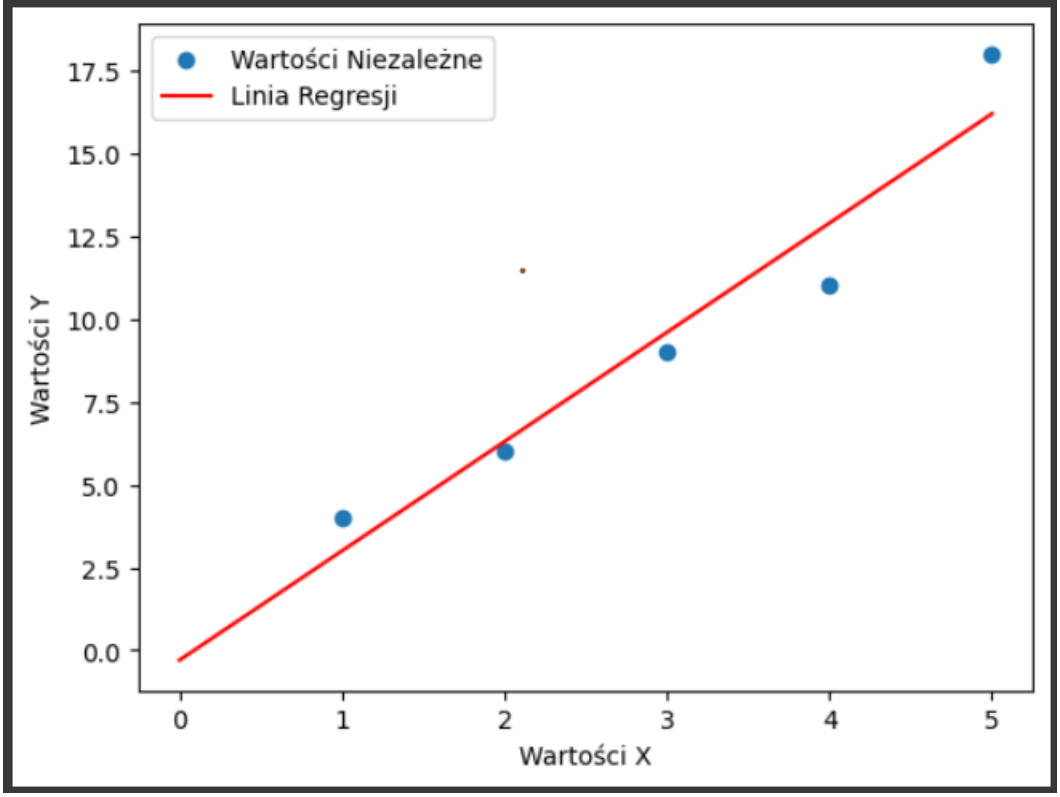
```python
import numpy as np

b = 
a = 
print("b = ", b)
print("a = ", a)

def linia_regresji(x):
    return (b * x) + a

x = np.linspace(0, 5,1000)
plt.scatter(df['X'], df['Y'], label='Wartości Niezależne')
plt.plot(x,linia_regresji(x), 'r', label='Linia Regresji')
plt.xlabel('Wartości X')
plt.ylabel('Wartości Y')
plt.legend()
plt.show()
```

```
b =  3.3
a =  -0.29999999999999893
```



Przewidywanie przyszłych wartości

1. Dodajemy nowy element do zbioru X:
   X={1,2,3,4,5,6}
   Y={4,6,9,11,18}

```
df = df.append({'X': 6, 'Y': np.nan}, ignore_index=True)
df
```

	X	Y
0	1.0	4.0
1	2.0	6.0
2	3.0	9.0
3	4.0	11.0
4	5.0	18.0
5	6.0	NaN
6	6.0	NaN
7	6.0	NaN

2. Przewidujemy odpowiadającą wartość dla zbioru Y(podstawiamy do wzoru na funkcję liniową) :

$$y=3.3\times6-0.29=19,51$$

```
def predict_y(x, b, a):
    return b * x + a

df.at[5, 'Y'] = predict_y(df['X'][5], b, a)
df
```

	X	Y
0	1.0	4.0
1	2.0	6.0
2	3.0	9.0
3	4.0	11.0
4	5.0	18.0
5	6.0	19.5

Zadania do wykonania:

Zadanie 1

Wykonaj wszystkie kroki po kolei w celu stworzenia prostego klasyfikatora wykorzystującego regresję liniową. W tym celu zaimplementuj samodzielnie funkcje średniej, odchylenia standardowego, współczynnika korelacji Pearsona.

Posłuż się zbiorami danych x i y z omawianego przykładu.
Dodatkowo przewidź wartości dla X=7 oraz X=8.