# lin_reg

July 18, 2017

## 1 Python

. , , Python, .

- 
- NumPy SciPy
- Matplotlib
- Pandas
- Pandas Cheat Sheet
- Seaborn

### 1.1 1. c Pandas

SOCR 25 .

**[1].** Seaborn - *conda install seaborn*. (Seaborn Anaconda, ).

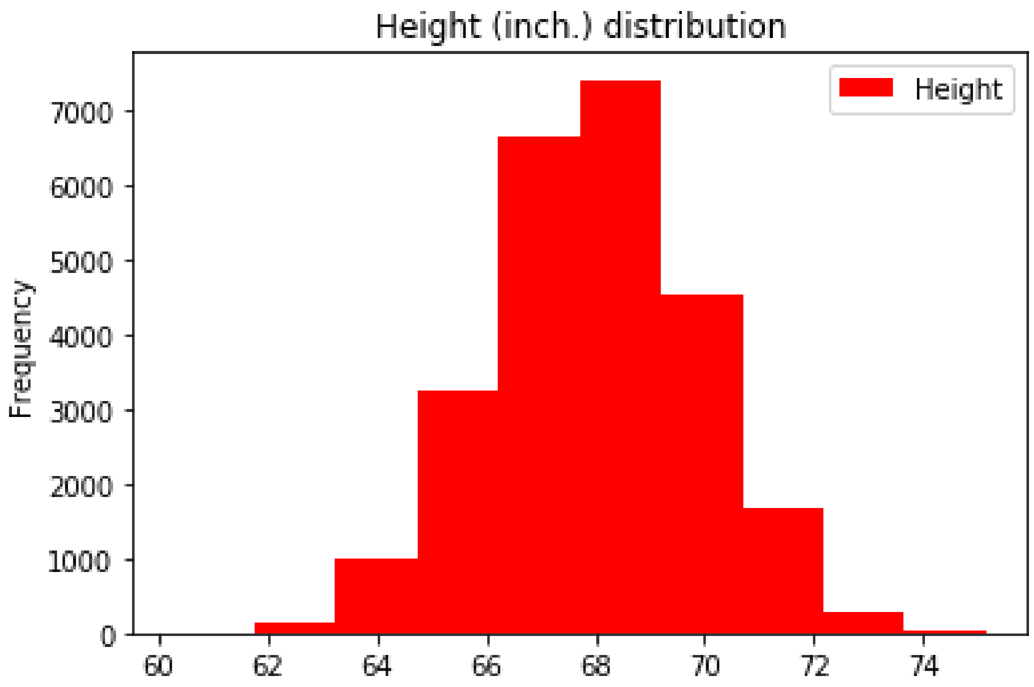
```
In [2]: import numpy as np
        import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        %matplotlib inline
```

(*weights_heights.csv*, ) Pandas DataFrame:

```
In [3]: data = pd.read_csv('weights_heights.csv', index_col='Index')
```

, - . (, 10 , 9 ). , , (, ..). - - ( , , - ). - , - "" . *plot* Pandas DataFrame *kind='hist'*. . *data*. *plot* DataFrame *data* c *y='Height'* ( , )

```
In [4]: data.plot(y='Height', kind='hist',
                  color='red',  title='Height (inch.) distribution')

Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x7f50a0f3d450>
```

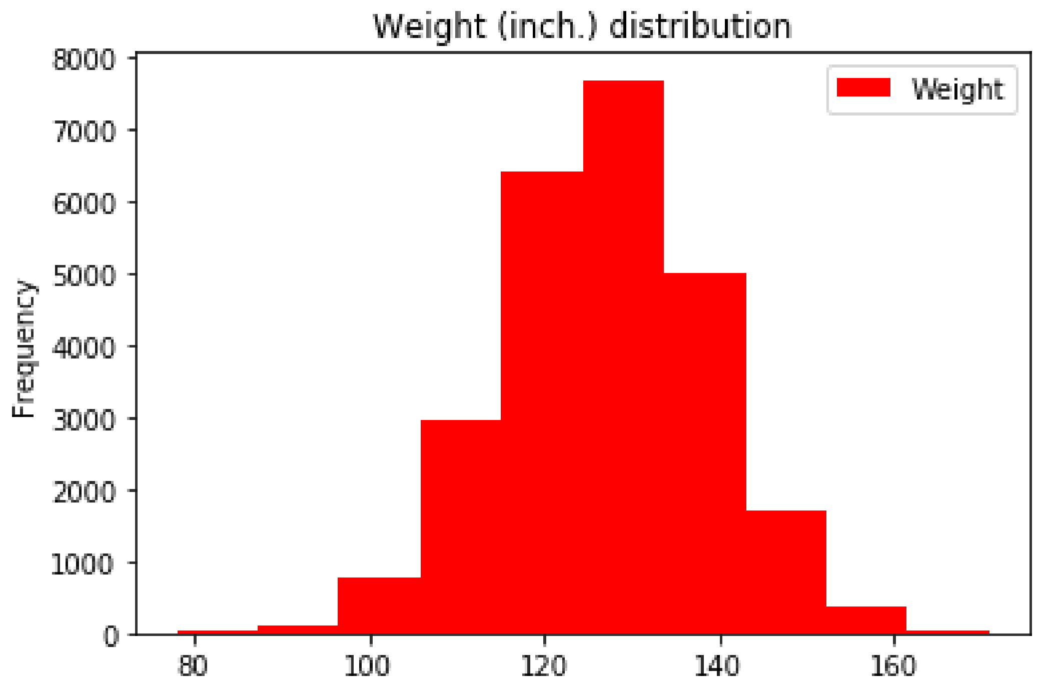


:

- *y='Height'* - ,
- *kind='hist'* - ,
- *color='red'* -

**[2]**. 5 *head* Pandas DataFrame. *plot* Pandas DataFrame. , .

```
In [5]: data.head()

Out[5]:          Height     Weight
        Index
        1       65.78331  112.9925
        2       71.51521  136.4873
        3       69.39874  153.0269
        4       68.21660  142.3354
        5       67.78781  144.2971

In [6]: data.plot(y='Weight', kind='hist',
                  color='red',  title='Weight (inch.) distribution')

Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x7f509e937050>
```



- . $m \times m$ ($m$ - ), , - scatter plots . *scatter_matrix* Pandas Data Frame *pairplot* Seaborn.
, . (BMI). *apply* Pandas DataFrame lambda- Python.

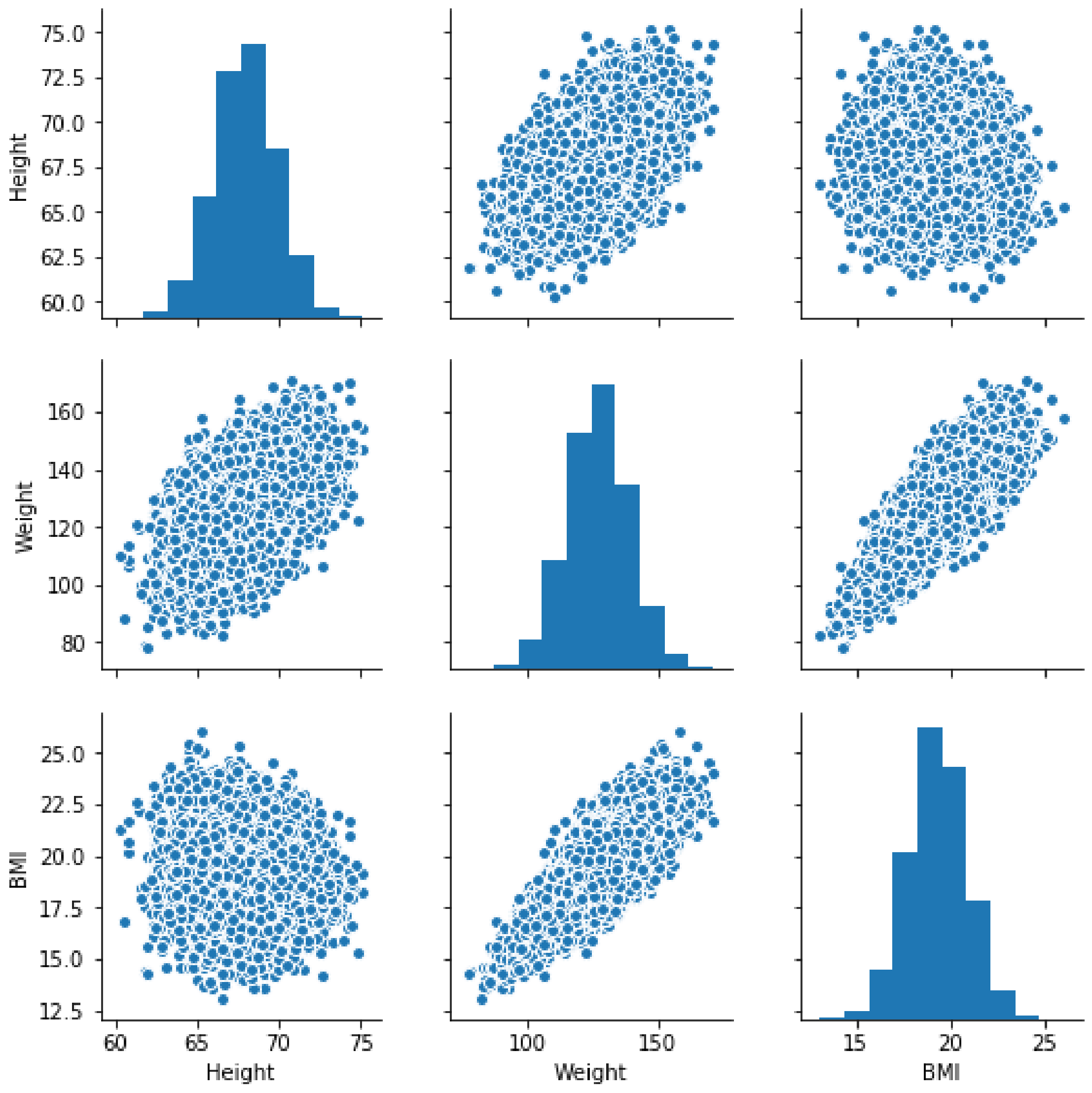
```
In [7]: def make_bmi(height_inch, weight_pound):
            METER_TO_INCH, KILO_TO_POUND = 39.37, 2.20462
            return (weight_pound / KILO_TO_POUND) / \
                   (height_inch / METER_TO_INCH) ** 2
```

```
In [8]: data['BMI'] = data.apply(lambda row: make_bmi(row['Height'],
                                                       row['Weight']), axis=1)
```

**[3].** , , 'Height', 'Weight' 'BMI' . *pairplot* Seaborn.

```
In [9]: sns.pairplot(data)
```

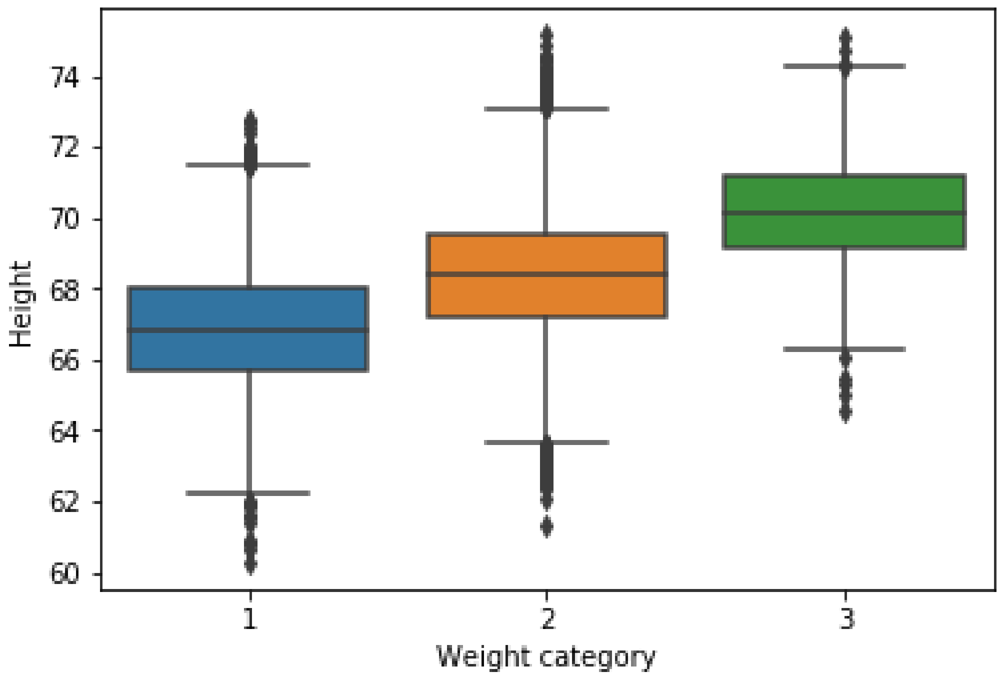
```
Out[9]: <seaborn.axisgrid.PairGrid at 0x7f509eb48550>
```



- (, ). " " - boxplots Seaborn. Box plot - ( ) . "" - , .

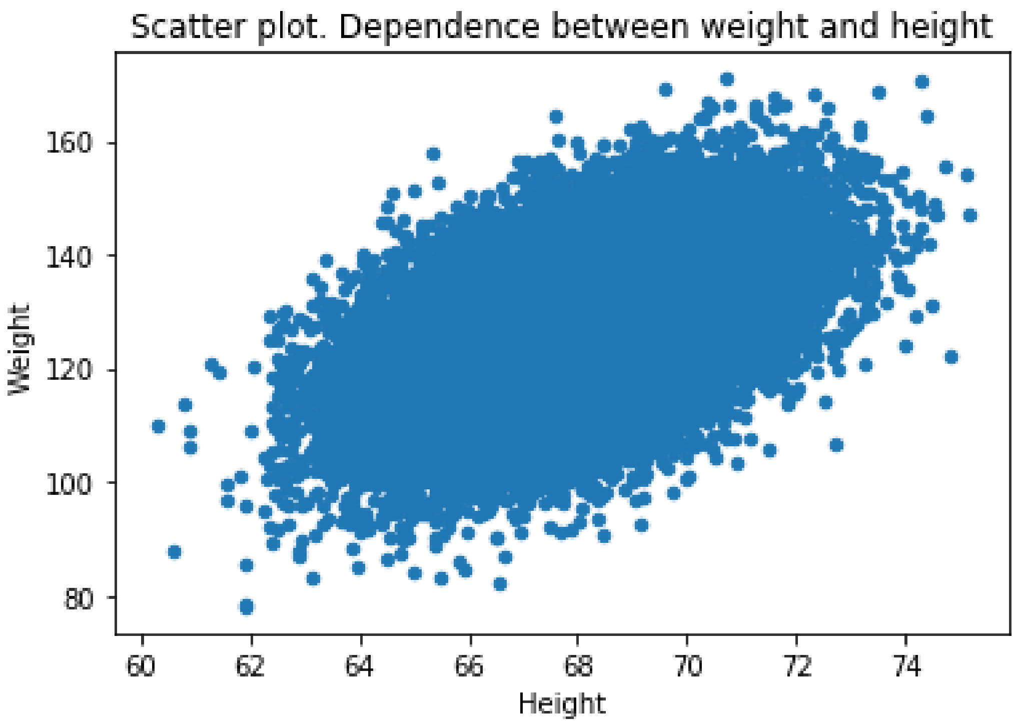
**[4]**. DataFrame *data* *weight_category*, 3 : 1 – 120 . (~ 54 .), 3 - 150 (~68 .), 2 – . ń ż (boxplot), . *boxplot* Seaborn *apply* Pandas DataFrame. *y* ńż, *x* – ń ż.

```
In [10]: def weight_category(weight):
             pass
             if weight < 120:
                 return 1
             if weight >=150:
                 return 3
             else:
                 return 2

         data['weight_cat'] = data['Weight'].apply(weight_category)
         chart = sns.boxplot(x=data['weight_cat'], y=data['Height'])
         chart.set(xlabel='Weight category', ylabel='Height')

Out[10]: [<matplotlib.text.Text at 0x7f509e49be50>,
          <matplotlib.text.Text at 0x7f509e5152d0>]
```



**[5].** scatter plot , *plot* Pandas DataFrame *kind='scatter'*. .

```
In [11]: data.plot(x='Height', y='Weight', kind='scatter', title = 'Scatter plot. Dependence bet

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x7f509e64d390>
```



## 1.2 2.

( ) .

**[6].** , $w_0$ $w_1$ $y$ $x$ $y = w_0 + w_1 * x$:

$$error(w_0, w_1) = \sum_{i=1}^{n} (y_i - (w_0 + w_1 * x_i))^2$$

$n -$ , $y_i$ $x_i -$ $i$- .

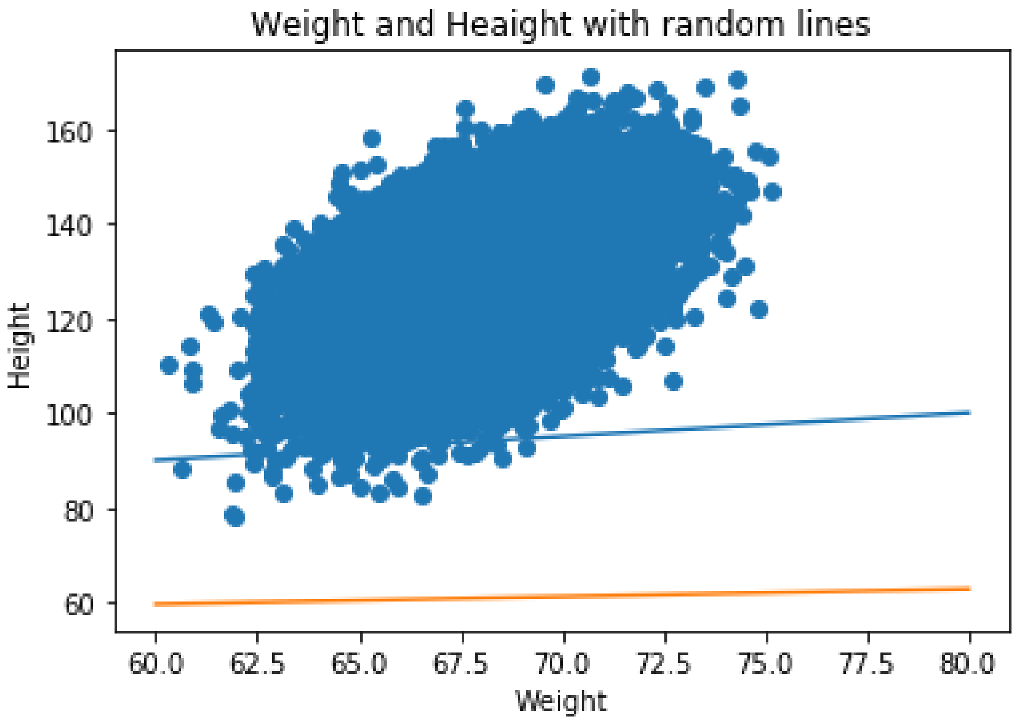
```
In [12]: def squar_error(w):
             return sum((data['Height']-(w[0] + w[1]*data['Weight']))**2)
```

, : , , "" "" , . 6. - , .

**[7].** . 5 1 , $(w_0, w_1) = (60, 0.05)$ $(w_0, w_1) = (50, 0.16)$. *plot* *matplotlib.pyplot*, *linspace* NumPy. .

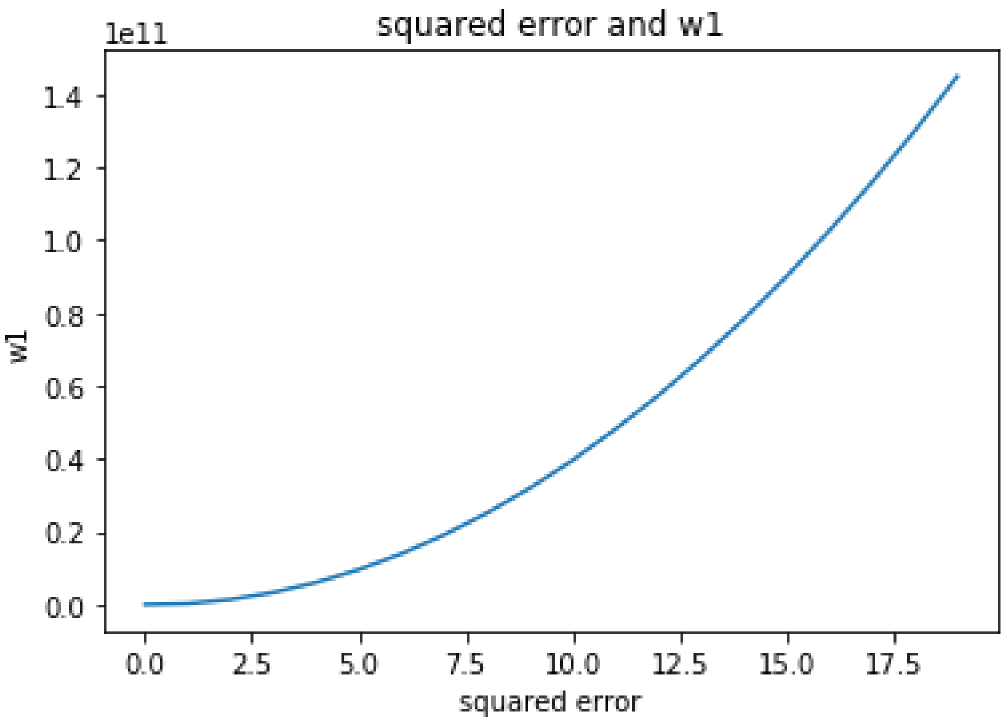
```
In [13]: plt.scatter(data['Height'], data['Weight'])
         X_plot = np.linspace(60,80,20)
         plt.plot(X_plot, 0.5*X_plot + 60)
         plt.plot(X_plot, 0.16*X_plot + 50)
         plt.xlabel('Weight')
         plt.ylabel('Height')
         plt.title('Weight and Heaight with random lines')
```

```
Out[13]: <matplotlib.text.Text at 0x7f509c95ed50>
```



- , . ., ( ), ( ) .
**[8].** , . 6, $w_1$ $w_0 = 50$. .

```
In [14]: plt_w1 = []
         plt_error = []
         for i in range(20):
             plt_w1.append(i)
             plt_error.append(squar_error([50,i]))
         plt.plot(plt_w1,plt_error)
         plt.xlabel('squared error')
         plt.ylabel('w1')
         plt.title('squared error and w1')

Out[14]: <matplotlib.text.Text at 0x7f509ca02250>
```



"" , , $w_0 = 50$.

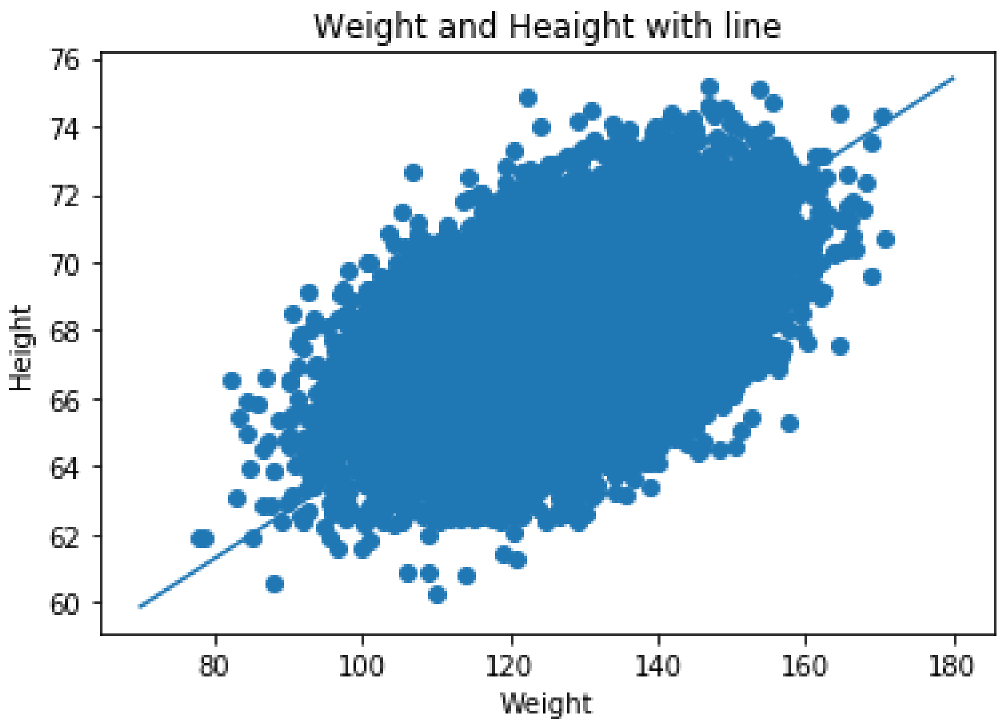
**[9].** *minimize_scalar* *scipy.optimize* , . 6, $w_1$ [-5,5]. . 5 1 , $(w_0, w_1) = (50, w_1\_opt)$, $w_1\_opt$ – . 8 $w_1$.

```
In [15]: from scipy.optimize import minimize_scalar

         w1_opt = minimize_scalar(lambda w1: squar_error([50,w1]), bounds=(-5, 5), method='bound
         w1_opt
```

```
Out[15]:      fun: 79512.217286994884
          message: 'Solution found.'
             nfev: 6
           status: 0
          success: True
                x: 0.14109203728834441
```

```
In [16]: plt.scatter(data['Weight'], data['Height'])
         X_plot = np.linspace(70,180,20)
         plt.plot(X_plot, w1_opt.x*X_plot + 50)
         plt.xlabel('Weight')
         plt.ylabel('Height')
         plt.title('Weight and Heaight with line')
```

```
Out[16]: <matplotlib.text.Text at 0x7f509c8ca050>
```

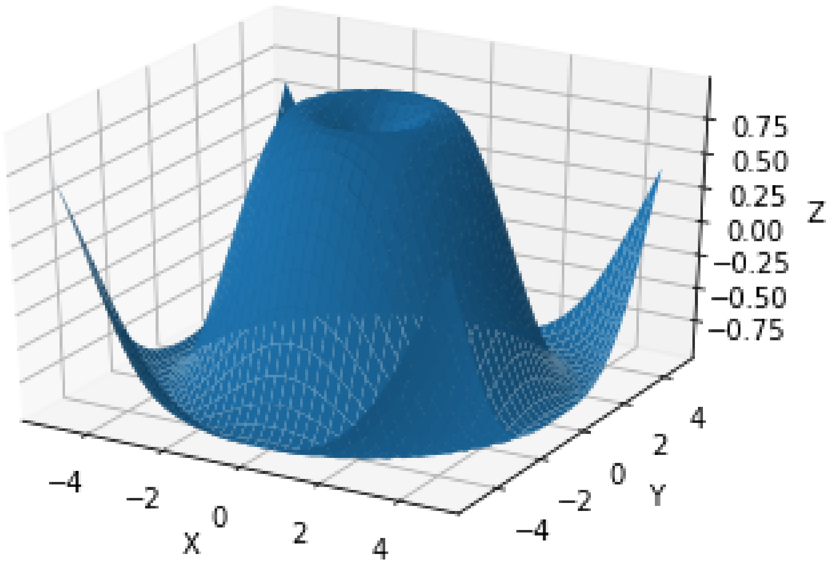


. ,   3   .   2D  3D   2 , , 3  (   -   )   .
, Python 3D ,  $z(x,y) = sin(\sqrt{x^2+y^2})$ $x$ $y$ [-5,5] c 0.25.

```
In [17]: from mpl_toolkits.mplot3d import Axes3D
```

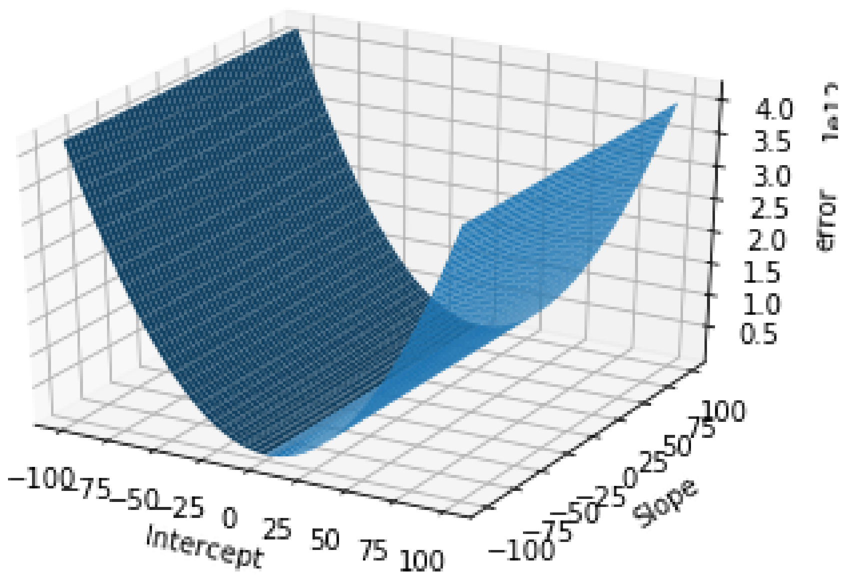
matplotlib.figure.Figure ()  matplotlib.axes._subplots.Axes3DSubplot ().

```
In [18]: fig = plt.figure()
         ax = fig.gca(projection='3d') # get current axis

         #    NumPy       X   .
         #    meshgrid,
         #    .    Z(x, y).
         X = np.arange(-5, 5, 0.25)
         Y = np.arange(-5, 5, 0.25)
         X, Y = np.meshgrid(X, Y)
         Z = np.sin(np.sqrt(X**2 + Y**2))

         # ,   *plot_surface*
         #  Axes3DSubplot.    .
         surf = ax.plot_surface(X, Y, Z)
         ax.set_xlabel('X')
         ax.set_ylabel('Y')
         ax.set_zlabel('Z')
         plt.show()
```



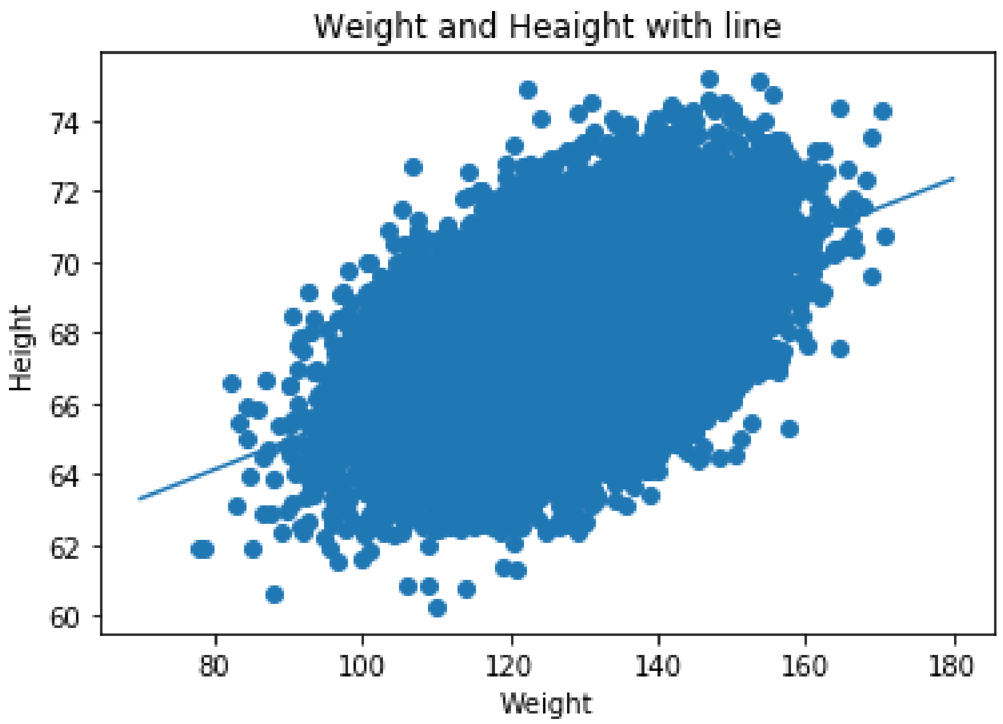
**[10].** 3D- , .6 $w_0$ $w_1$. $x$ ńInterceptż, $y$ – ńSlopeż, a $z$ – ńErrorż.

```
In [19]: fig = plt.figure()
         ax = fig.gca(projection='3d') # get current axis

         X = np.arange(-100, 100, 1)
         Y = np.arange(-100, 100, 1)
         Z = [squar_error([X[i],Y[i]]) for i in range(len(X))]
         X, Y = np.meshgrid(X, Y)


         surf = ax.plot_surface(X, Y, Z)
         ax.set_xlabel('Intercept')
         ax.set_ylabel('Slope')
         ax.set_zlabel('error')
         plt.show()
```



**[11].** *minimize* scipy.optimize , . 6, $w_0$ [-100,100] $w_1$ - [-5, 5]. – $(w_0, w_1) = (0, 0)$. L-BFGS-B ( method minimize). . 5 1 , $w_0$ $w_1$. .

```
In [20]: from scipy.optimize import minimize
         w_opt = minimize(squar_error,x0=(0.,0.), bounds=((-100,100),(-5,5)), method='L-BFGS-B')
         w_opt

Out[20]:       fun: 67545.287085690099
          hess_inv: <2x2 LbfgsInvHessProduct with dtype=float64>
               jac: array([ 0.01309672,  0.13824319])
           message: 'CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH'
              nfev: 51
               nit: 12
            status: 0
           success: True
                 x: array([ 57.57175421,   0.08200666])

In [22]: plt.scatter(data['Weight'], data['Height'])
         X_plot = np.linspace(70,180,20)
         plt.plot(X_plot, (w_opt.x[0] + w_opt.x[1]*X_plot))
         plt.xlabel('Weight')
         plt.ylabel('Height')
         plt.title('Weight and Heaight with line')

Out[22]: <matplotlib.text.Text at 0x7f509c4e7050>
```



## 1.3

- IPython ? (15 )
- . 2? (3 ). ? (1 )
- . 3? (3 ). ? (1 )
- . 4? (3 ). ? (1 )
- scatter plot . 5? (3 ). ? (1 )
- . 6? (10 )
- . 7? (3 ) ? (1 )
- . 8? (3 ) ? (1 )
- minimize_scalar scipy.optimize? (6 ). . 9? (3 ) ? (1 )
- 3D- . 10? (6 ) ? (1 )
- minimize scipy.optimize? (6 ). . 11? (3 ). ? (1 )