Zad 1

```
import pandas as pd
import numpy as np
data = pd.read_csv('Boston.csv')
print(data)
```

```
     Unnamed: 0     crim    zn  indus  chas    nox     rm   age     dis  rad  \
0             1  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1
1             2  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2
2             3  0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2
3             4  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3
4             5  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3
..          ...      ...   ...    ...   ...    ...    ...   ...     ...  ...
501         502  0.06263   0.0  11.93     0  0.573  6.593  69.1  2.4786    1
502         503  0.04527   0.0  11.93     0  0.573  6.120  76.7  2.2875    1
503         504  0.06076   0.0  11.93     0  0.573  6.976  91.0  2.1675    1
504         505  0.10959   0.0  11.93     0  0.573  6.794  89.3  2.3889    1
505         506  0.04741   0.0  11.93     0  0.573  6.030  80.8  2.5050    1

     tax  ptratio   black  lstat  medv
0    296     15.3  396.90   4.98  24.0
1    242     17.8  396.90   9.14  21.6
2    242     17.8  392.83   4.03  34.7
3    222     18.7  394.63   2.94  33.4
4    222     18.7  396.90   5.33  36.2
..   ...      ...     ...    ...   ...
501  273     21.0  391.99   9.67  22.4
502  273     21.0  396.90   9.08  20.6
503  273     21.0  396.90   5.64  23.9
504  273     21.0  393.45   6.48  22.0
505  273     21.0  396.90   7.88  11.9

[506 rows x 15 columns]
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  506 non-null    int64
 1   crim        506 non-null    float64
 2   zn          506 non-null    float64
 3   indus       506 non-null    float64
 4   chas        506 non-null    int64
 5   nox         506 non-null    float64
 6   rm          506 non-null    float64
 7   age         506 non-null    float64
 8   dis         506 non-null    float64
 9   rad         506 non-null    int64
 10  tax         506 non-null    int64
 11  ptratio     506 non-null    float64
 12  black       506 non-null    float64
 13  lstat       506 non-null    float64
```

```
 14  medv        506 non-null    float64
dtypes: float64(11), int64(4)
memory usage: 59.4 KB
```

```
P1=pd.DataFrame(data=data,columns=['crim'],copy=True)
print(P1)
```

```
        crim
0    0.00632
1    0.02731
2    0.02729
3    0.03237
4    0.06905
..       ...
501  0.06263
502  0.04527
503  0.06076
504  0.10959
505  0.04741

[506 rows x 1 columns]
```

```
np.array(data.loc[:,"rm"])
```

```
array([6.575, 6.421, 7.185, 6.998, 7.147, 6.43 , 6.012, 6.172, 5.631,
       6.004, 6.377, 6.009, 5.889, 5.949, 6.096, 5.834, 5.935, 5.99 ,
       5.456, 5.727, 5.57 , 5.965, 6.142, 5.813, 5.924, 5.599, 5.813,
       6.047, 6.495, 6.674, 5.713, 6.072, 5.95 , 5.701, 6.096, 5.933,
       5.841, 5.85 , 5.966, 6.595, 7.024, 6.77 , 6.169, 6.211, 6.069,
       5.682, 5.786, 6.03 , 5.399, 5.602, 5.963, 6.115, 6.511, 5.998,
       5.888, 7.249, 6.383, 6.816, 6.145, 5.927, 5.741, 5.966, 6.456,
       6.762, 7.104, 6.29 , 5.787, 5.878, 5.594, 5.885, 6.417, 5.961,
       6.065, 6.245, 6.273, 6.286, 6.279, 6.14 , 6.232, 5.874, 6.727,
       6.619, 6.302, 6.167, 6.389, 6.63 , 6.015, 6.121, 7.007, 7.079,
       6.417, 6.405, 6.442, 6.211, 6.249, 6.625, 6.163, 8.069, 7.82 ,
       7.416, 6.727, 6.781, 6.405, 6.137, 6.167, 5.851, 5.836, 6.127,
       6.474, 6.229, 6.195, 6.715, 5.913, 6.092, 6.254, 5.928, 6.176,
       6.021, 5.872, 5.731, 5.87 , 6.004, 5.961, 5.856, 5.879, 5.986,
       5.613, 5.693, 6.431, 5.637, 6.458, 6.326, 6.372, 5.822, 5.757,
       6.335, 5.942, 6.454, 5.857, 6.151, 6.174, 5.019, 5.403, 5.468,
       4.903, 6.13 , 5.628, 4.926, 5.186, 5.597, 6.122, 5.404, 5.012,
       5.709, 6.129, 6.152, 5.272, 6.943, 6.066, 6.51 , 6.25 , 7.489,
       7.802, 8.375, 5.854, 6.101, 7.929, 5.877, 6.319, 6.402, 5.875,
       5.88 , 5.572, 6.416, 5.859, 6.546, 6.02 , 6.315, 6.86 , 6.98 ,
       7.765, 6.144, 7.155, 6.563, 5.604, 6.153, 7.831, 6.782, 6.556,
       7.185, 6.951, 6.739, 7.178, 6.8  , 6.604, 7.875, 7.287, 7.107,
       7.274, 6.975, 7.135, 6.162, 7.61 , 7.853, 8.034, 5.891, 6.326,
       5.783, 6.064, 5.344, 5.96 , 5.404, 5.807, 6.375, 5.412, 6.182,
       5.888, 6.642, 5.951, 6.373, 6.951, 6.164, 6.879, 6.618, 8.266,
       8.725, 8.04 , 7.163, 7.686, 6.552, 5.981, 7.412, 8.337, 8.247,
       6.726, 6.086, 6.631, 7.358, 6.481, 6.606, 6.897, 6.095, 6.358,
       6.393, 5.593, 5.605, 6.108, 6.226, 6.433, 6.718, 6.487, 6.438,
       6.957, 8.259, 6.108, 5.876, 7.454, 8.704, 7.333, 6.842, 7.203,
       7.52 , 8.398, 7.327, 7.206, 5.56 , 7.014, 8.297, 7.47 , 5.92 ,
       5.856, 6.24 , 6.538, 7.691, 6.758, 6.854, 7.267, 6.826, 6.482,
       6.812, 7.82 , 6.968, 7.645, 7.923, 7.088, 6.453, 6.23 , 6.209,
       6.315, 6.565, 6.861, 7.148, 6.63 , 6.127, 6.009, 6.678, 6.549,
       5.79 , 6.345, 7.041, 6.871, 6.59 , 6.495, 6.982, 7.236, 6.616,
```

```
       7.42 , 6.849, 6.635, 5.972, 4.973, 6.122, 6.023, 6.266, 6.567,
       5.705, 5.914, 5.782, 6.382, 6.113, 6.426, 6.376, 6.041, 5.708,
       6.415, 6.431, 6.312, 6.083, 5.868, 6.333, 6.144, 5.706, 6.031,
       6.316, 6.31 , 6.037, 5.869, 5.895, 6.059, 5.985, 5.968, 7.241,
       6.54 , 6.696, 6.874, 6.014, 5.898, 6.516, 6.635, 6.939, 6.49 ,
       6.579, 5.884, 6.728, 5.663, 5.936, 6.212, 6.395, 6.127, 6.112,
       6.398, 6.251, 5.362, 5.803, 8.78 , 3.561, 4.963, 3.863, 4.97 ,
       6.683, 7.016, 6.216, 5.875, 4.906, 4.138, 7.313, 6.649, 6.794,
       6.38 , 6.223, 6.968, 6.545, 5.536, 5.52 , 4.368, 5.277, 4.652,
       5.   , 4.88 , 5.39 , 5.713, 6.051, 5.036, 6.193, 5.887, 6.471,
       6.405, 5.747, 5.453, 5.852, 5.987, 6.343, 6.404, 5.349, 5.531,
       5.683, 4.138, 5.608, 5.617, 6.852, 5.757, 6.657, 4.628, 5.155,
       4.519, 6.434, 6.782, 5.304, 5.957, 6.824, 6.411, 6.006, 5.648,
       6.103, 5.565, 5.896, 5.837, 6.202, 6.193, 6.38 , 6.348, 6.833,
       6.425, 6.436, 6.208, 6.629, 6.461, 6.152, 5.935, 5.627, 5.818,
       6.406, 6.219, 6.485, 5.854, 6.459, 6.341, 6.251, 6.185, 6.417,
       6.749, 6.655, 6.297, 7.393, 6.728, 6.525, 5.976, 5.936, 6.301,
       6.081, 6.701, 6.376, 6.317, 6.513, 6.209, 5.759, 5.952, 6.003,
       5.926, 5.713, 6.167, 6.229, 6.437, 6.98 , 5.427, 6.162, 6.484,
       5.304, 6.185, 6.229, 6.242, 6.75 , 7.061, 5.762, 5.871, 6.312,
       6.114, 5.905, 5.454, 5.414, 5.093, 5.983, 5.983, 5.707, 5.926,
       5.67 , 5.39 , 5.794, 6.019, 5.569, 6.027, 6.593, 6.12 , 6.976,
       6.794, 6.03 ])
```

```
np.array(data.loc[:,"rm"]).reshape(-1,1)
```

```
array([[6.575],
       [6.421],
       [7.185],
       [6.998],
       [7.147],
       [6.43 ],
       [6.012],
       [6.172],
       [5.631],
       [6.004],
       [6.377],
       [6.009],
       [5.889],
       [5.949],
       [6.096],
       [5.834],
       [5.935],
       [5.99 ],
       [5.456],
       [5.727],
       [5.57 ],
       [5.965],
       [6.142],
       [5.813],
       [5.924],
       [5.599],
       [5.813],
       [6.047],
       [6.495],
       [6.674],
       [5.713],
       [6.072],
       [5.95 ],
```

```
       [5.701],
       [6.096],
       [5.933],
       [5.841],
       [5.85 ],
       [5.966],
       [6.595],
       [7.024],
       [6.77 ],
       [6.169],
       [6.211],
       [6.069],
       [5.682],
       [5.786],
       [6.03 ],
       [5.399],
       [5.602],
       [5.963],
       [6.115],
       [6.511],
       [5.998],
       [5.888],
       [7.249],
       [6.383],
       [6.816],
```

Zadanie 2

```
data.describe()
```

	Unnamed: 0	crim	zn	indus	chas	nox	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.0000
mean	253.500000	3.613524	11.363636	11.136779	0.069170	0.554695	6.2846
std	146.213884	8.601545	23.322453	6.860353	0.253994	0.115878	0.7026
min	1.000000	0.006320	0.000000	0.460000	0.000000	0.385000	3.5610
25%	127.250000	0.082045	0.000000	5.190000	0.000000	0.449000	5.8855
50%	253.500000	0.256510	0.000000	9.690000	0.000000	0.538000	6.2085
75%	379.750000	3.677083	12.500000	18.100000	0.000000	0.624000	6.6235
max	506.000000	88.976200	100.000000	27.740000	1.000000	0.871000	8.7800

```
for (columnName, columnData) in data.iteritems():
    print('Column Name : ', columnName)
    #print('Column Contents : ', columnData.values)
    print("Min")
    print(columnData.values.min())
```

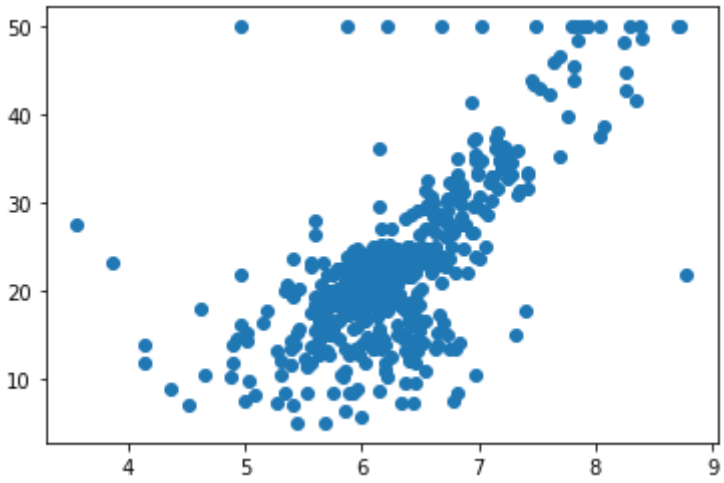
```
print("Max")
print(columnData.values.max())
print("Std")
print(columnData.values.std())
print("Mean")
print(columnData.values.mean())
```

```
12.1265
Std
2.1036283563444593
Mean
3.795042687747036
Column Name :  rad
Min
1
Max
24
Std
8.698651117790636
Mean
9.549407114624506
Column Name :  tax
Min
187
Max
711
Std
168.37049503938118
Mean
408.2371541501976
Column Name :  ptratio
Min
12.6
Max
22.0
Std
2.1628051914821365
Mean
18.455533596837945
Column Name :  black
Min
0.32
Max
396.9
Std
91.20460745217277
Mean
356.6740316205534
Column Name :  lstat
Min
1.73
Max
37.97
Std
7.134001636650485
Mean
12.653063241106722
Column Name :  medv
```

```
Min
5.0
Max
50.0
Std
9.188011545278203
Mean
22.532806324110677
```

Zadanie 3

```
import matplotlib.pyplot as plt


rm=pd.DataFrame(data,columns=['rm'])
medv=pd.DataFrame(data,columns=['medv'])


plt.scatter(rm,medv)
plt.show()
```




```
number_of_points=506
x_point = []
y_point = []
x_1=[]
y_1=[]


for i in range(number_of_points):
  x_1.append(1)
  y_1.append(1)


x_s=np.array(rm)
y_s=np.array(medv)


x2=np.sum(x_s*x_s)
x1=np.sum(x_s)
x1sum=np.sum(x_1)
y1sum=np.sum(y_1)
```

```
xy=np.sum(x_s*y_s)
y1=np.sum(y_s)
```

```
print(x2,x1,x1sum,y1sum,xy,y1)
```

```
20234.598247 3180.025 506 506 73924.0776 11401.600000000002
```

```
M = np.array([[x2,x1],[x1,x1sum]])
```

```
M_1=np.linalg.inv(M)
```

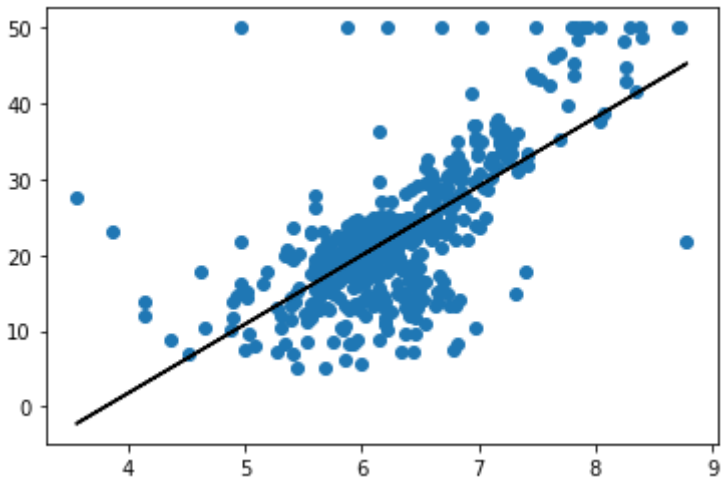
```
N=np.array([[xy],[y1]])
print(N)
```

```
[[73924.0776]
 [11401.6   ]]
```

```
a,b = np.matmul(M_1,N)
print(a,b)
```

```
[9.10210898] [-34.67062078]
```

```
plt.scatter(x_s,y_s)
plt.plot(x_s,a*x_s+b,"black")
plt.show()
```



Zadanie 4

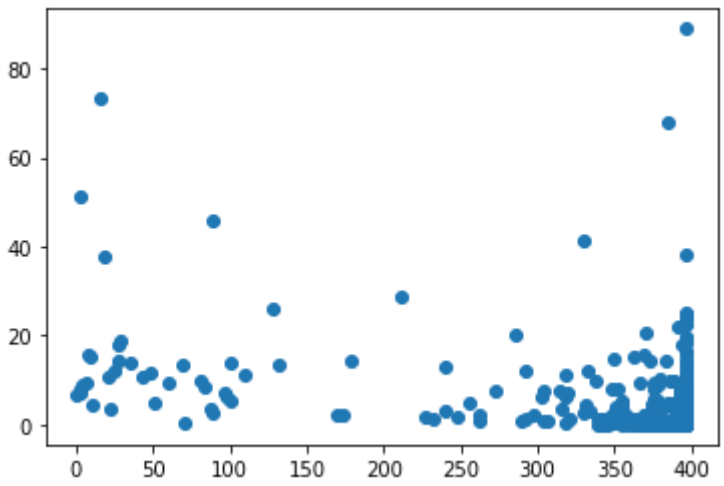
Kowiariacja

```
cov = data.iloc[:,1:].cov()
cov.style.background_gradient(cmap='coolwarm')
```

	crim	zn	indus	chas	nox	rm	
crim	73.986578	-40.215956	23.992339	-0.122109	0.419594	-1.325038	85.40
zn	-40.215956	543.936814	-85.412648	-0.252925	-1.396148	5.112513	-373.90
indus	23.992339	-85.412648	47.064442	0.109669	0.607074	-1.887957	124.51
chas	-0.122109	-0.252925	0.109669	0.064513	0.002684	0.016285	0.61
nox	0.419594	-1.396148	0.607074	0.002684	0.013428	-0.024603	2.38
rm	-1.325038	5.112513	-1.887957	0.016285	-0.024603	0.493671	-4.75
age	85.405322	-373.901548	124.513903	0.618571	2.385927	-4.751929	792.35
dis	-6.876722	32.629304	-10.228097	-0.053043	-0.187696	0.303663	-44.32
rad	46.847761	-63.348695	35.549971	-0.016296	0.616929	-1.283815	111.77
tax	844.821538	-1236.453735	833.360290	-1.523367	13.046286	-34.583448	2402.69
ptratio	5.399331	-19.776571	5.692104	-0.066819	0.047397	-0.540763	15.93
black	-302.381816	373.721402	-223.579756	1.131325	-4.020570	8.215006	-702.94
lstat	27.986168	-68.783037	29.580270	-0.097816	0.488946	-3.079741	121.07
medv	-30.718508	77.315176	-30.520823	0.409409	-0.455412	4.493446	-97.58

```
black=pd.DataFrame(data,columns=['black'])
crim=pd.DataFrame(data,columns=['crim'])
```

```
plt.scatter(black,crim)
plt.show()
```



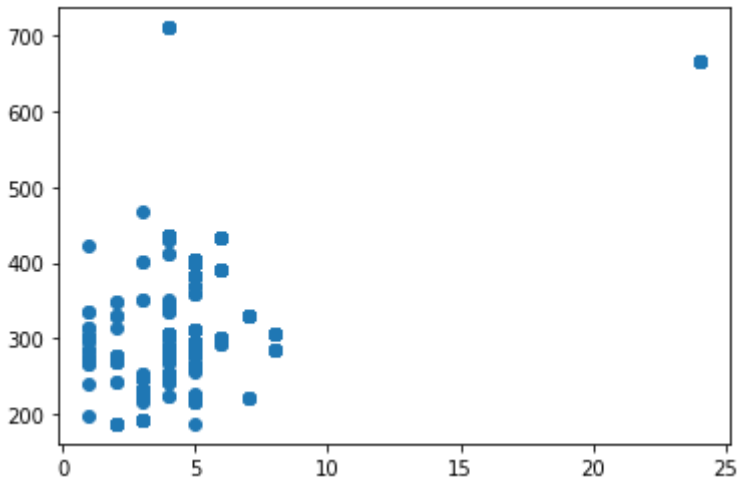
Korelacja

```
cor = data.iloc[:,1:].corr()
cor.style.background_gradient(cmap='coolwarm')
```

	crim	zn	indus	chas	nox	rm	age	
crim	1.000000	-0.200469	0.406583	-0.055892	0.420972	-0.219247	0.352734	-0.37
zn	-0.200469	1.000000	-0.533828	-0.042697	-0.516604	0.311991	-0.569537	0.66
indus	0.406583	-0.533828	1.000000	0.062938	0.763651	-0.391676	0.644779	-0.70
chas	-0.055892	-0.042697	0.062938	1.000000	0.091203	0.091251	0.086518	-0.09
nox	0.420972	-0.516604	0.763651	0.091203	1.000000	-0.302188	0.731470	-0.76
rm	-0.219247	0.311991	-0.391676	0.091251	-0.302188	1.000000	-0.240265	0.20
age	0.352734	-0.569537	0.644779	0.086518	0.731470	-0.240265	1.000000	-0.74
dis	-0.379670	0.664408	-0.708027	-0.099176	-0.769230	0.205246	-0.747881	1.00
rad	0.625505	-0.311948	0.595129	-0.007368	0.611441	-0.209847	0.456022	-0.49
tax	0.582764	-0.314563	0.720760	-0.035587	0.668023	-0.292048	0.506456	-0.53
ptratio	0.289946	-0.391679	0.383248	-0.121515	0.188933	-0.355501	0.261515	-0.23
black	-0.385064	0.175520	-0.356977	0.048788	-0.380051	0.128069	-0.273534	0.29
lstat	0.455621	-0.412995	0.603800	-0.053929	0.590879	-0.613808	0.602339	-0.49
medv	-0.388305	0.360445	-0.483725	0.175260	-0.427321	0.695360	-0.376955	0.24

```
rad=pd.DataFrame(data,columns=['rad'])
tax=pd.DataFrame(data,columns=['tax'])
nox=pd.DataFrame(data,columns=['nox'])
indus=pd.DataFrame(data,columns=['indus'])
```

```
plt.scatter(rad,tax)
plt.show()
```



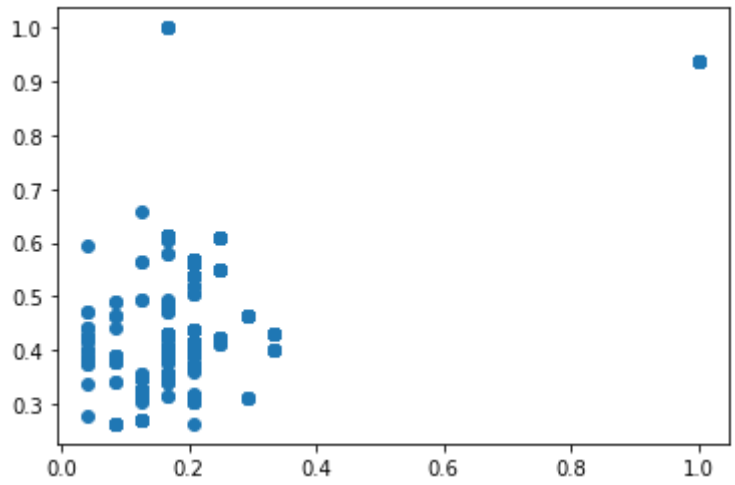

```
for column in rad:
  rad[column]=rad[column] / rad[column].abs().max()
```

```
for column in tax:
```

```
  tax[column]=tax[column] / tax[column].abs().max()

print(rad)
```

```
            rad
0      0.041667
1      0.083333
2      0.083333
3      0.125000
4      0.125000
..          ...
501    0.041667
502    0.041667
503    0.041667
504    0.041667
505    0.041667

[506 rows x 1 columns]
```

```
plt.scatter(rad,tax)
plt.show()
```



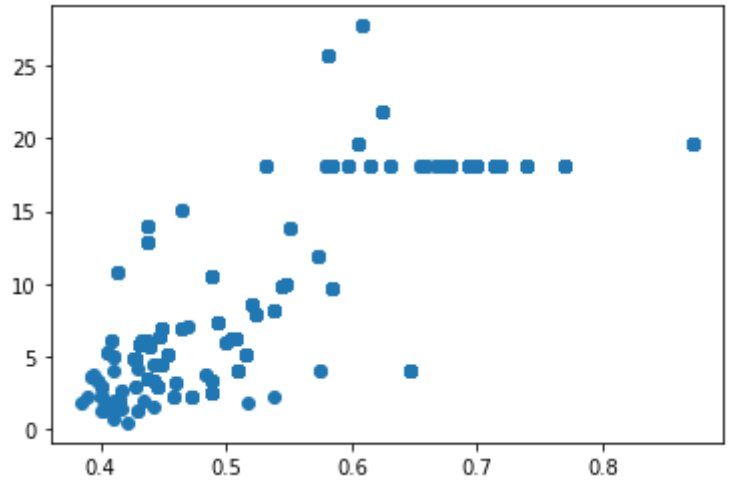

```
print(nox)
```

```
          nox
0      0.538
1      0.469
2      0.469
3      0.458
4      0.458
..       ...
501    0.573
502    0.573
503    0.573
504    0.573
505    0.573

[506 rows x 1 columns]
```

```
print(indus)
```

```
      indus
0      2.31
1      7.07
2      7.07
3      2.18
4      2.18
..      ...
501   11.93
502   11.93
503   11.93
504   11.93
505   11.93

[506 rows x 1 columns]
```

```
plt.scatter(nox,indus)
plt.show()
```




```
number_of_points=506
x_point = []
y_point = []
x_1=[]
y_1=[]


for i in range(number_of_points):
  x_1.append(1)
  y_1.append(1)


print(rad)
```

```
      rad
0       1
1       2
2       2
3       3
4       3
..    ...
501     1
502     1
503     1
```

```
504    1
505    1

[506 rows x 1 columns]
```

```
print(tax)
```

```
     tax
0    296
1    242
2    242
3    222
4    222
..   ...
501  273
502  273
503  273
504  273
505  273

[506 rows x 1 columns]
```

```
x_s=np.array(nox)
y_s=np.array(indus)
```

```
x2=np.sum(x_s*x_s)
x1=np.sum(x_s)
x1sum=np.sum(x_1)
y1sum=np.sum(y_1)
xy=np.sum(x_s*y_s)
y1=np.sum(y_s)
```

```
print(x2,x1,x1sum,y1sum,xy,y1)
```

```
162.47038009 280.6757 506 506 3432.39536 5635.209999999999
```

```
M = np.array([[x2,x1],[x1,x1sum]])
```

```
M_1=np.linalg.inv(M)
```

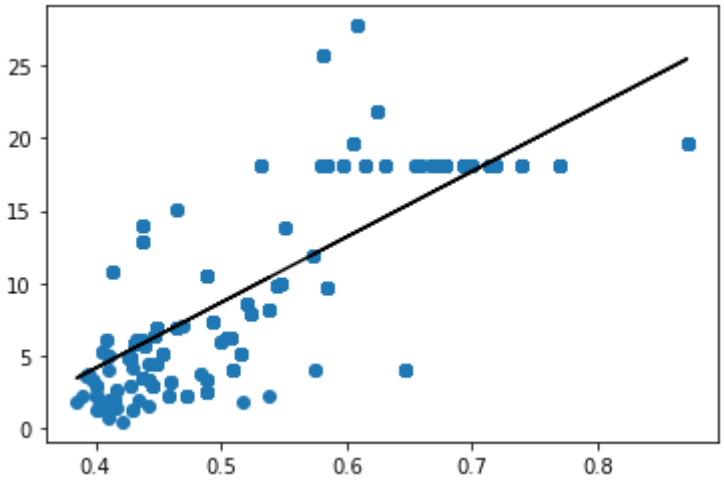
```
N=np.array([[xy],[y1]])
print(N)
```

```
[[3432.39536]
 [5635.21   ]]
```

```
a,b = np.matmul(M_1,N)
print(a,b)
```

```
[45.21076575] [-13.94140973]
```

```
plt.scatter(x_s,y_s)
plt.plot(x_s,a*x_s+b,"black")
plt.show()
```



✓ 0 s ukończono o 11:37