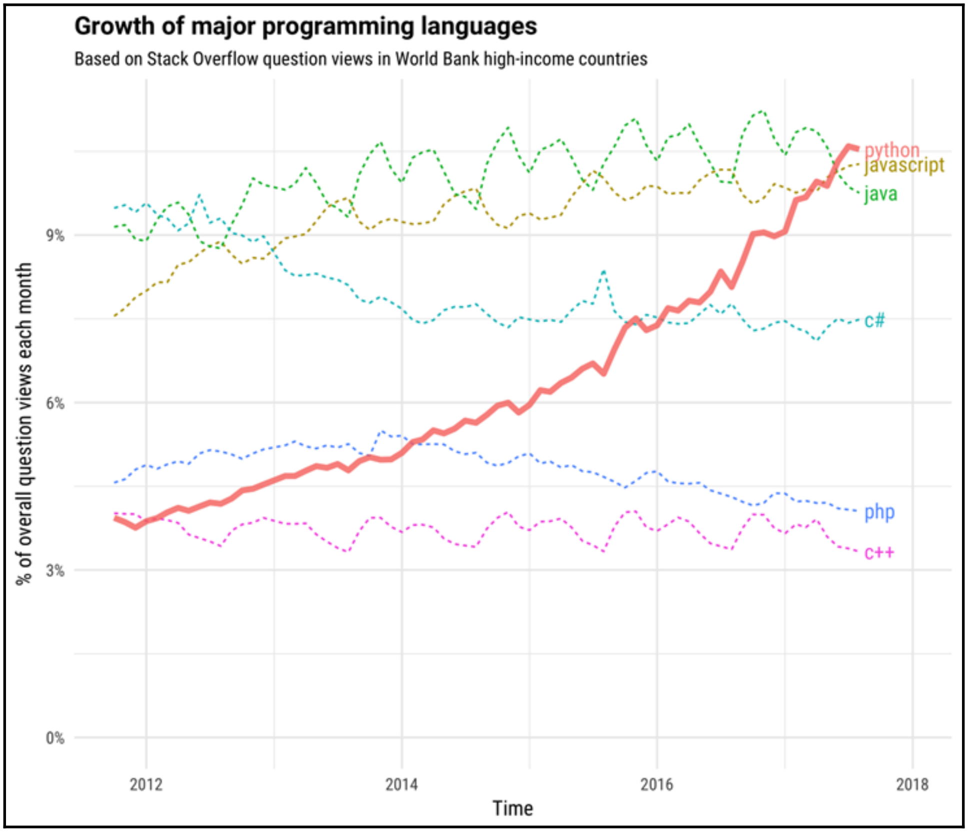
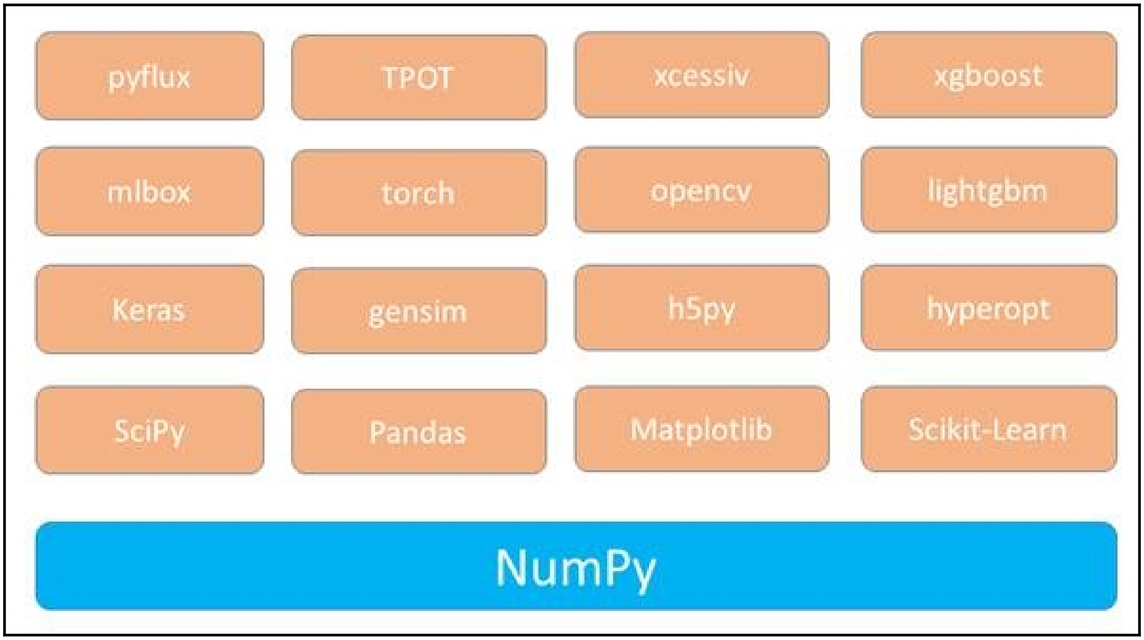
Table of Contents

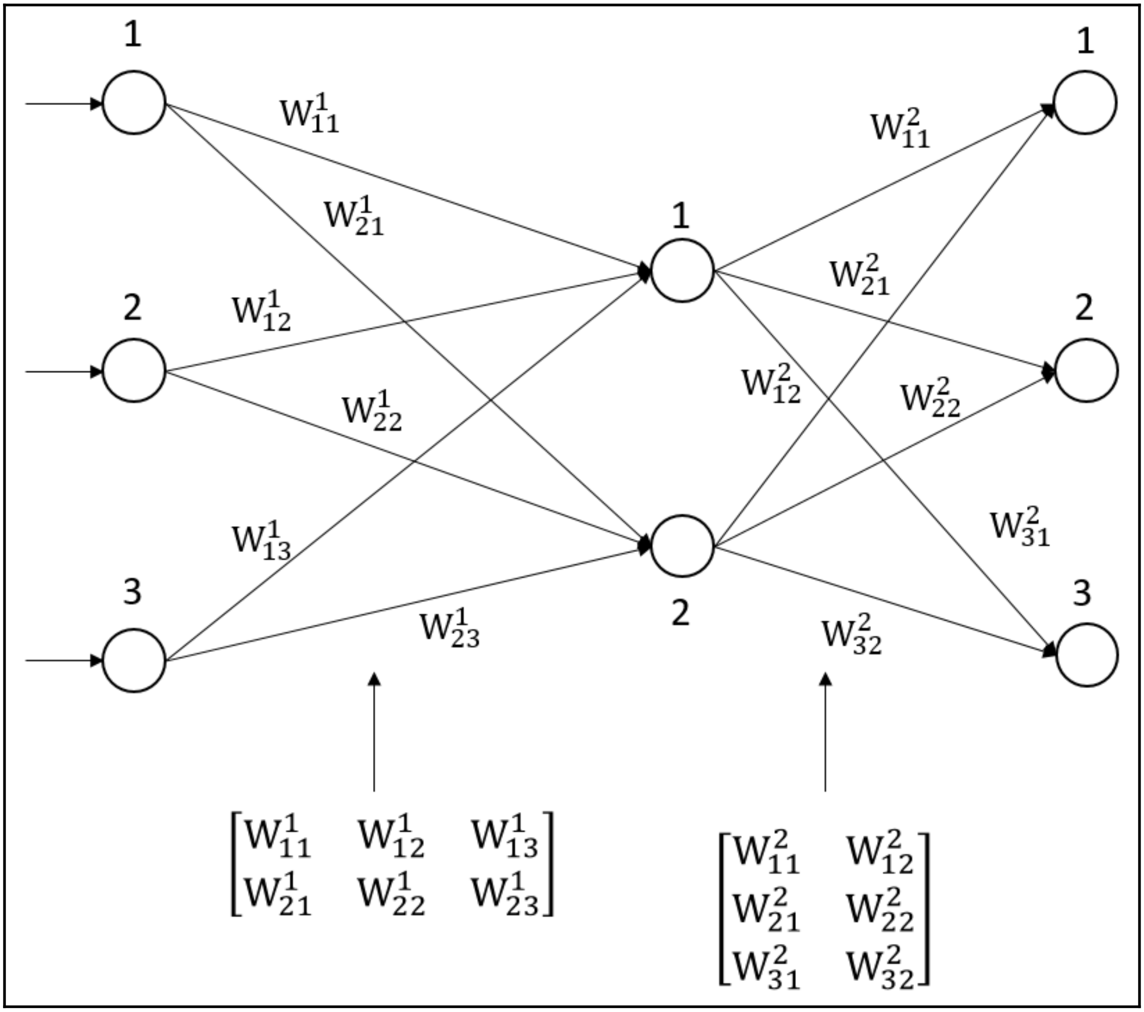
1
Graphic Bundle

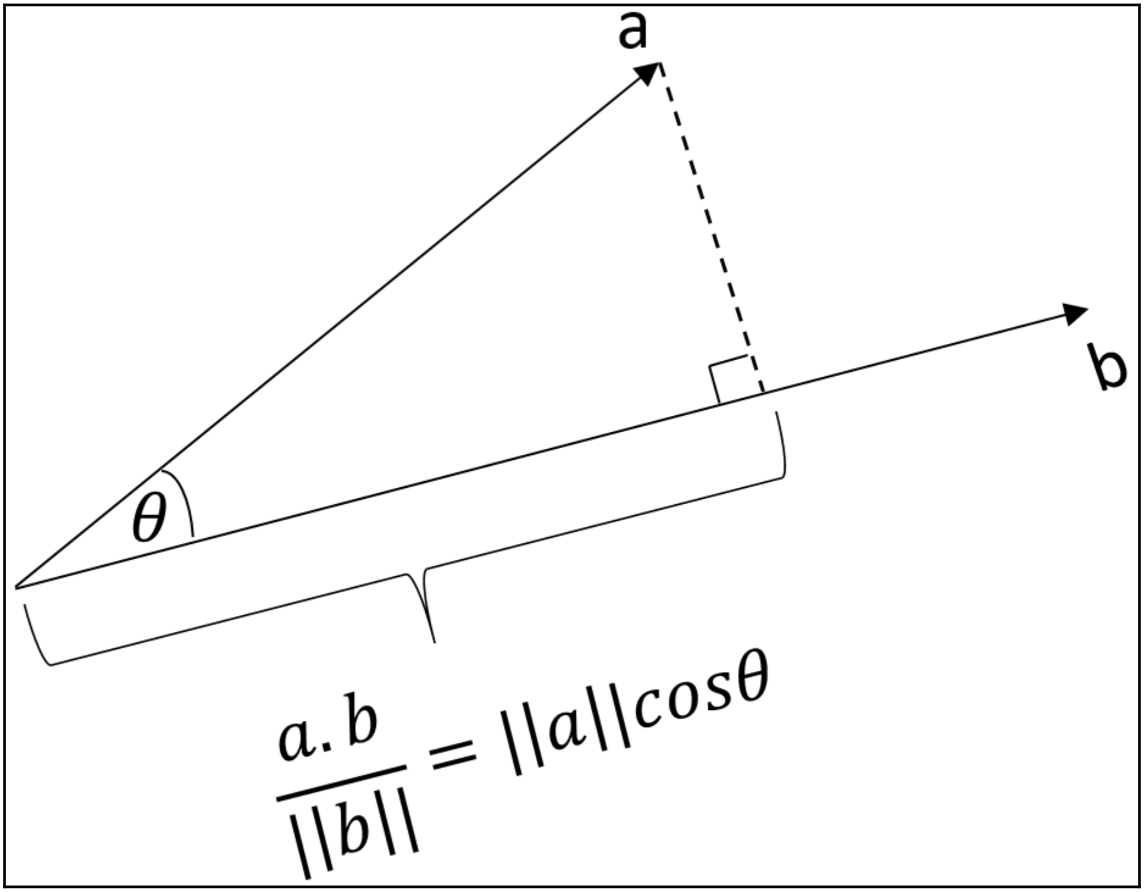
Chapter 1: Working with NumPy Arrays

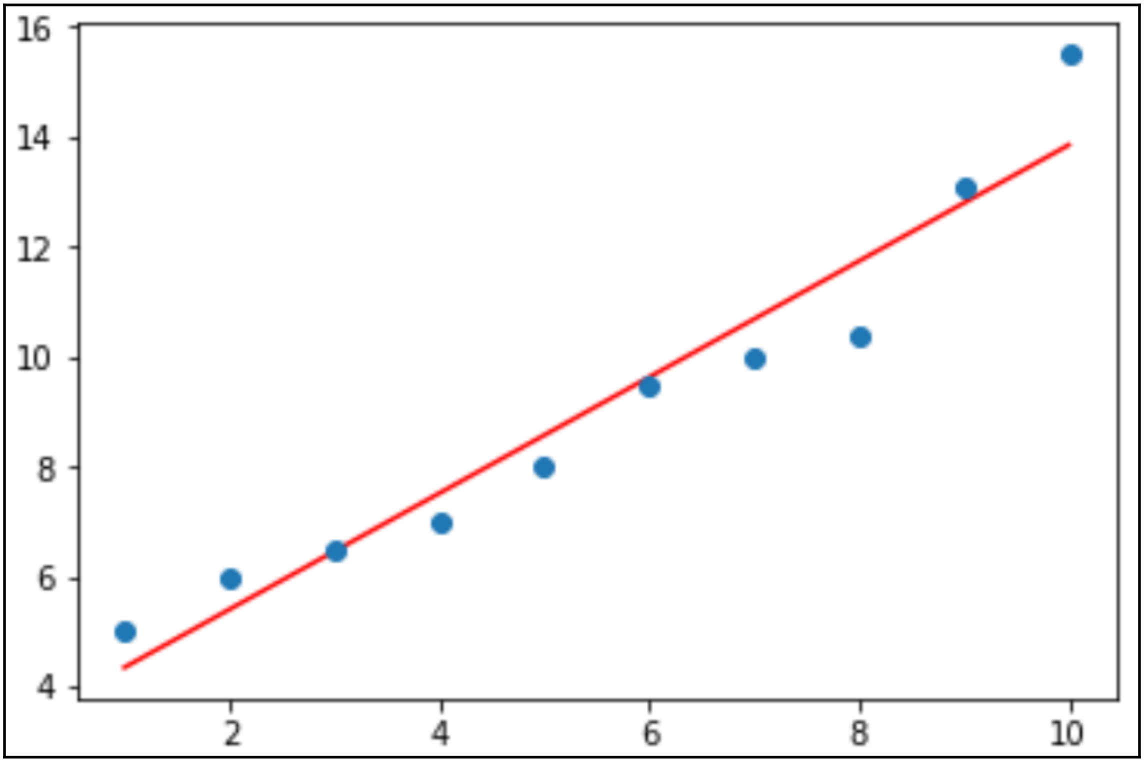



pyflux
TPOT
xcessiv
xgboost
mlbox
torch
opencv
lightgbm
Keras
gensim
h5py
hyperopt
SciPy
Pandas
Matplotlib
Scikit-Learn
NumPy

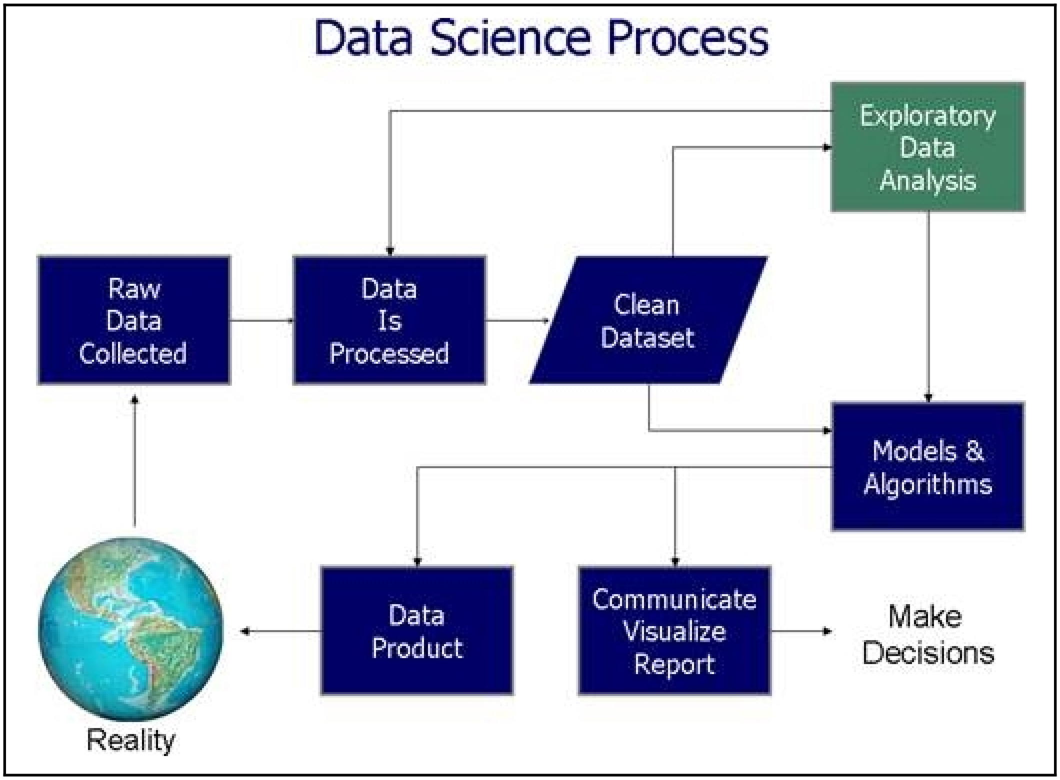
Chapter 2: Linear Algebra with NumPy

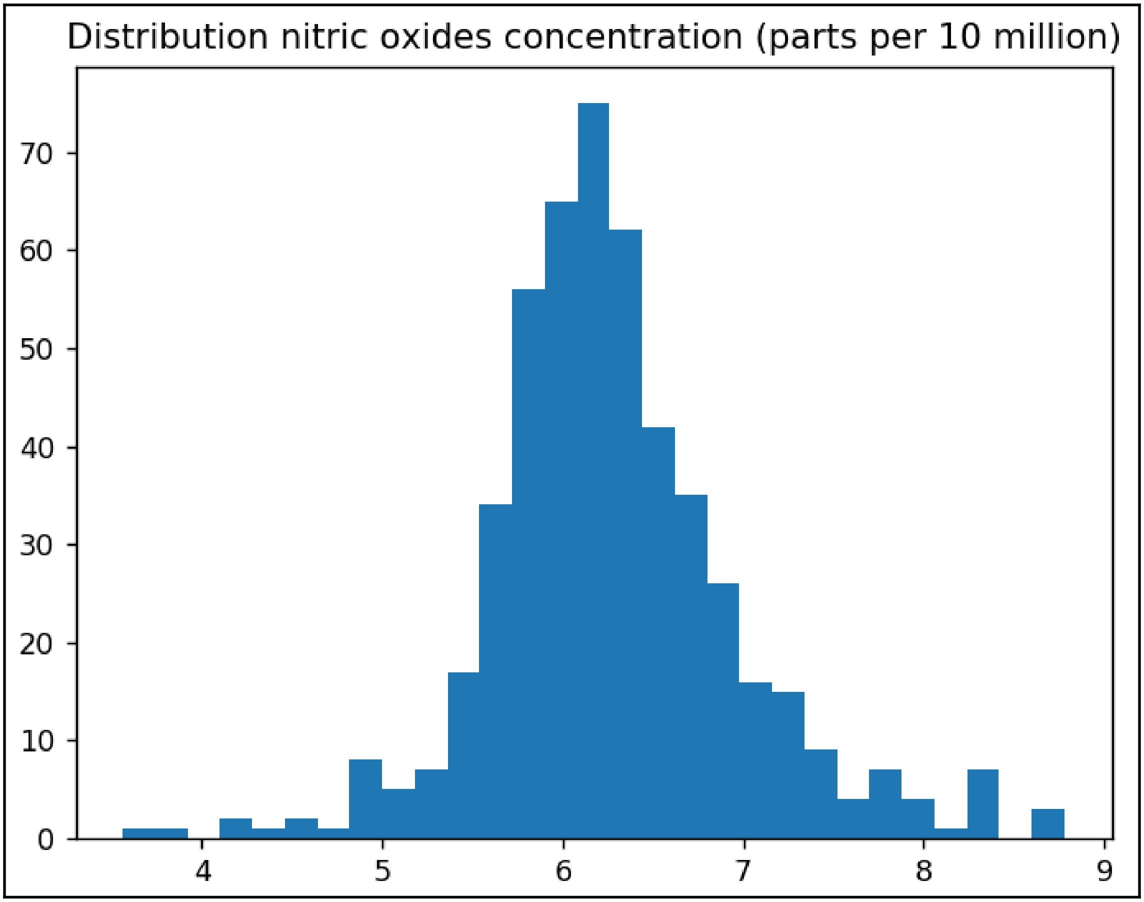


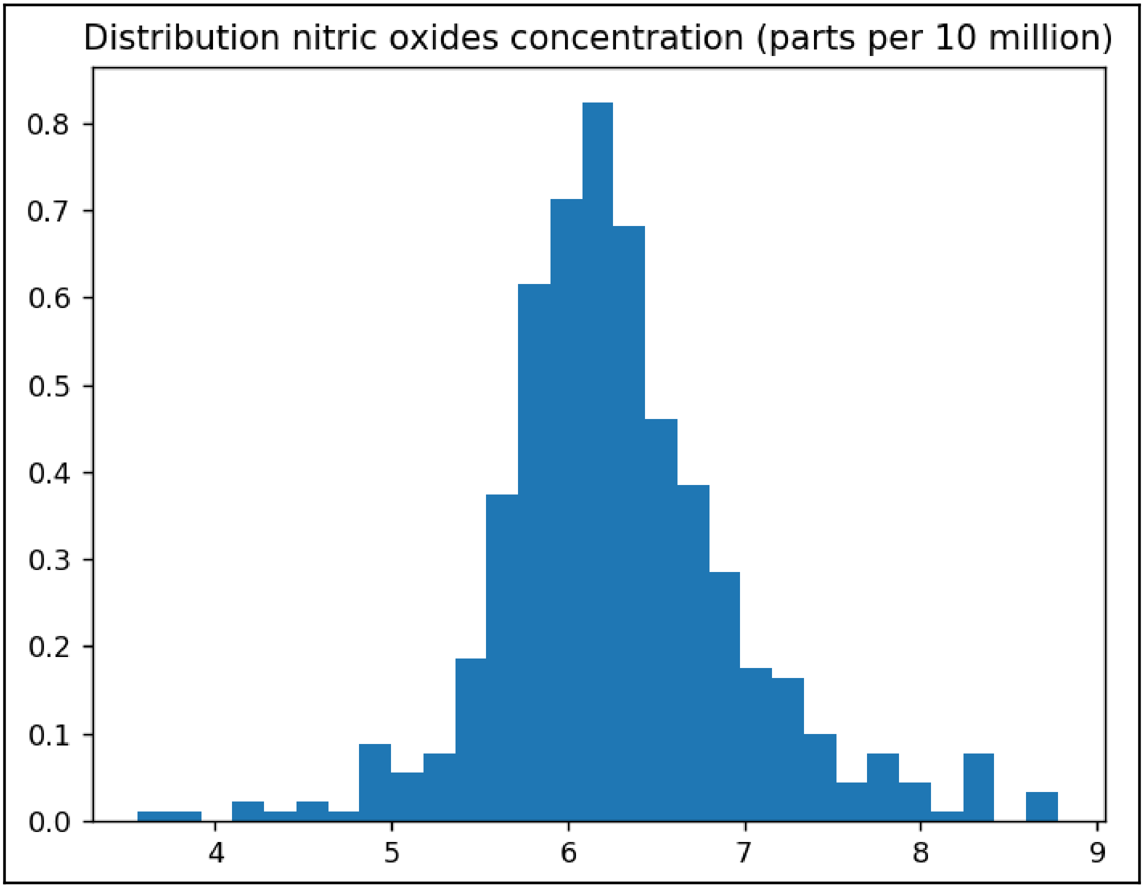

a
b
θ
a.b / ||b|| = ||a||cosθ


16
14
12
10
8
6
4
2
4
6
8
10

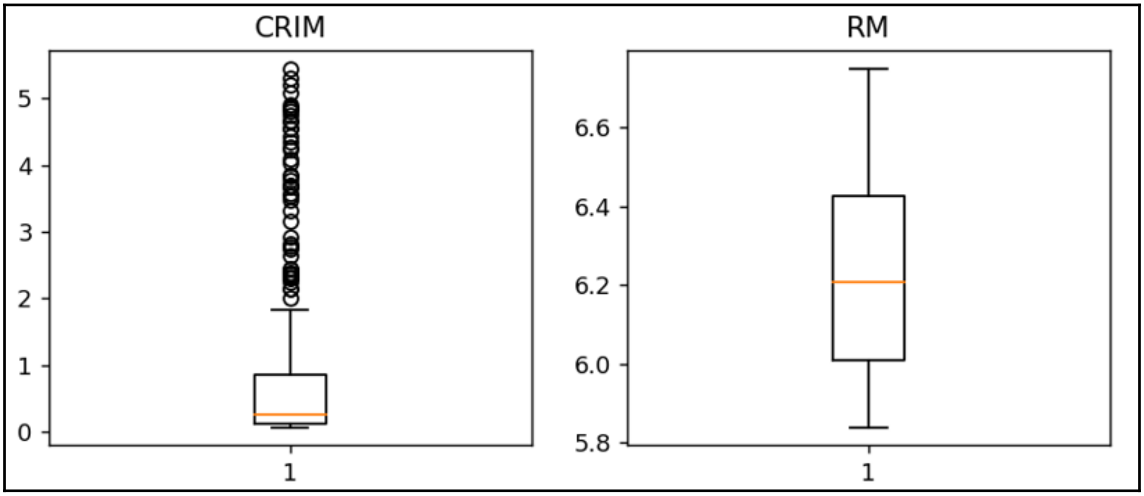
Chapter 3: Explanatory Data Analysis of US Housing Data with NumPy Statistics

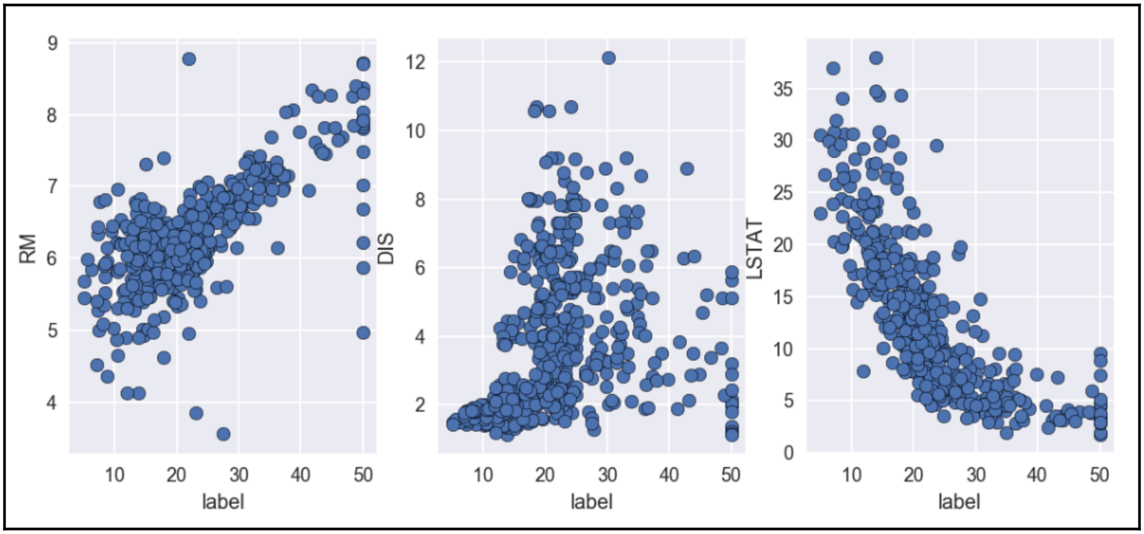



Distribution nitric oxides concentration (parts per 10 million)
70
60
50
40
30
20
10
0
4
5
6
7
8
9




```
Bin Sizes
[ 1.  1.  0.  2.  1.  2.  1.  8.  5.  7. 17. 34. 56. 65. 75. 62. 42. 35. 26. 16. 15.  9.  4.  7.  4.  1.  7.  0.  3.]
Bin Edges
[3.561      3.74096552 3.92093103 4.10089655 4.28086207 4.46082759 4.6407931  4.82075862 5.00072414 5.18068966 5.36065517
 5.54062069 5.72058621 5.90055172 6.08051724 6.26048276 6.44044828 6.62041379 6.80037931 6.98034483 7.16031034 7.34027586
 7.52024138 7.7002069  7.88017241 8.06013793 8.24010345 8.42006897 8.60003448 8.78      ]
```


CRIM
RM
5
4
3
2
1
0
1
6.6
6.4
6.2
6.0
5.8
1


RM
9
8
7
6
5
4
10
20
30
40
50
label
DIS
12
10
8
6
4
2
10
20
30
40
50
label
LSTAT
35
30
25
20
15
10
5
0
10
20
30
40
50
label

```
In [14]: a = np.loadtxt("My_file.txt", delimiter='\t')

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-14-bde8ee3f2c6d> in <module>()
----> 1 a = np.loadtxt("My_file.txt", delimiter='\t')

c:\users\mert_cuhadaroglu\appdata\local\programs\python\python36\lib\site-packages\numpy\lib\npyio.py in loadtxt(fname, dtype,
comments, delimiter, converters, skiprows, usecols, unpack, ndmin, encoding)
   1090         # converting the data
   1091         X = None
-> 1092         for x in read_data(_loadtxt_chunksize):
   1093             if X is None:
   1094                 X = np.array(x, dtype)

c:\users\mert_cuhadaroglu\appdata\local\programs\python\python36\lib\site-packages\numpy\lib\npyio.py in read_data(chunk_size)
   1017 
   1018             # Convert each value according to its column and store
-> 1019             items = [conv(val) for (conv, val) in zip(converters, vals)]
   1020 
   1021             # Then pack it according to the dtype's nesting

c:\users\mert_cuhadaroglu\appdata\local\programs\python\python36\lib\site-packages\numpy\lib\npyio.py in <listcomp>(.0)
   1017 
   1018             # Convert each value according to its column and store
-> 1019             items = [conv(val) for (conv, val) in zip(converters, vals)]
   1020 
   1021             # Then pack it according to the dtype's nesting

c:\users\mert_cuhadaroglu\appdata\local\programs\python\python36\lib\site-packages\numpy\lib\npyio.py in floatconv(x)
    736         if '0x' in x:
    737             return float.fromhex(x)
--> 738         return float(x)
    739 
    740     typ = dtype.type

ValueError: could not convert string to float: 'The following numbers are generated for the purpose of this chapter'
```

```
Out[29]: {'data': array([[6.3200e-03, 1.8000e+01, 2.3100e+00, ..., 1.5300e+01, 3.9690e+02,
          4.9800e+00],
         [2.7310e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9690e+02,
          9.1400e+00],
         [2.7290e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9283e+02,
          4.0300e+00],
         ...,
         [6.0760e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+02,
          5.6400e+00],
         [1.0959e-01, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9345e+02,
          6.4800e+00],
         [4.7410e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+02,
          7.8800e+00]]),
 'target': array([24. , 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15. ,
        18.9, 21.7, 20.4, 18.2, 19.9, 23.1, 17.5, 20.2, 18.2, 13.6, 19.6,
        15.2, 14.5, 15.6, 13.9, 16.6, 14.8, 18.4, 21. , 12.7, 14.5, 13.2,
        13.1, 13.5, 18.9, 20. , 21. , 24.7, 30.8, 34.9, 26.6, 25.3, 24.7,
        21.2, 19.3, 20. , 16.6, 14.4, 19.4, 19.7, 20.5, 25. , 23.4, 18.9,
        35.4, 24.7, 31.6, 23.3, 19.6, 18.7, 16. , 22.2, 25. , 33. , 23.5,
        19.4, 22. , 17.4, 20.9, 24.2, 21.7, 22.8, 23.4, 24.1, 21.4, 20. ,
        20.8, 21.2, 20.3, 28. , 23.9, 24.8, 22.9, 23.9, 26.6, 22.5, 22.2,
        23.6, 28.7, 22.6, 22. , 22.9, 25. , 20.6, 28.4, 21.4, 38.7, 43.8,
        33.2, 27.5, 26.5, 18.6, 19.3, 20.1, 19.5, 19.5, 20.4, 19.8, 19.4,
        21.7, 22.8, 18.8, 18.7, 18.5, 18.3, 21.2, 19.2, 20.4, 19.3, 22. ,
        20.3, 20.5, 17.3, 18.8, 21.4, 15.7, 16.2, 18. , 14.3, 19.2, 19.6,
        23. , 18.4, 15.6, 18.1, 17.4, 17.1, 13.3, 17.8, 14. , 14.4, 13.4,
        15.6, 11.8, 13.8, 15.6, 14.6, 17.8, 15.4, 21.5, 19.6, 15.3, 19.4,
        17. , 15.6, 13.1, 41.3, 24.3, 23.3, 27. , 50. , 50. , 50. , 22.7,
        25. , 50. , 23.8, 23.8, 22.3, 17.4, 19.1, 23.1, 23.6, 22.6, 29.4,
        23.2, 24.6, 29.9, 37.2, 39.8, 36.2, 37.9, 32.5, 26.4, 29.6, 50. ,
        32. , 29.8, 34.9, 37. , 30.5, 36.4, 31.1, 29.1, 50. , 33.3, 30.3,
        34.6, 34.9, 32.9, 24.1, 42.3, 48.5, 50. , 22.6, 24.4, 22.5, 24.4,
        20. , 21.7, 19.3, 22.4, 28.1, 23.7, 25. , 23.3, 28.7, 21.5, 23. ,
        26.7, 21.7, 27.5, 30.1, 44.8, 50. , 37.6, 31.6, 46.7, 31.5, 24.3,
        31.7, 41.7, 48.3, 29. , 24. , 25.1, 31.5, 23.7, 23.3, 22. , 20.1,
        22.2, 23.7, 17.6, 18.5, 24.3, 20.5, 24.5, 26.2, 24.4, 24.8, 29.6,
        42.8, 21.9, 20.9, 44. , 50. , 36. , 30.1, 33.8, 43.1, 48.8, 31. ,
        36.5, 22.8, 30.7, 50. , 43.5, 20.7, 21.1, 25.2, 24.4, 35.2, 32.4,
        32. , 33.2, 33.1, 29.1, 35.1, 45.4, 35.4, 46. , 50. , 32.2, 22. ,
```

```
Data Set Characteristics:

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive

    :Median Value (attribute 14) is usually the target

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 25,000 sq.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 1940
        - DIS      weighted distances to five Boston employment centres
        - RAD      index of accessibility to radial highways
        - TAX      full-value property-tax rate per $10,000
        - PTRATIO  pupil-teacher ratio by town
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's
```

```
In [41]: Basic_Statistics = np.vstack((minimums,maximums,range_column,mean,median, variance, tenth_percentile,ninety_percentile))
         Basic_Statistics

Out[41]: array([[   0. ,     0. ,     0.5,     0. ,     0.4,     3.6,     2.9,     1.1,     1. ,   187. ,    12.6,     0.3,
            1.7],
          [  89. ,   100. ,    27.7,     1. ,     0.9,     8.8,   100. ,    12.1,    24. ,   711. ,    22. ,   396.9,
            38. ],
          [  89. ,   100. ,    27.3,     1. ,     0.5,     5.2,    97.1,    11. ,    23. ,   524. ,     9.4,   396.6,
            36.2],
          [   3.6,    11.4,    11.1,     0.1,     0.6,     6.3,    68.6,     3.8,     9.5,   408.2,    18.5,   356.7,
            12.7],
          [   0.3,     0. ,     9.7,     0. ,     0.5,     6.2,    77.5,     3.2,     5. ,   330. ,    19. ,   391.4,
            11.4],
          [  73.8,   542.9,    47. ,     0.1,     0. ,     0.5,   790.8,     4.4,    75.7, 28348.6,    4.7,  8318.3,
            50.9],
          [   0. ,     0. ,     2.9,     0. ,     0.4,     5.6,    27. ,     1.6,     3. ,   233. ,    14.8,   290.3,
            4.7],
          [  10.5,    42.5,    19.6,     0. ,     0.7,     7.2,    98.8,     6.8,    24. ,   666. ,    20.9,   396.9,
            23. ]])
```

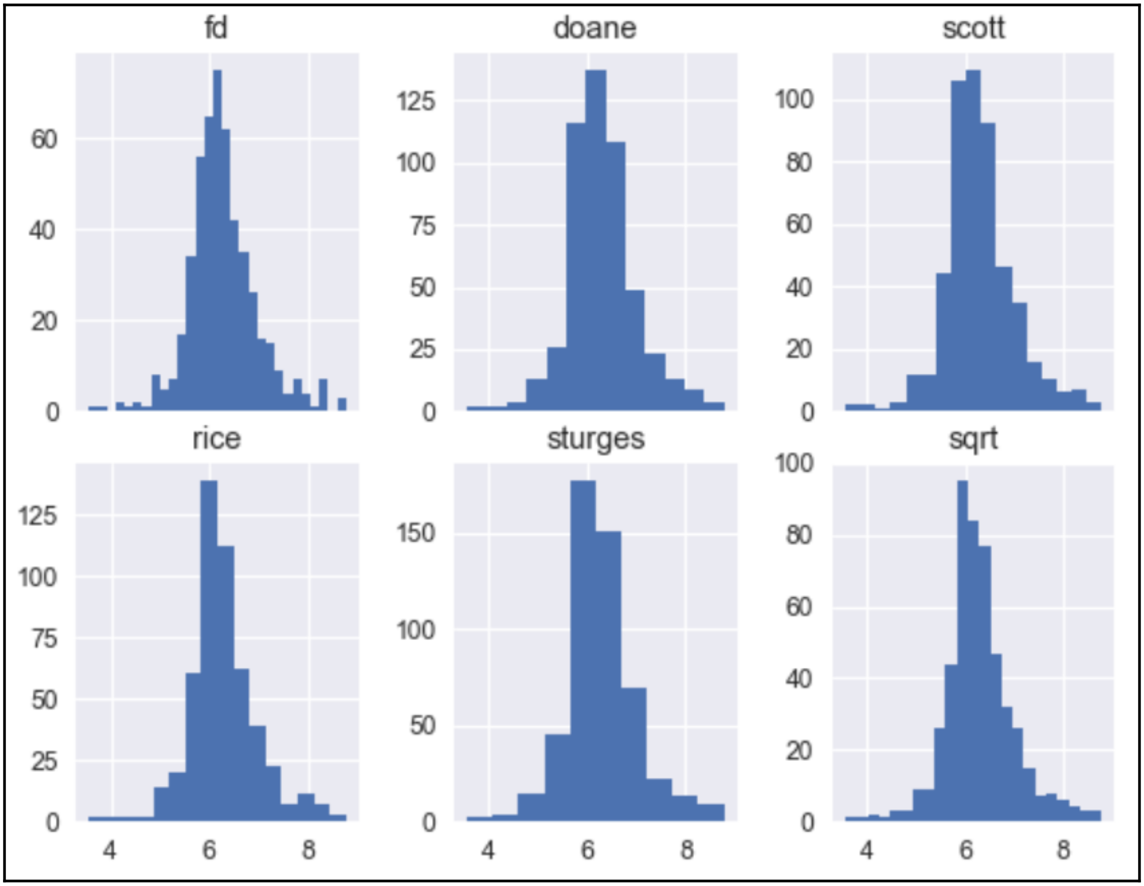
```
In [42]: stat_labels = ['minm', 'maxm', 'rang', 'mean','medi', 'vari','50%t','90%t']
```

```
In [43]: print("          F1      F2      F3      F4      F5      F6      F7      F8      F9      F10     F11     F12    F13   ")
         for stat_labels , row in zip(stat_labels,Basic_Statistics):
             print('%s [%s]' % (stat_labels, ''.join('%07s' % i for i in row)))

              F1      F2      F3      F4      F5      F6      F7      F8      F9      F10     F11     F12    F13
         minm [    0.0     0.0     0.5     0.0     0.4     3.6     2.9     1.1     1.0   187.0    12.6     0.3    1.7]
         maxm [   89.0   100.0    27.7     1.0     0.9     8.8   100.0    12.1    24.0   711.0    22.0   396.9   38.0]
         rang [   89.0   100.0    27.3     1.0     0.5     5.2    97.1    11.0    23.0   524.0     9.4   396.6   36.2]
         mean [    3.6    11.4    11.1     0.1     0.6     6.3    68.6     3.8     9.5   408.2    18.5   356.7   12.7]
         medi [    0.3     0.0     9.7     0.0     0.5     6.2    77.5     3.2     5.0   330.0    19.0   391.4   11.4]
         vari [   73.8   542.9    47.0     0.1     0.0     0.5   790.8     4.4   75.728348.6     4.7  8318.3   50.9]
         50%t [    0.0     0.0     2.9     0.0     0.4     5.6    27.0     1.6     3.0   233.0    14.8   290.3    4.7]
         90%t [   10.5    42.5    19.6     0.0     0.7     7.2    98.8     6.8    24.0   666.0    20.9   396.9   23.0]
```

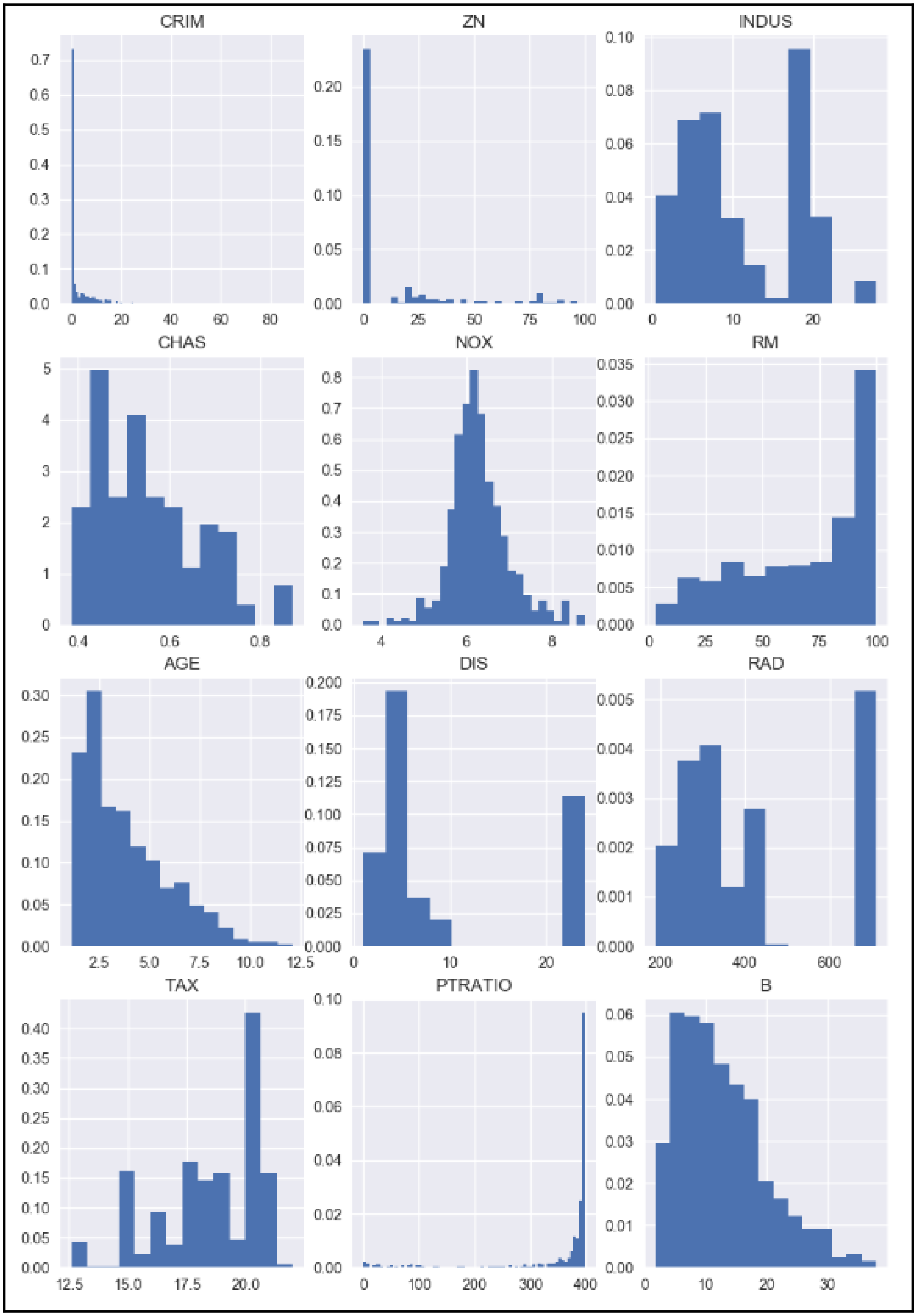
```
In [44]: from scipy import stats
         arr= stats.describe(samples, axis=0)
         arr

Out[44]: DescribeResult(nobs=506, minmax=(array([  0.00632,   0.     ,   0.46   ,   0.     ,   0.385  ,   3.561  ,   2.9    ,   1.1296 ,
         1.     , 187.     ,
                12.6    ,   0.32   ,   1.73   ]), array([ 88.9762, 100.    ,  27.74  ,   1.    ,   0.871 ,   8.78  , 100.    ,  12.126
         5,  24.    , 711.    ,  22.    ,
               396.9   ,  37.97   ])), mean=array([  3.59376071,  11.36363636,  11.13677866,   0.06916996,   0.55469506,   6.28463439,
         68.57490119,   3.79504269,
                 9.54940711, 408.23715415,  18.4555336 , 356.67403162,  12.65306324]), variance=array([   73.90467096,   543.93681368,
         47.06444247,     0.06451297,     0.01342764,     0.49367085,   792.35839851,
                  4.43401514,    75.81636598, 28404.75948812,     4.68698912,  8334.75226292,    50.99475951]), skewness=array([ 5.222
         03907,  2.21906306,  0.29414628,  3.39579929,  0.72714416,  0.40241467, -0.59718559,  1.00877876,  1.00183349,
                0.66796827, -0.79994453, -2.88179835,  0.90377074]), kurtosis=array([36.88811011,  3.97994877, -1.23321847,  9.5314528
         4, -0.07586422,  1.86102697, -0.97001393,  0.47129857, -0.8705205 ,
               -1.14298488, -0.29411638,  7.14376929,  0.47654476]))
```

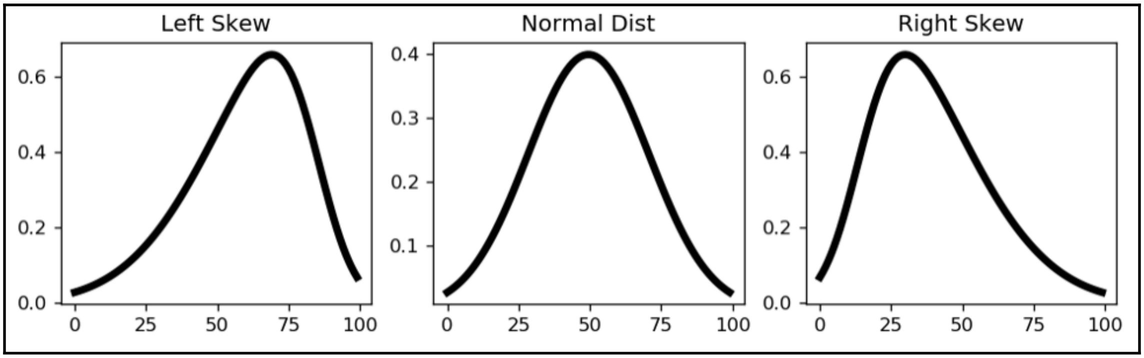
```
In [47]: np.set_printoptions(suppress= True, linewidth= 125)
         print("          F1      F2      F3      F4      F5      F6      F7      F8      F9      F10     F11     F12     F13")
         for stat_labels1, row1 in zip(stat_labels1, Basic_Statistics1):
             print ('%s [%s]' % (stat_labels1, ''.join('%07s' % a for a in row1)))

              F1      F2      F3      F4      F5      F6      F7      F8      F9      F10     F11     F12     F13
         minm [    0.0     0.0     0.5     0.0     0.4     3.6     2.9     1.1     1.0   187.0    12.6     0.3    1.7]
         maxm [   89.0   100.0    27.7     1.0     0.9     8.8   100.0    12.1    24.0   711.0    22.0   396.9   38.0]
         rang [    5.4    20.0    13.7     0.0     0.2     0.9    57.8     3.7    20.0   393.0     3.6    32.6   11.8]
         mean [    3.6    11.4    11.1     0.1     0.6     6.3    68.6     3.8     9.5   408.2    18.5   356.7   12.7]
         medi [    0.3     0.0     9.7     0.0     0.5     6.2    77.5     3.2     5.0   330.0    19.0   391.4   11.4]
         vari [   73.9   543.9    47.1     0.1     0.0     0.5   792.4     4.4   75.828404.8     4.7  8334.8   51.0]
         50%t [    0.0     0.0     2.9     0.0     0.4     5.6    27.0     1.6     3.0   233.0    14.8   290.3    4.7]
         90%t [   10.5    42.5    19.6     0.0     0.7     7.2    98.8     6.8    24.0   666.0    20.9   396.9   23.0]
```

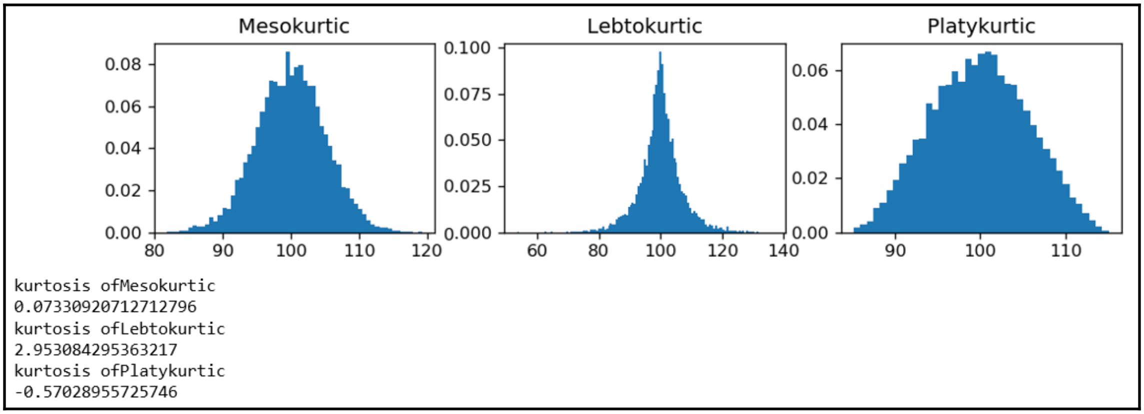
```
Out[51]: array([['3.561-3.740965517241379', '1.0'],
                ['3.740965517241379-3.920931034482758', '1.0'],
                ['3.920931034482758-4.100896551724138', '0.0'],
                ['4.100896551724138-4.280862068965517', '2.0'],
                ['4.280862068965517-4.460827586206896', '1.0'],
                ['4.460827586206896-4.640793103448276', '2.0'],
                ['4.640793103448276-4.820758620689655', '1.0'],
                ['4.820758620689655-5.000724137931034', '8.0'],
                ['5.000724137931034-5.180689655172413', '5.0'],
                ['5.180689655172413-5.360655172413793', '7.0'],
                ['5.360655172413793-5.540620689655173', '17.0'],
                ['5.540620689655173-5.720586206896551', '34.0'],
                ['5.720586206896551-5.900551724137931', '56.0'],
                ['5.900551724137931-6.080517241379310', '65.0'],
                ['6.080517241379310-6.260482758620689', '75.0'],
                ['6.260482758620689-6.440448275862069', '62.0'],
                ['6.440448275862069-6.620413793103448', '42.0'],
                ['6.620413793103448-6.800379310344827', '35.0'],
                ['6.800379310344827-6.980344827586206', '26.0'],
                ['6.980344827586206-7.160310344827586', '16.0'],
                ['7.160310344827586-7.340275862068966', '15.0'],
                ['7.340275862068966-7.520241379310344', '9.0'],
                ['7.520241379310344-7.700206896551724', '4.0'],
                ['7.700206896551724-7.880172413793103', '7.0'],
                ['7.880172413793103-8.060137931034483', '4.0'],
                ['8.060137931034483-8.240103448275860', '1.0'],
                ['8.240103448275860-8.420068965517242', '7.0'],
                ['8.420068965517242-8.600034482758620', '0.0'],
                ['8.600034482758620-8.78', '3.0']], dtype='<U36')
```

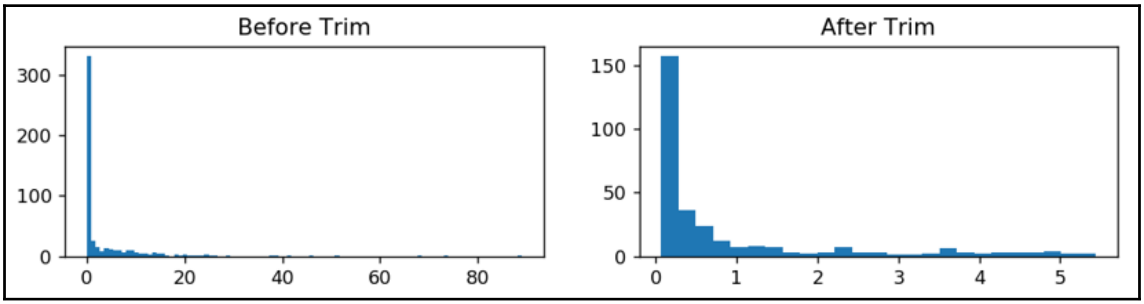


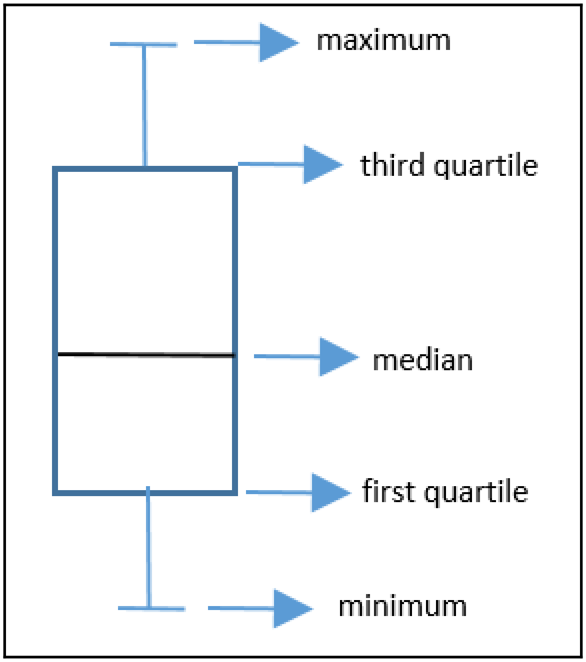
fd
doane
scott
rice
sturges
sqrt
0
20
40
60
25
50
75
100
125
80
150
4
6
8


CRIM
ZN
INDUS
CHAS
NOX
RM
AGE
DIS
RAD
TAX
PTRATIO
B

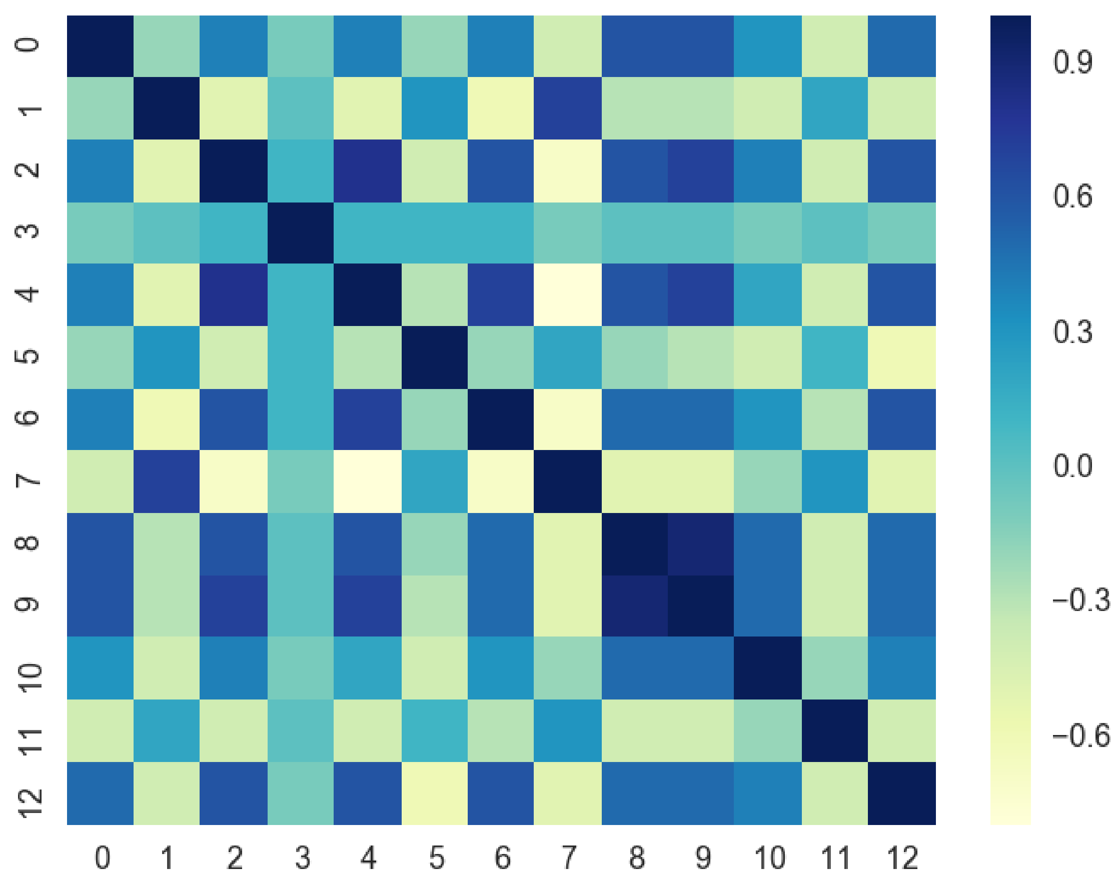

Left Skew
Normal Dist
Right Skew
0.0
0.2
0.4
0.6
0.1
0.3
0
25
50
75
100


Mesokurtic
Lebtokurtic
Platykurtic
0.00
0.02
0.04
0.06
0.08
0.000
0.025
0.050
0.075
0.100
80
90
100
110
120
60
140
kurtosis ofMesokurtic
0.07330920712712796
kurtosis ofLebtokurtic
2.953084295363217
kurtosis ofPlatykurtic
-0.57028955725746

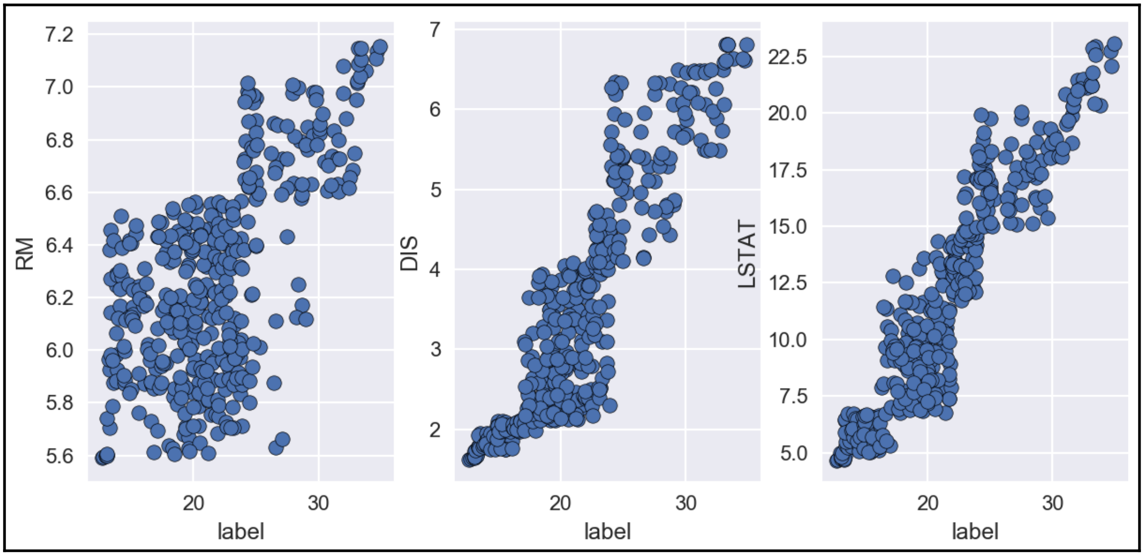

Before Trim
After Trim
0
100
200
300
20
40
60
80
50
150
1
2
3
4
5



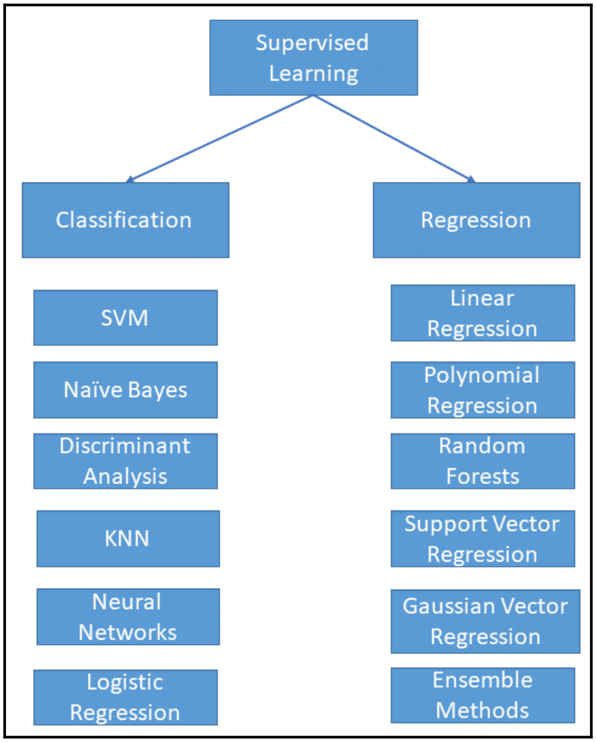

```
Out[61]: array([[ 1. , -0.2,  0.4, -0.1,  0.4, -0.2,  0.4, -0.4,  0.6,  0.6,  0.3, -0.4,  0.5],
                [-0.2,  1. , -0.5, -0. , -0.5,  0.3, -0.6,  0.7, -0.3, -0.3, -0.4,  0.2, -0.4],
                [ 0.4, -0.5,  1. ,  0.1,  0.8, -0.4,  0.6, -0.7,  0.6,  0.7,  0.4, -0.4,  0.6],
                [-0.1, -0. ,  0.1,  1. ,  0.1,  0.1,  0.1, -0.1, -0. , -0. , -0.1,  0. , -0.1],
                [ 0.4, -0.5,  0.8,  0.1,  1. , -0.3,  0.7, -0.8,  0.6,  0.7,  0.2, -0.4,  0.6],
                [-0.2,  0.3, -0.4,  0.1, -0.3,  1. , -0.2,  0.2, -0.2, -0.3, -0.4,  0.1, -0.6],
                [ 0.4, -0.6,  0.6,  0.1,  0.7, -0.2,  1. , -0.7,  0.5,  0.5,  0.3, -0.3,  0.6],
                [-0.4,  0.7, -0.7, -0.1, -0.8,  0.2, -0.7,  1. , -0.5, -0.5, -0.2,  0.3, -0.5],
                [ 0.6, -0.3,  0.6, -0. ,  0.6, -0.2,  0.5, -0.5,  1. ,  0.9,  0.5, -0.4,  0.5],
                [ 0.6, -0.3,  0.7, -0. ,  0.7, -0.3,  0.5, -0.5,  0.9,  1. ,  0.5, -0.4,  0.5],
                [ 0.3, -0.4,  0.4, -0.1,  0.2, -0.4,  0.3, -0.2,  0.5,  0.5,  1. , -0.2,  0.4],
                [-0.4,  0.2, -0.4,  0. , -0.4,  0.1, -0.3,  0.3, -0.4, -0.4, -0.2,  1. , -0.4],
                [ 0.5, -0.4,  0.6, -0.1,  0.6, -0.6,  0.6, -0.5,  0.5,  0.5,  0.4, -0.4,  1. ]])
```

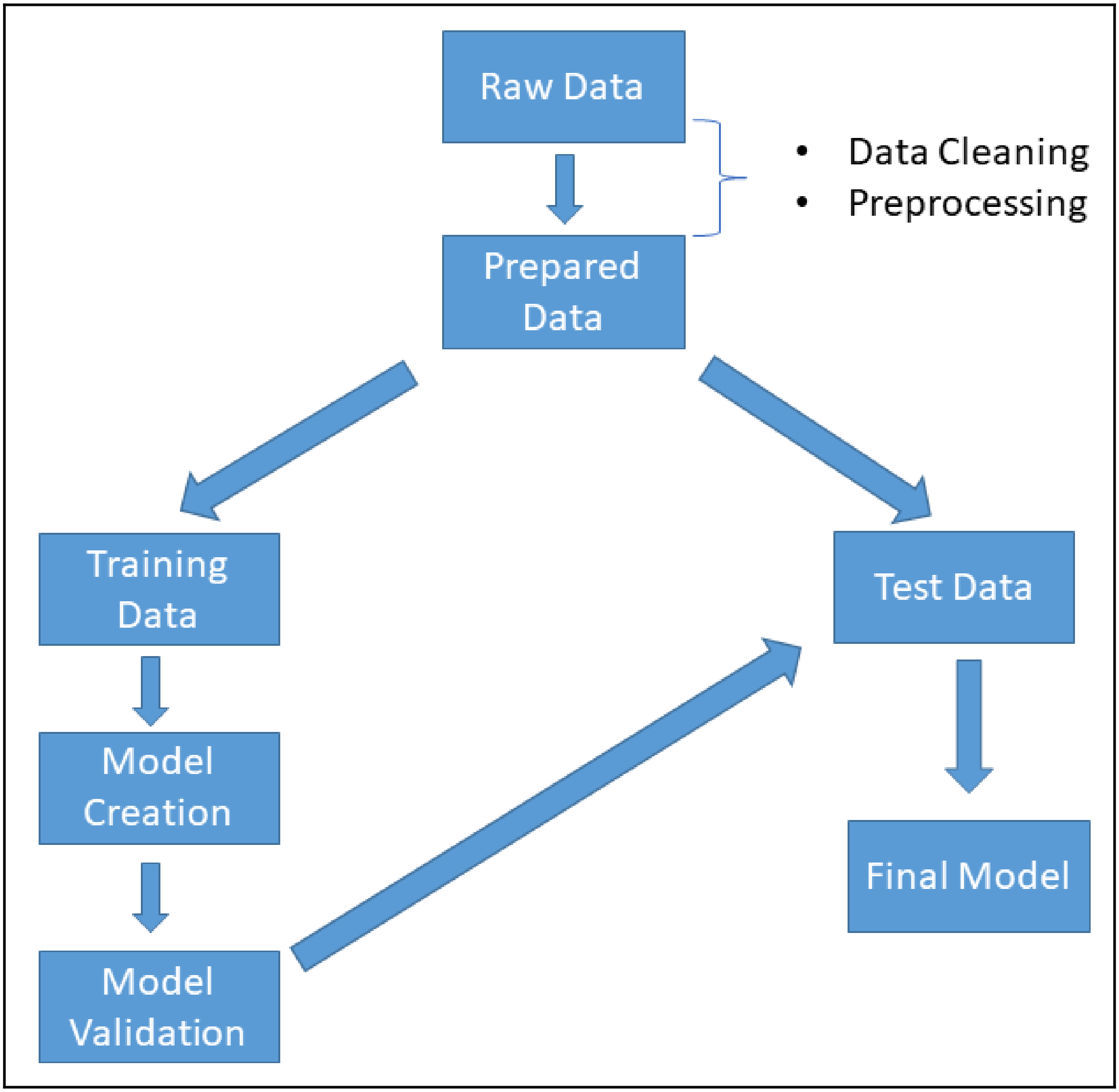



```
In [63]: np.set_printoptions(suppress= True, linewidth= 125)
         CorrelationCoef_Matrix2 = np.round(np.corrcoef(samples, label, rowvar= False), decimals= 2)
         print("     F1     F2     F3     F4     F5    F6      F7     F8     F9    F10    F11    F12    F13")
         print(CorrelationCoef_Matrix2[0:13,13:14].T)

     F1     F2     F3     F4     F5    F6      F7     F8     F9    F10    F11    F12    F13
[[-0.39  0.36 -0.48  0.18 -0.43  0.7   -0.38  0.25 -0.38 -0.47 -0.51  0.33 -0.74]]
```


RM
7.2
7.0
6.8
6.6
6.4
6.2
6.0
5.8
5.6
20
30
label
DIS
7
6
5
4
3
2
20
30
label
LSTAT
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
20
30
label

Chapter 4: Predicting Housing Prices Using Linear Regression




Raw Data
Data Cleaning
Preprocessing
Prepared Data
Training Data
Test Data
Model Creation
Model Validation
Final Model

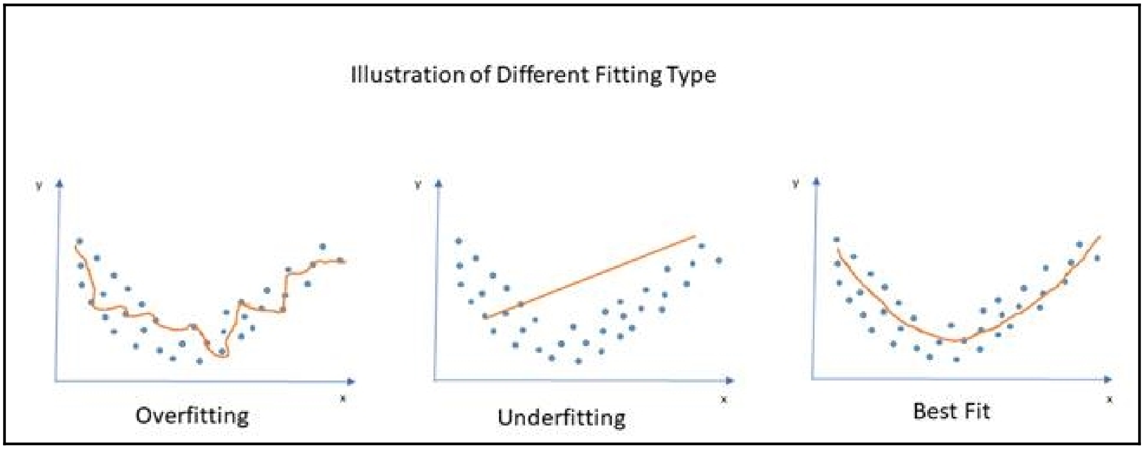
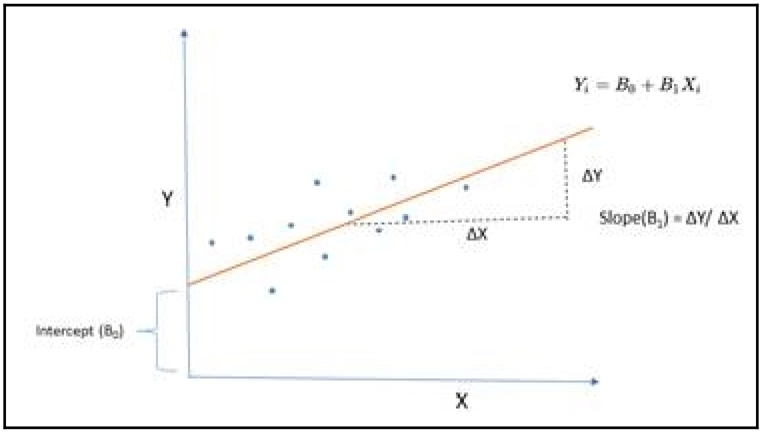
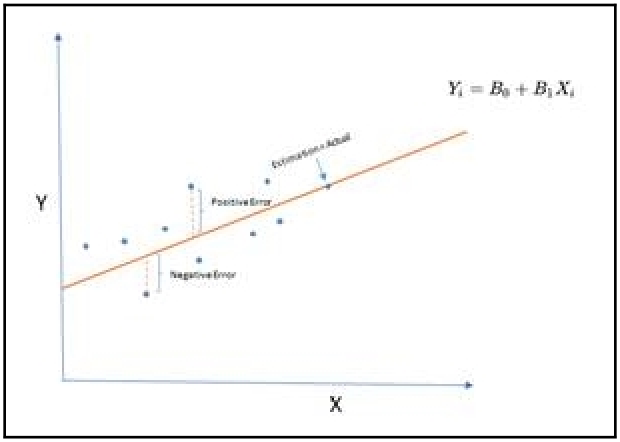
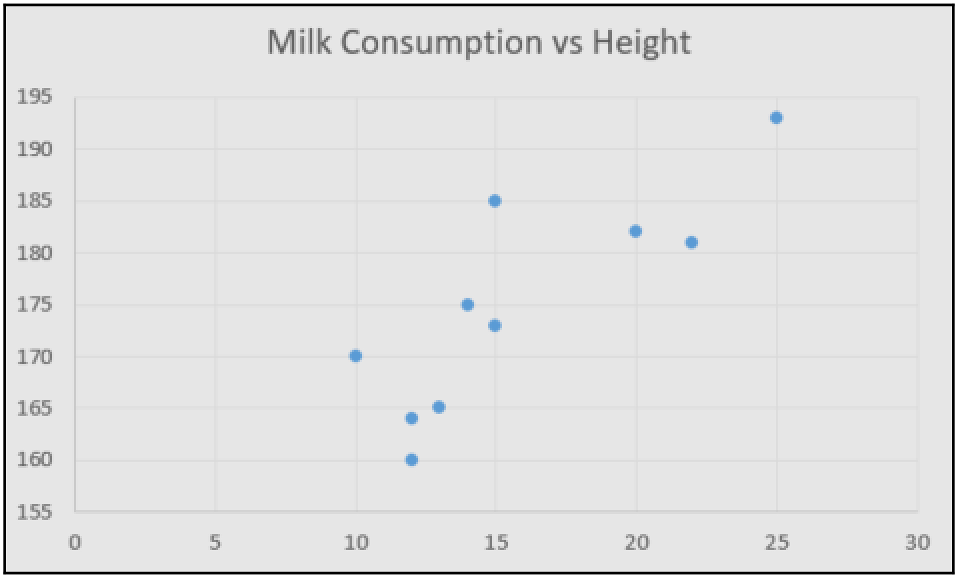


Illustration of Different Fitting Type
y
x
Overfitting
y
x
Underfitting
y
x
Best Fit

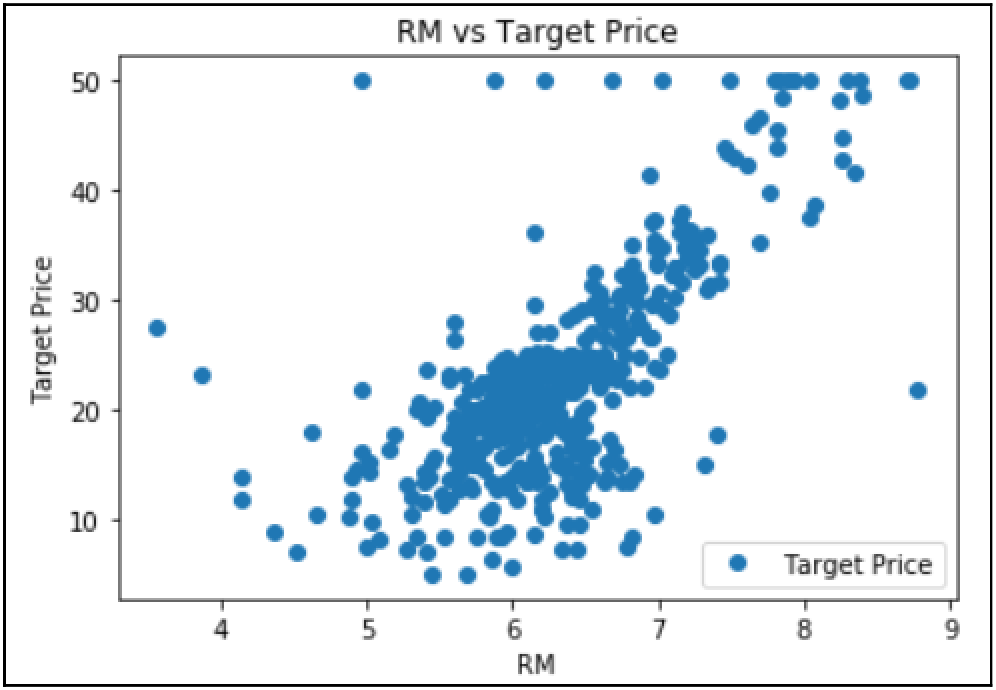

$Y_i = B_0 + B_1 X_i$
Y
ΔY
Slope(B_1) = ΔY/ ΔX
ΔX
Intercept (B_0)
X


$Y_i = B_0 + B_1 X_i$
Y
Positive Error
Negative Error
X




```
In [1]: import numpy as np
        import pandas as pd
        from sklearn.cross_validation import train_test_split
        from sklearn.linear_model import LinearRegression

c:\users\mert_cuhadaroglu\appdata\local\programs\python\python36\lib\site-packages\sklearn\cross_validation.py:41: DeprecationW
arning: This module was deprecated in version 0.18 in favor of the model_selection module into which all the refactored classes
and functions are moved. Also note that the interface of the new CV iterators are different from that of this module. This modu
le will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)

In [3]: from sklearn.datasets import load_boston
        dataset = load_boston()

In [4]: samples, label, feature_names = dataset.data, dataset.target, dataset.feature_names

In [5]: bostondf = pd.DataFrame(dataset.data)

In [6]: bostondf.columns = dataset.feature_names

In [7]: bostondf.head()

Out[7]:
```

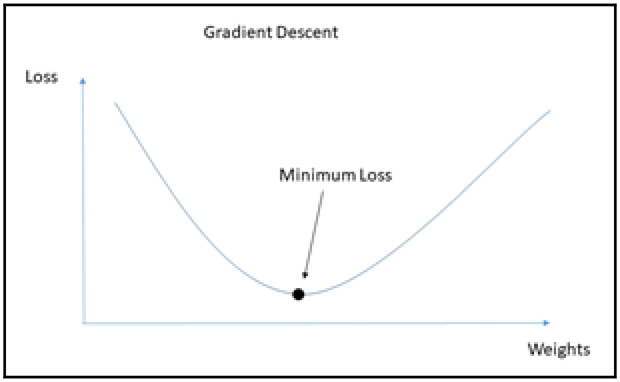
	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

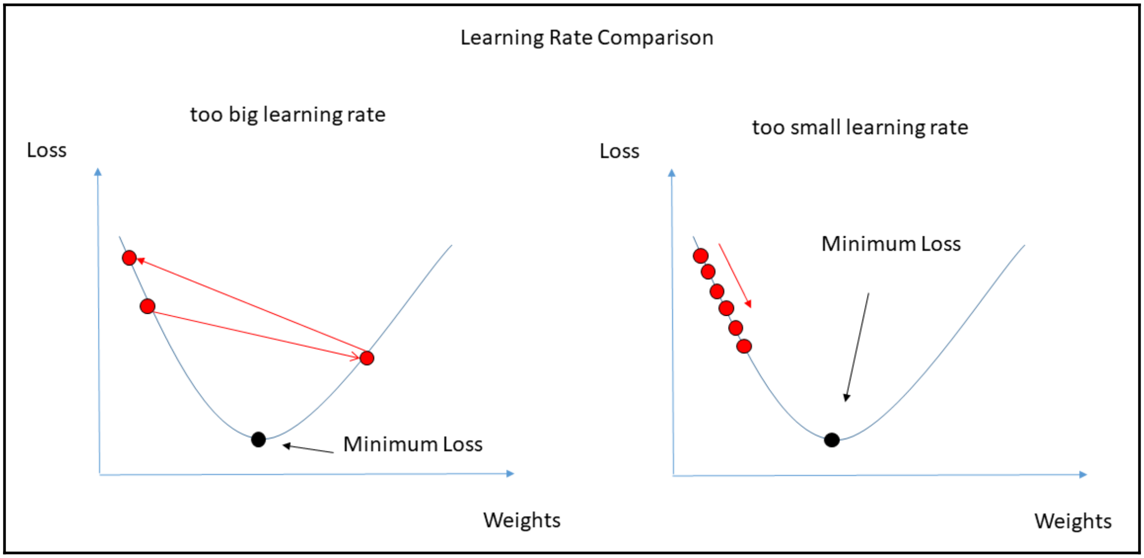


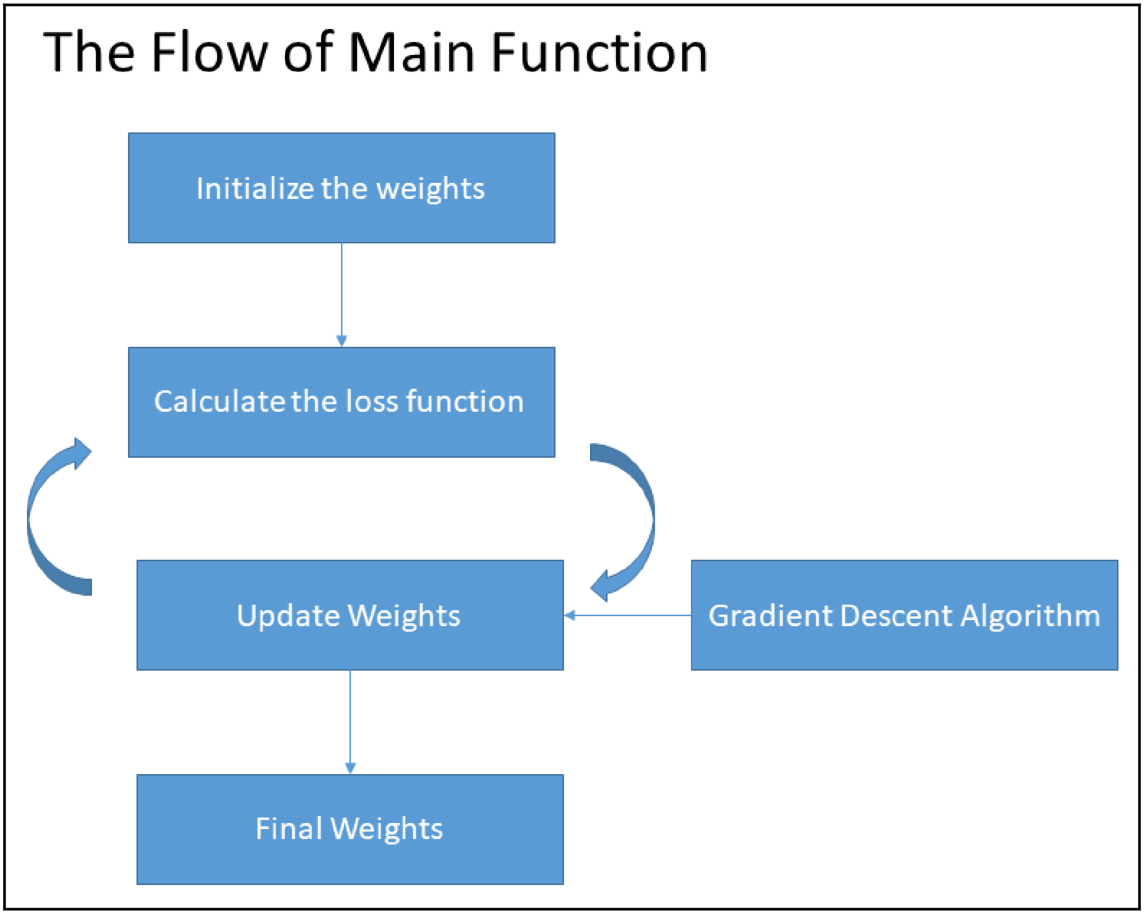

```
In [ ]: def prediction(X, coefficient, intercept):
            return X*coefficient + intercept
```

```
In [ ]: def cost_function(X, Y, coefficient, intercept):
            MSE = 0.0
            for i in range(len(X)):
                MSE += (Y[i] - (coefficient*X[i] + intercept))**2
            return MSE / len(X)
```

```
In [ ]: def update_weights(X, Y, coefficient, intercept, LearningRate):
            coefficient_derivative = 0
            intercept_derivative = 0

            for i in range(len(X)):
                coefficient_derivative += -2*X[i] * (Y[i] - (coefficient*X[i] + intercept))
                intercept_derivative += -2*(Y[i] - (coefficient*X[i] + intercept))

            coefficient -= (coefficient_derivative / len(X)) * LearningRate
            intercept -= (intercept_derivative / len(X)) * LearningRate

            return coefficient, intercept
```





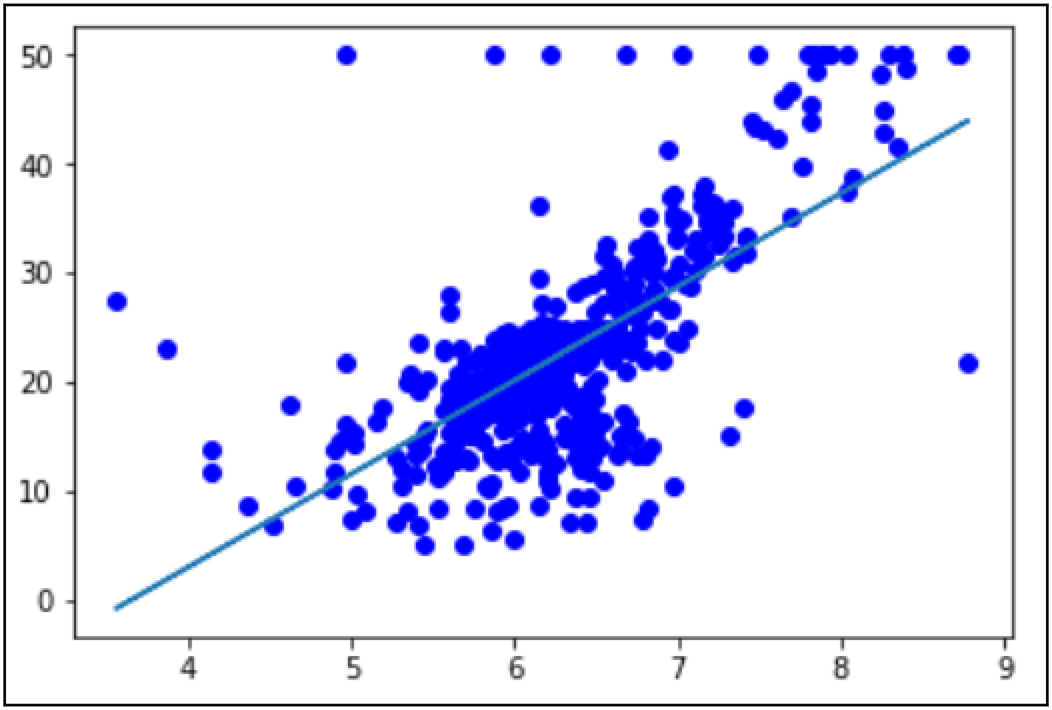



```
In [ ]: def train(X, Y, coefficient, intercept, LearningRate, iteration):
            cost_hist = []
            for i in range(iteration):
                coefficient,intercept = update_weights(X, Y, coefficient, iteration, LearningRate)
                cost = cost_function(X, Y, coefficient, intercept)
                cost_hist.append(cost)
            plt.plot(X, Y, 'bo')
            return coefficient, intercept, cost_hist
```

```
learning_rate = 0.01
iters = 10001
coefficient = 0.3
intercept = 2

weight_f, bias_f, cost_history = train(X, Y, coefficient, intercept, learning_rate, iters)
```

```
weight_f, bias_f, cost_history = train(X, Y, coefficient, intercept= 2, learning_rate =0.01, iters =10001)
```

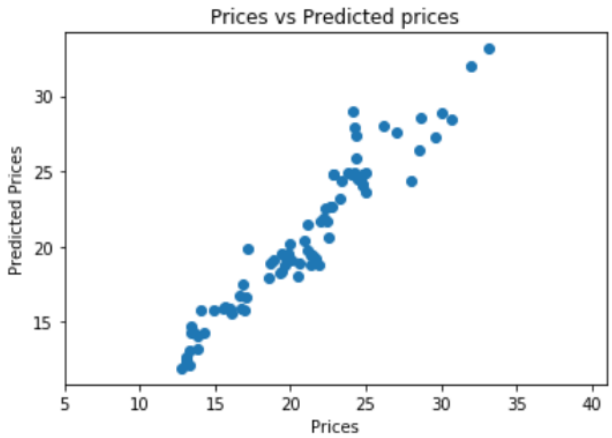
```
In [31]: coefficient

Out[31]: array([8.57526661])

In [32]: intercept

Out[32]: array([-31.31931428])

In [33]: cost_hist

         array([54.20074313]),
         array([54.19564563]),
         array([54.19055059]),
         array([54.18545801]),
         array([54.18036786]),
         array([54.17528017]),
         array([54.17019493]),
         array([54.16511212]),
         array([54.16003177]),
         array([54.15495385]),
         array([54.14987838]),
         array([54.14480535]),
         array([54.13973476]),
         array([54.13466661]),
         array([54.12960089]),
         array([54.12453761]),
         array([54.11947677]),
         array([54.11441836]),
         array([54.10936238]),
```


50
40
30
20
10
0
4
5
6
7
8
9

```
In [115]: import numpy as np
          import pandas as pd
          from scipy import stats
          from sklearn.cross_validation import train_test_split
          from sklearn.linear_model import LinearRegression

In [116]: from sklearn.datasets import load_boston
          dataset = load_boston()

In [117]: samples, label, feature_names = dataset.data, dataset.target, dataset.feature_names

In [118]: samples_trim = stats.trimboth(samples, 0.1)
          label_trim = stats.trimboth(label,0.1)

In [119]: print(samples.shape)
          print(label.shape)

          (506, 13)
          (506,)

In [120]: print(samples_trim.shape)
          print(label_trim.shape)

          (406, 13)
          (406,)
```

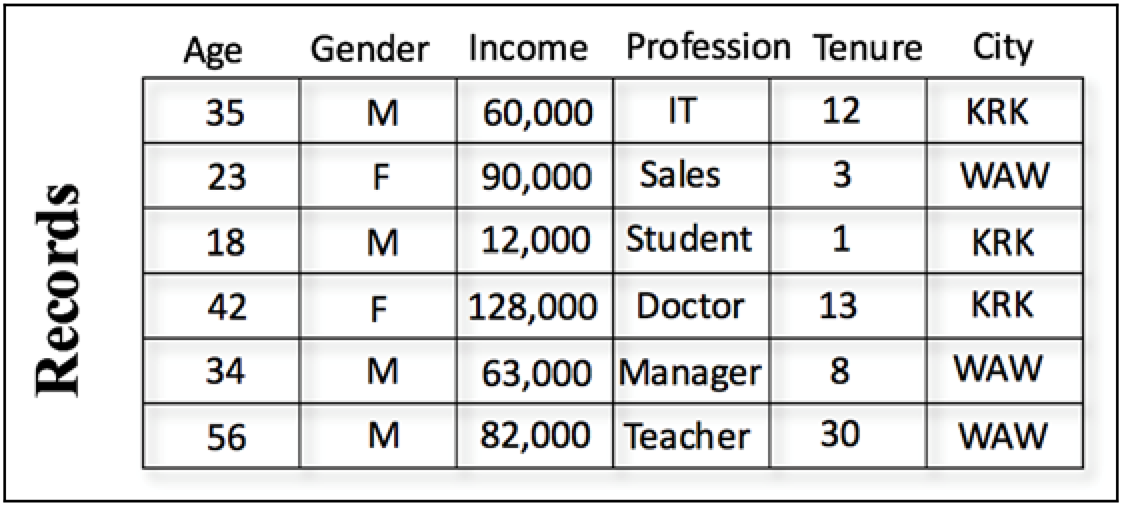
```
In [100]: from sklearn.model_selection import train_test_split
          samples_train, samples_test, label_train, label_test = train_test_split(samples_trim, label_trim, test_size=0.2, random_state=0)

In [101]: print(samples_train.shape)

          print(samples_test.shape)

          print(label_train.shape)

          print(label_test.shape)

          (324, 13)
          (82, 13)
          (324,)
          (82,)

In [102]: regressor = LinearRegression()
          regressor.fit(samples_train, label_train)

Out[102]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)

In [103]: regressor.coef_

Out[103]: array([ 2.12924665e-01,  9.16706914e-02,  1.04316071e-01, -3.18634008e-14,
                  5.34177385e+00, -7.81823481e-02,  1.91366342e-02,  2.81852916e-01,
                  3.19533878e-04, -4.24007416e-03,  1.94206366e-01,  3.96802252e-02,
                  3.81858253e-01])

In [104]: regressor.intercept_

Out[104]: -6.899291747292615
```

```
In [114]: label_pred = regressor.predict(samples_test)

In [126]: plt.scatter(label_test, label_pred)
          plt.xlabel("Prices")
          plt.ylabel("Predicted Prices")
          plt.title("Prices vs Predicted prices")
          plt.axis('equal')

Out[126]: (11.770143369175626, 34.22985663082437, 10.865962968036989, 34.20549738482051)
```

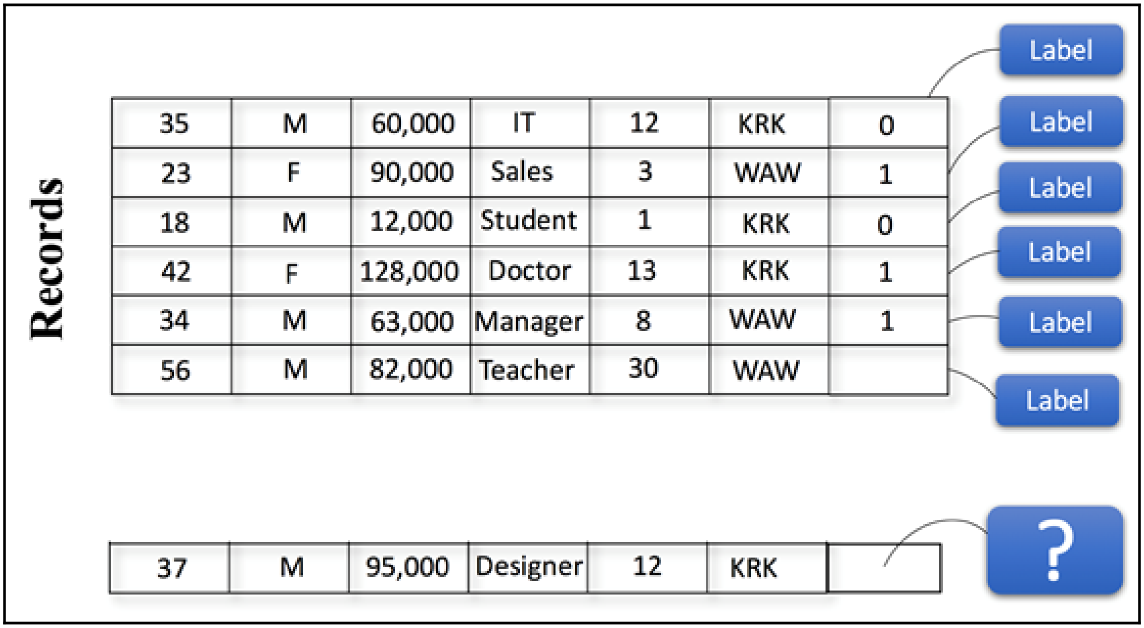



```
In [129]: from sklearn.metrics import mean_squared_error
          from sklearn.metrics import r2_score
          mse = mean_squared_error(label_test, label_pred)
          r2 = r2_score(label_test, label_pred)
          print(mse)
          print(r2)

          2.032691267250478
          0.9154474686142619
```

Chapter 5: Clustering Clients of Wholesale Distributor Using NumPy

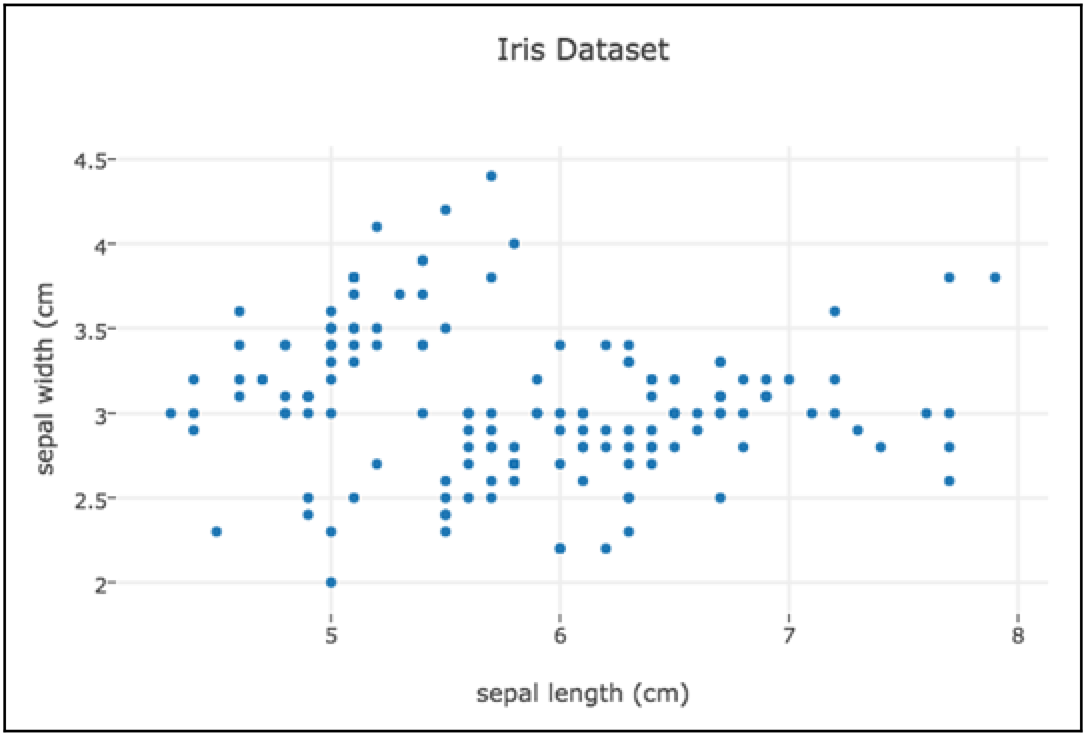


Age	Gender	Income	Profession	Tenure	City
35	M	60,000	IT	12	KRK
23	F	90,000	Sales	3	WAW
18	M	12,000	Student	1	KRK
42	F	128,000	Doctor	13	KRK
34	M	63,000	Manager	8	WAW
56	M	82,000	Teacher	30	WAW

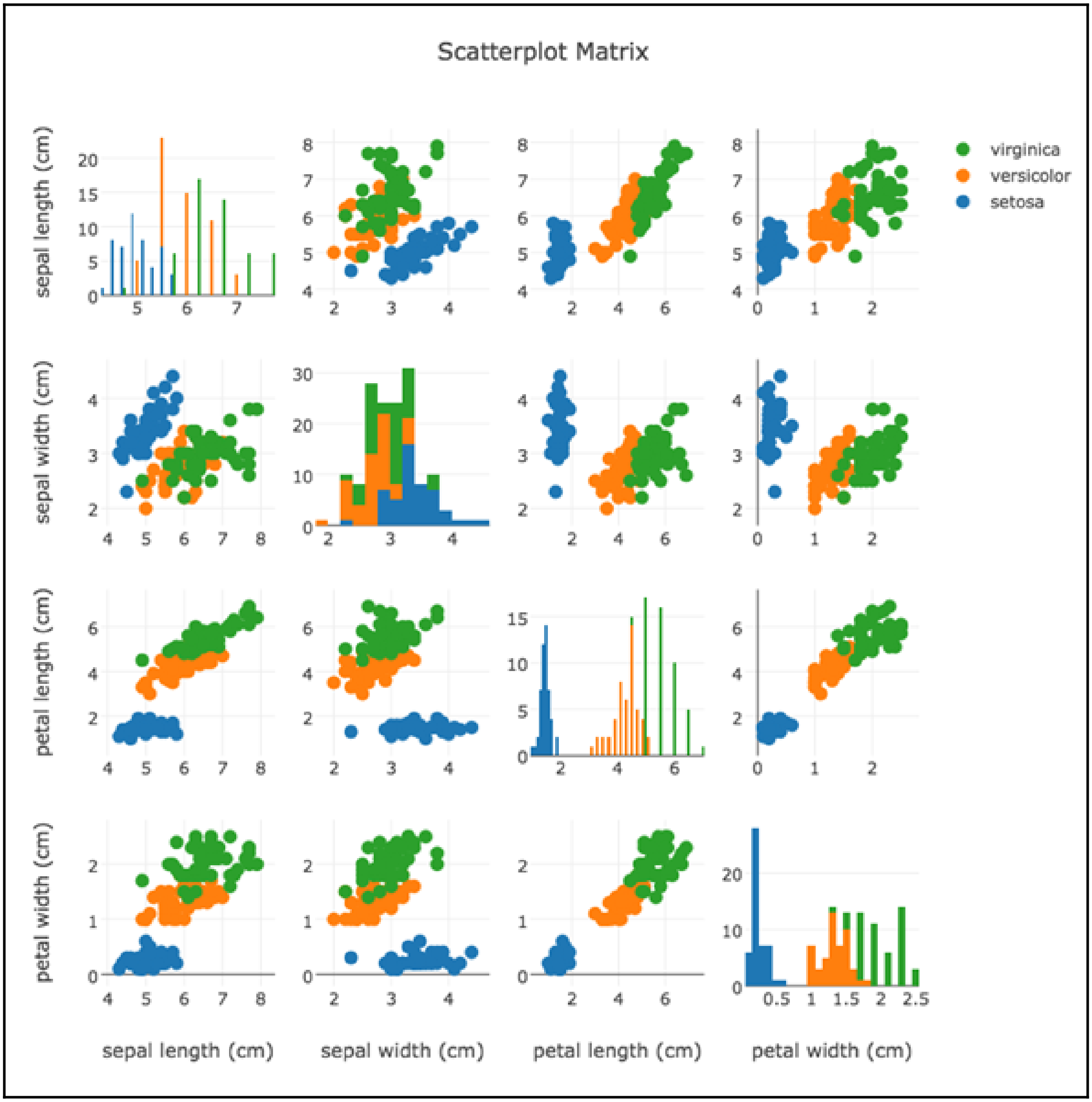


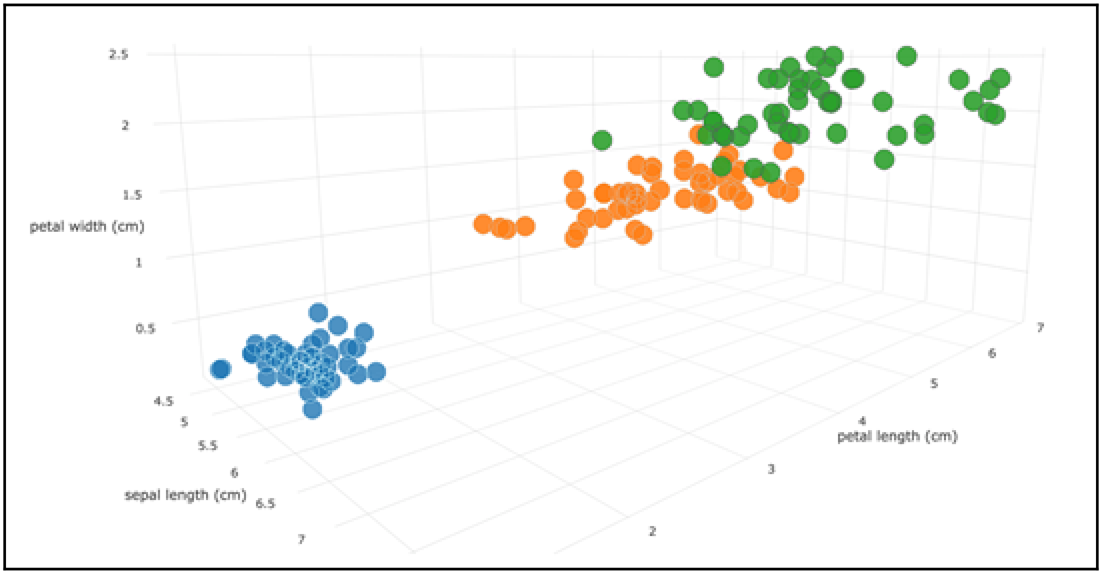
35	M	60,000	IT	12	KRK	0
23	F	90,000	Sales	3	WAW	1
18	M	12,000	Student	1	KRK	0
42	F	128,000	Doctor	13	KRK	1
34	M	63,000	Manager	8	WAW	1
56	M	82,000	Teacher	30	WAW	

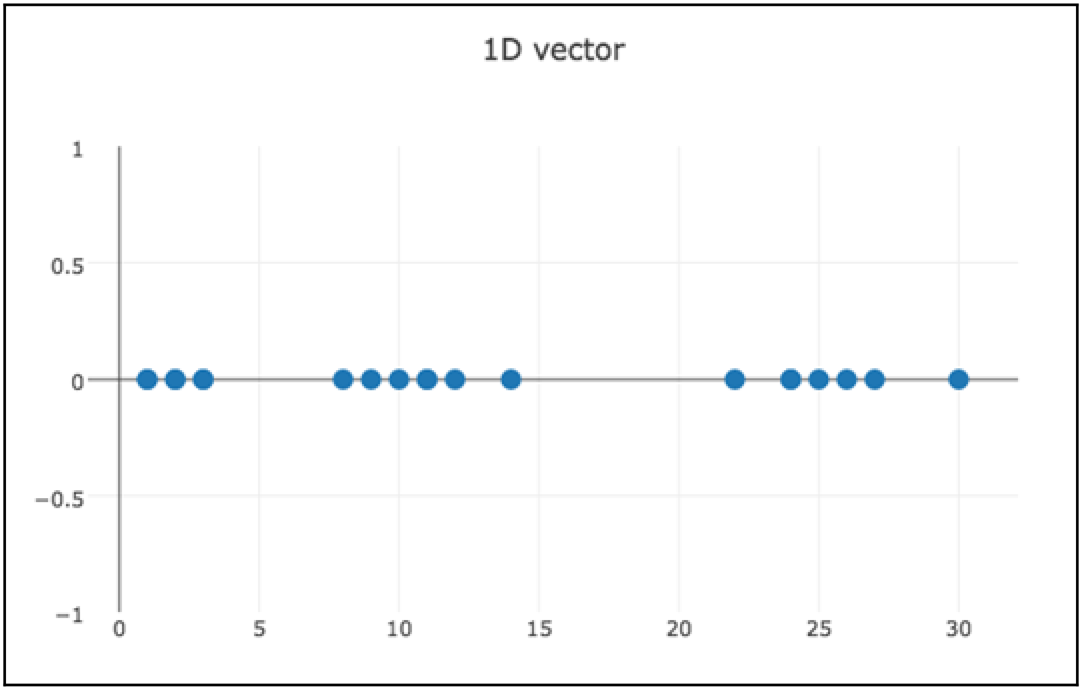
37	M	95,000	Designer	12	KRK	

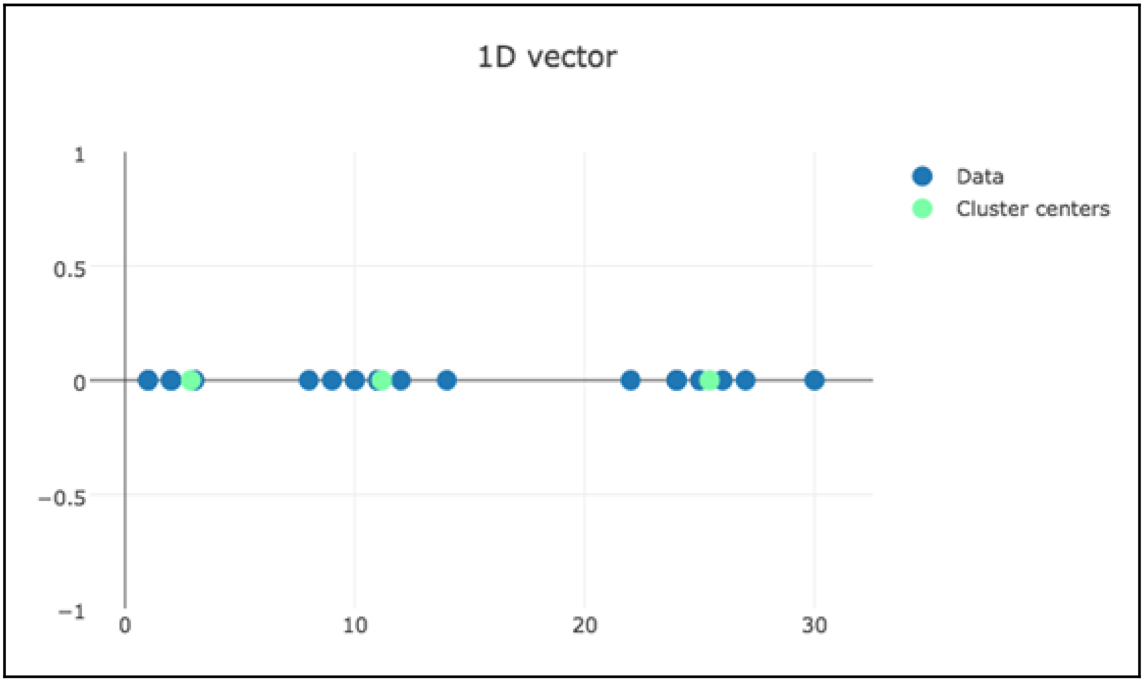


	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

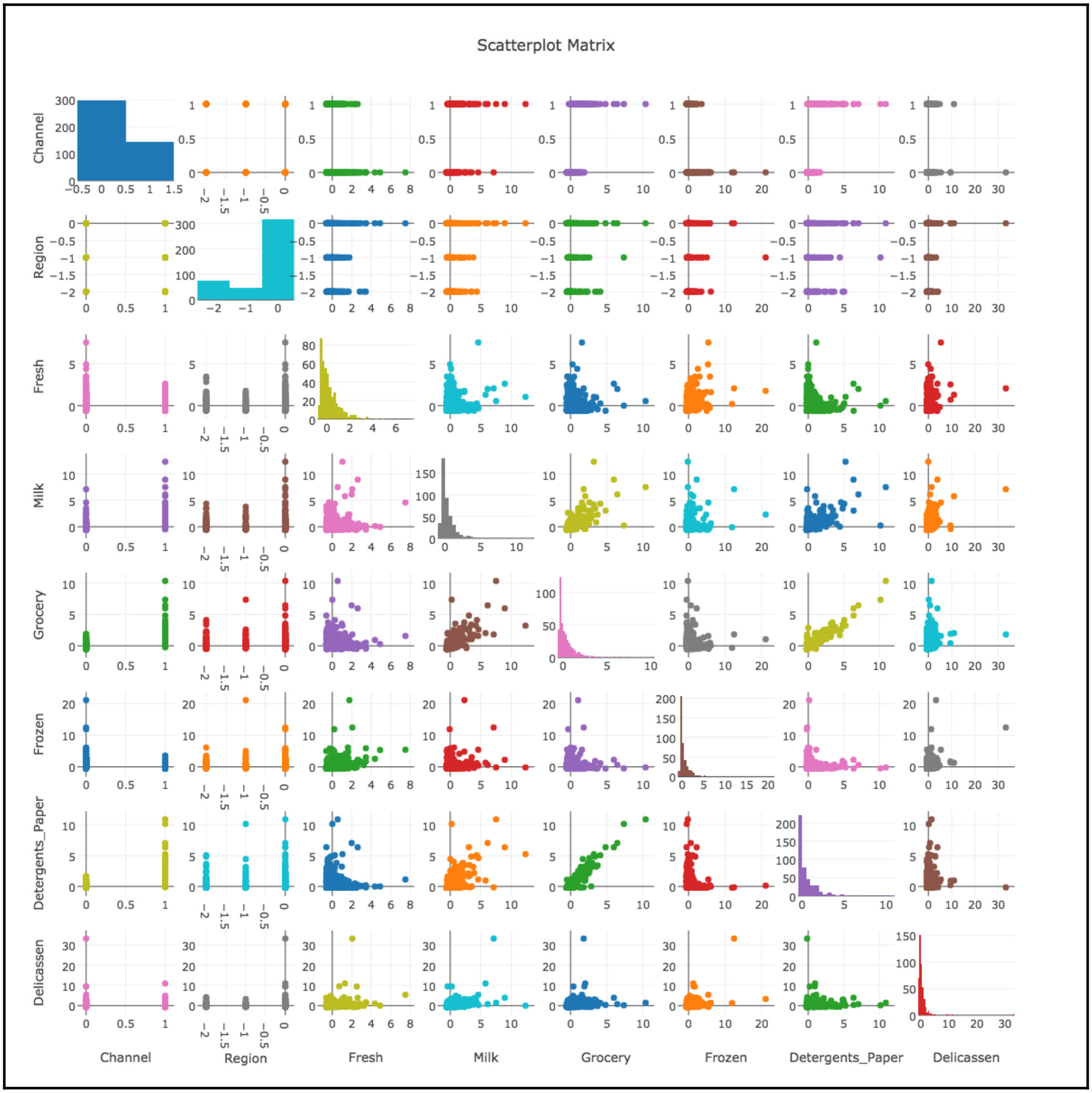

Scatterplot Matrix
virginica
versicolor
setosa
sepal length (cm)
sepal width (cm)
petal length (cm)
petal width (cm)
sepal length (cm)
sepal width (cm)
petal length (cm)
petal width (cm)


2.5
2
1.5
petal width (cm)
1
0.5
4.5
5
5.5
6
sepal length (cm)
6.5
7
2
3
4
petal length (cm)
5
6
7

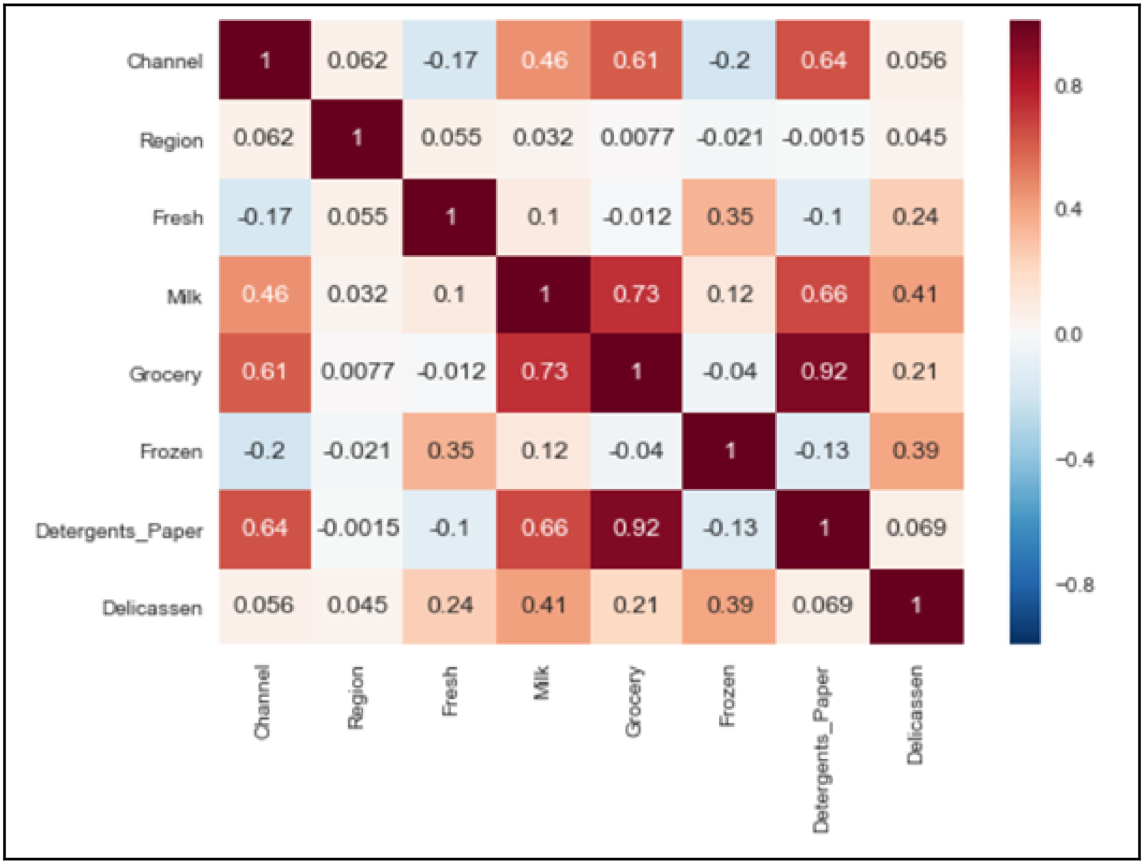

1D vector
1
0.5
0
−0.5
−1
0
5
10
15
20
25
30

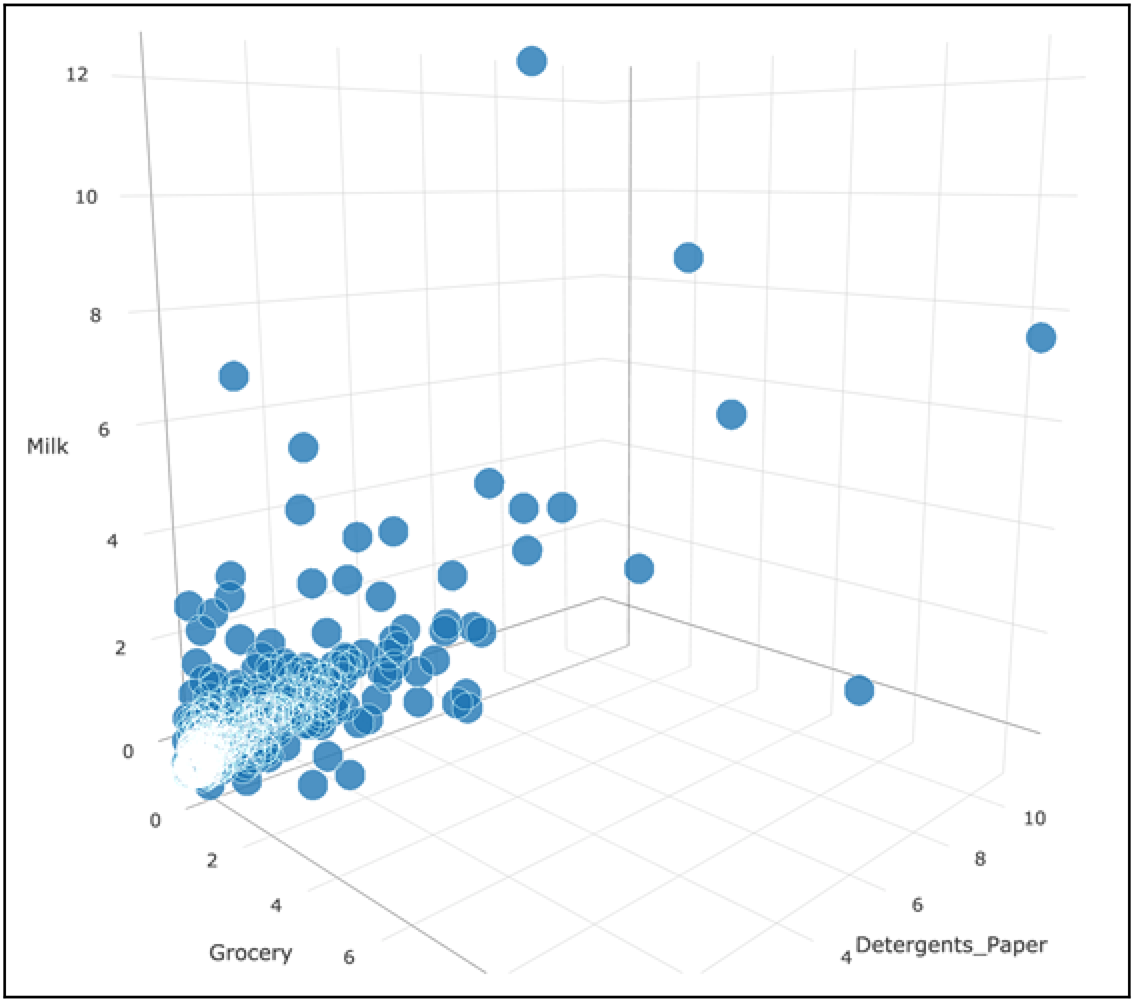


	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	1.0	0.0	0.301680	1.065712	0.329952	-0.466572	0.506787	0.263810
1	1.0	0.0	-0.104810	1.092934	0.565993	0.083926	0.675670	0.574008
2	1.0	0.0	-0.155802	0.915816	0.344418	0.312589	0.736512	4.871459
3	0.0	0.0	0.344850	-0.429714	-0.062862	1.734708	-0.084442	0.582507
4	1.0	0.0	1.022092	0.315171	0.287260	0.849573	0.262056	2.988314
5	1.0	0.0	0.065841	0.818772	0.043574	-0.305832	0.266967	0.343839
6	1.0	0.0	0.262350	-0.075655	0.261033	-0.371977	0.633927	-0.297805
7	1.0	0.0	-0.067000	0.234920	0.549293	0.050853	0.683309	1.133499
8	0.0	0.0	-0.184050	0.003712	0.168945	-0.391536	0.245413	-0.152620
9	1.0	0.0	-0.180936	1.319722	1.661286	-0.130512	1.803015	0.802054

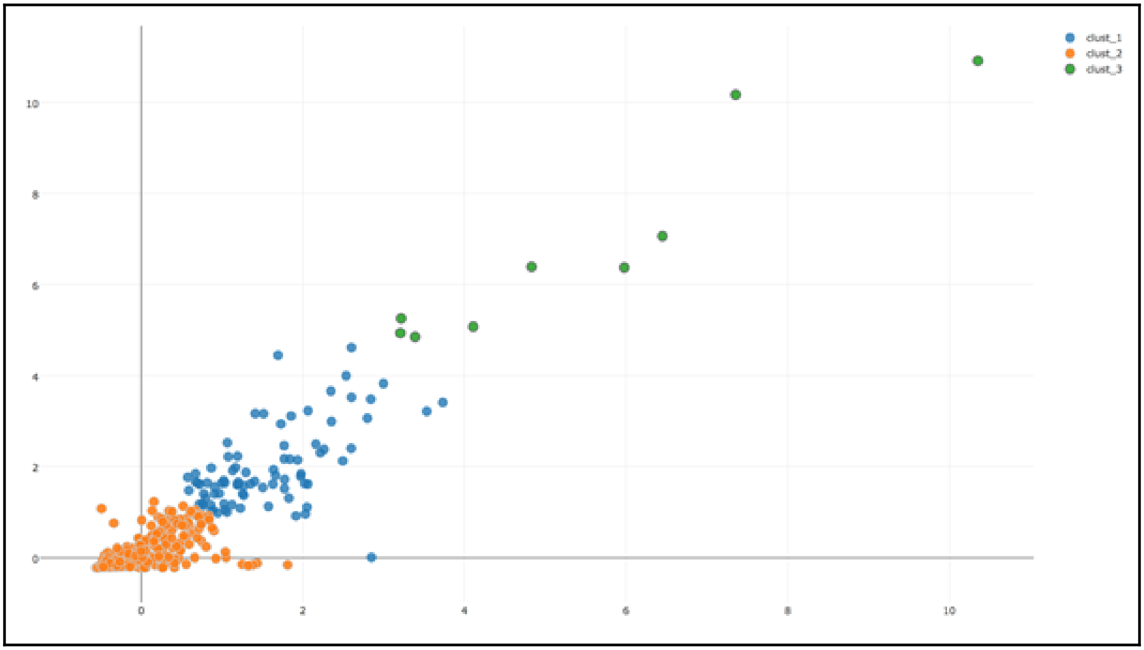

Scatterplot Matrix
Channel
Region
Fresh
Milk
Grocery
Frozen
Detergents_Paper
Delicassen

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
Channel	1.000000	0.062028	-0.169172	0.460720	0.608792	-0.202046	0.636026	0.056011
Region	0.062028	1.000000	0.055287	0.032288	0.007696	-0.021044	-0.001483	0.045212
Fresh	-0.169172	0.055287	1.000000	0.100510	-0.011854	0.345881	-0.101953	0.244690
Milk	0.460720	0.032288	0.100510	1.000000	0.728335	0.123994	0.661816	0.406368
Grocery	0.608792	0.007696	-0.011854	0.728335	1.000000	-0.040193	0.924641	0.205497
Frozen	-0.202046	-0.021044	0.345881	0.123994	-0.040193	1.000000	-0.131525	0.390947
Detergents_Paper	0.636026	-0.001483	-0.101953	0.661816	0.924641	-0.131525	1.000000	0.069291
Delicassen	0.056011	0.045212	0.244690	0.406368	0.205497	0.390947	0.069291	1.000000




12
10
8
6
Milk
4
2
0
0
2
4
Grocery
6
10
8
6
4
Detergents_Paper

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	0.301680	1.065712	0.329952	-0.466572	0.506787	0.263810
1	-0.104810	1.092934	0.565993	0.083926	0.675670	0.574008
2	-0.155802	0.915816	0.344418	0.312589	0.736512	4.871459
3	0.344850	-0.429714	-0.062862	1.734708	-0.084442	0.582507
4	1.022092	0.315171	0.287260	0.849573	0.262056	2.988314
5	0.065841	0.818772	0.043574	-0.305832	0.266967	0.343839
6	0.262350	-0.075655	0.261033	-0.371977	0.633927	-0.297805
7	-0.067000	0.234920	0.549293	0.050853	0.683309	1.133499
8	-0.184050	0.003712	0.168945	-0.391536	0.245413	-0.152620
9	-0.180936	1.319722	1.661286	-0.130512	1.803015	0.802054

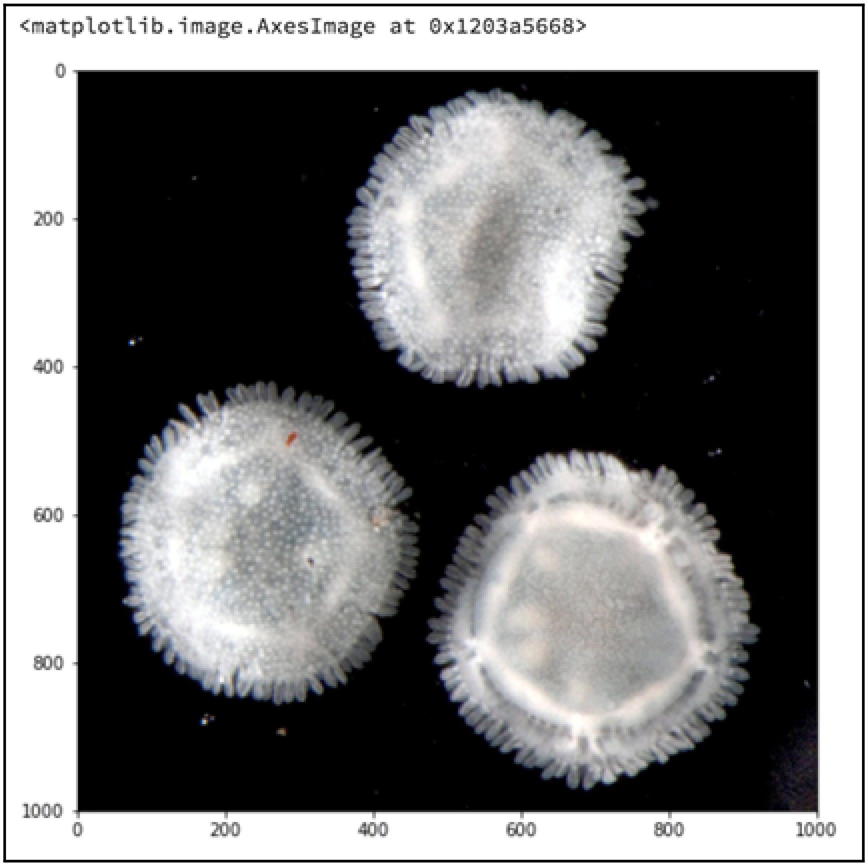


labels	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	-0.070577	1.316079	1.558293	0.037966	1.963503	0.665974
1	0.320582	0.026393	-0.036936	0.671540	0.068682	0.302569
2	0.635915	5.588699	5.432275	0.560929	6.780301	1.526479

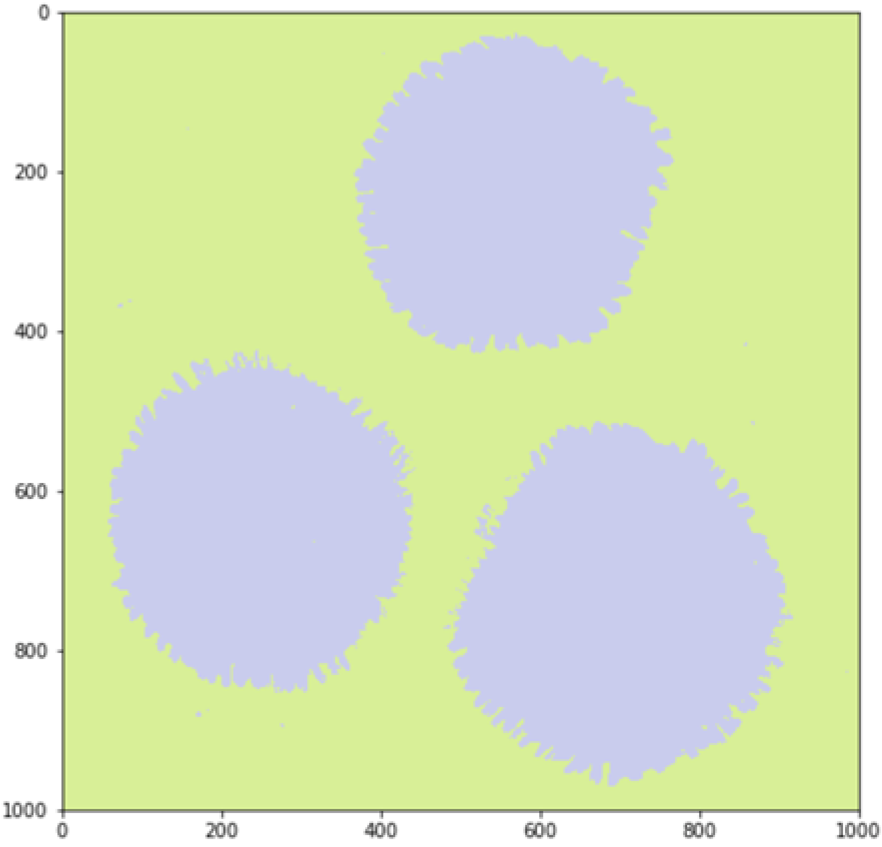
```
In [1]: from sklearn.cluster import KMeans
        KMeans?
```

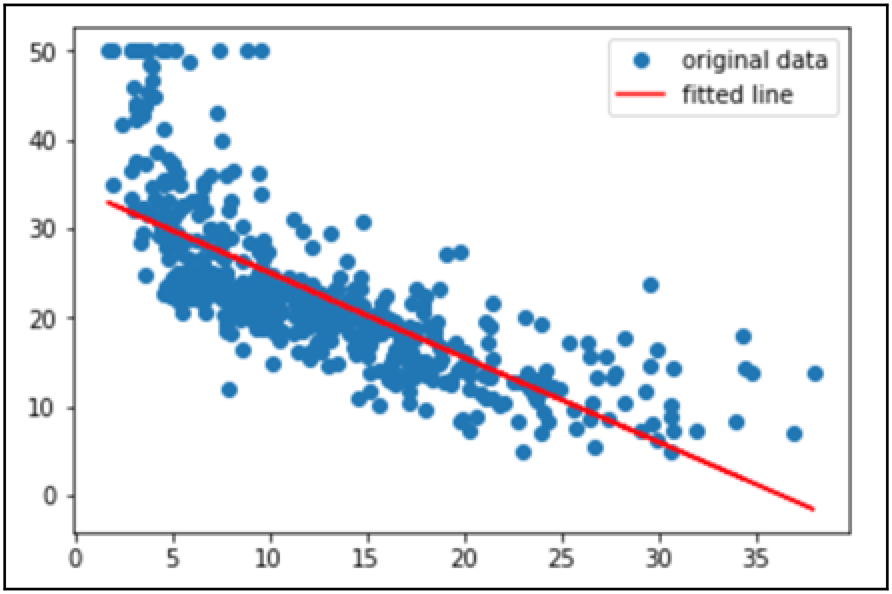
```
Init signature: KMeans(n_clusters=8, init='k-means++', n_init=10, max_iter=300, tol=0.0001, precompute_distances='auto', verbose=0, random_state=None, copy_x=True, n_jobs=1, a
lgorithm='auto')
Docstring:
K-Means clustering

Read more in the :ref:`User Guide <k_means>`.

Parameters
----------

n_clusters : int, optional, default: 8
    The number of clusters to form as well as the number of
    centroids to generate.

init : {'k-means++', 'random' or an ndarray}
    Method for initialization, defaults to 'k-means++':

    'k-means++' : selects initial cluster centers for k-mean
    clustering in a smart way to speed up convergence. See section
    Notes in k_init for more details.

    'random': choose k observations (rows) at random from data for
    the initial centroids.

    If an ndarray is passed, it should be of shape (n_clusters, n_features)
    and gives the initial centers.

n_init : int, default: 10
    Number of time the k-means algorithm will be run with different
    centroid seeds. The final results will be the best output of
    n_init consecutive runs in terms of inertia.

max_iter : int, default: 300
    Maximum number of iterations of the k-means algorithm for a
    single run.

tol : float, default: 1e-4
    Relative tolerance with regards to inertia to declare convergence
```

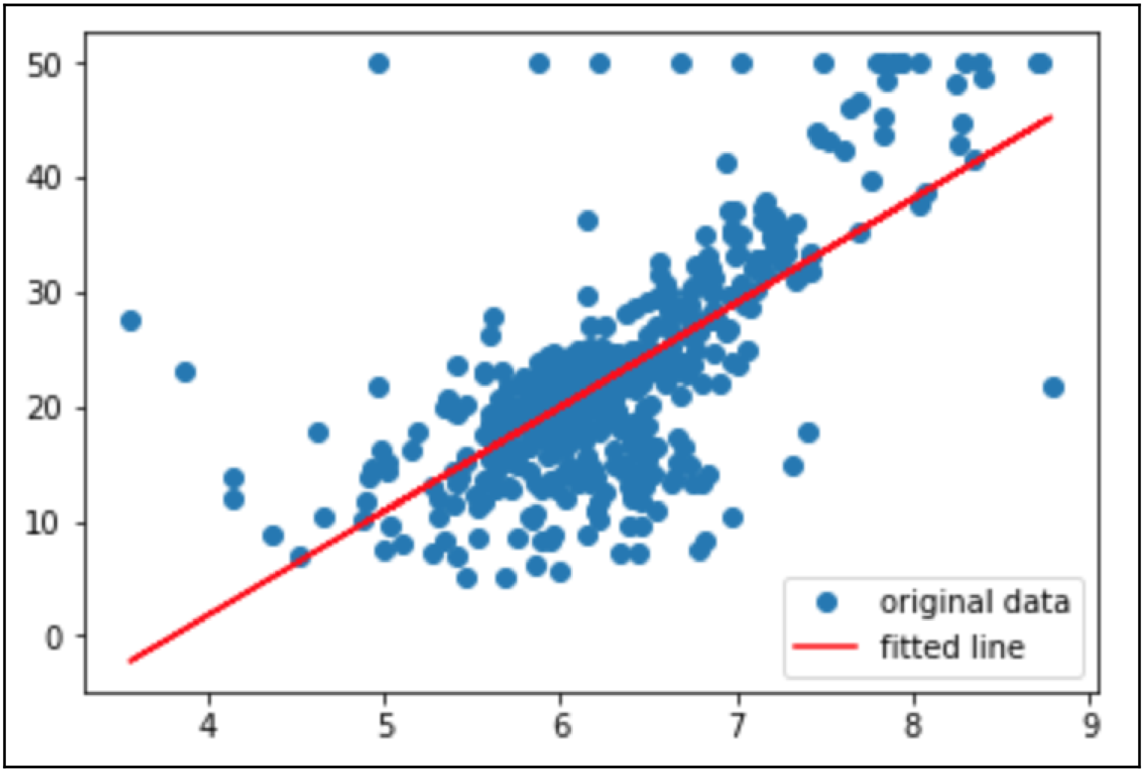
Chapter 6: NumPy, SciPy, Pandas, and Scikit-Learn



```
<matplotlib.image.AxesImage at 0x133c48080>
```




original data
fitted line
50
40
30
20
10
0
0
5
10
15
20
25
30
35

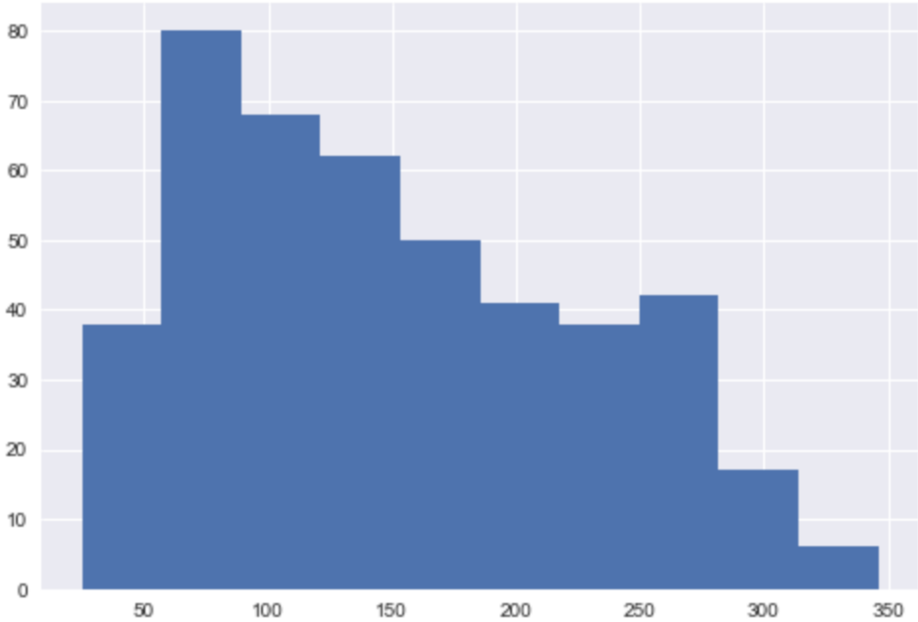


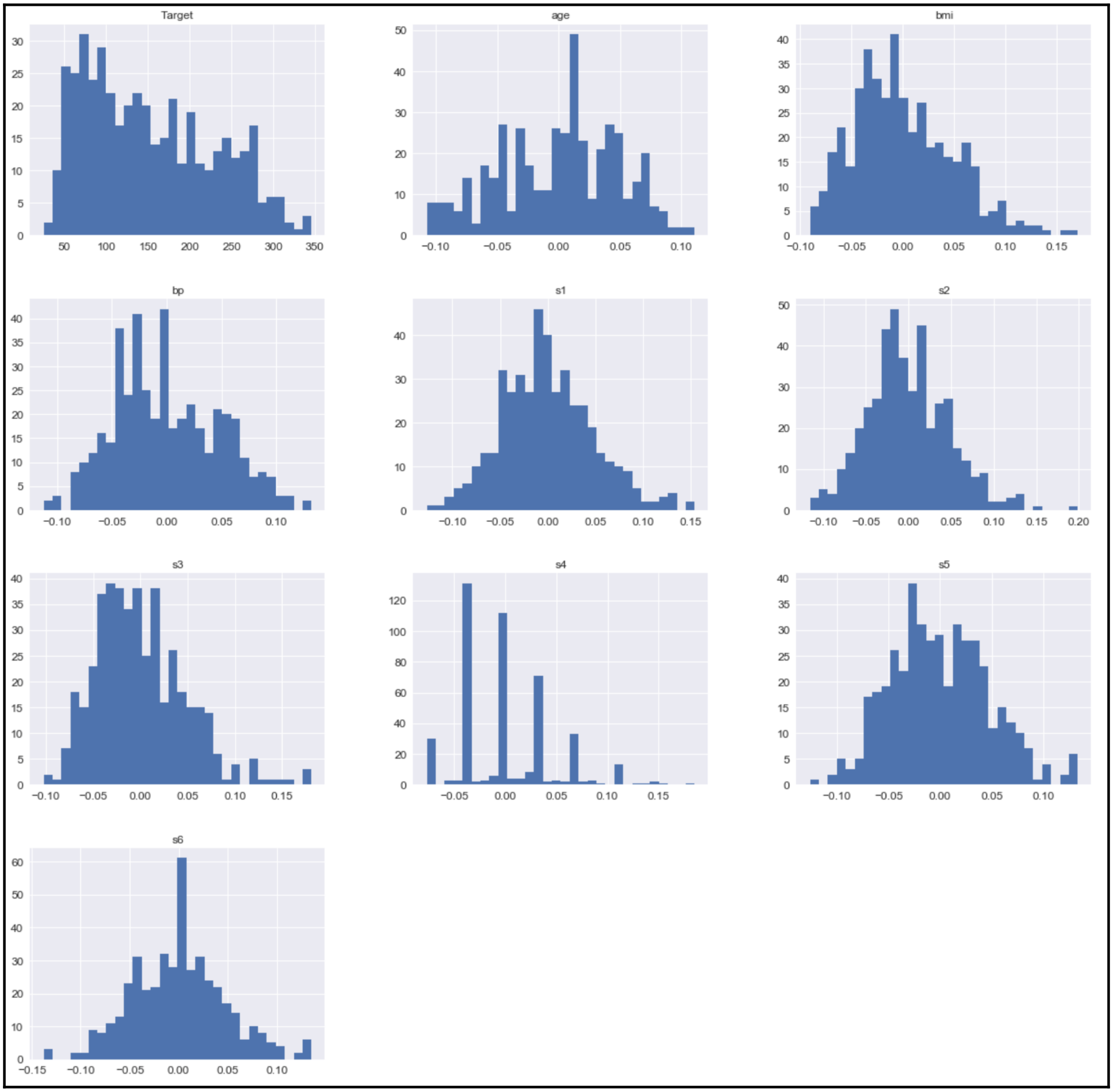
	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6	Target
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.019908	-0.017646	151.0
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.068330	-0.092204	75.0
2	0.085299	0.050680	0.044451	-0.005671	-0.045599	-0.034194	-0.032356	-0.002592	0.002864	-0.025930	141.0
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.022692	-0.009362	206.0
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.031991	-0.046641	135.0
5	-0.092695	-0.044642	-0.040696	-0.019442	-0.068991	-0.079288	0.041277	-0.076395	-0.041180	-0.096346	97.0
6	-0.045472	0.050680	-0.047163	-0.015999	-0.040096	-0.024800	0.000779	-0.039493	-0.062913	-0.038357	138.0
7	0.063504	0.050680	-0.001895	0.066630	0.090620	0.108914	0.022869	0.017703	-0.035817	0.003064	63.0
8	0.041708	0.050680	0.061696	-0.040099	-0.013953	0.006202	-0.028674	-0.002592	-0.014956	0.011349	110.0
9	-0.070900	-0.044642	0.039062	-0.033214	-0.012577	-0.034508	-0.024993	-0.002592	0.067736	-0.013504	310.0

	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6	Target
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.019908	-0.017646	151.0
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.068330	-0.092204	75.0
2	0.085299	0.050680	0.044451	-0.005671	-0.045599	-0.034194	-0.032356	-0.002592	0.002864	-0.025930	141.0
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.022692	-0.009362	206.0
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.031991	-0.046641	135.0
5	-0.092695	-0.044642	-0.040696	-0.019442	-0.068991	-0.079288	0.041277	-0.076395	-0.041180	-0.096346	97.0
6	-0.045472	0.050680	-0.047163	-0.015999	-0.040096	-0.024800	0.000779	-0.039493	-0.062913	-0.038357	138.0
7	0.063504	0.050680	-0.001895	0.066630	0.090620	0.108914	0.022869	0.017703	-0.035817	0.003064	63.0
8	0.041708	0.050680	0.061696	-0.040099	-0.013953	0.006202	-0.028674	-0.002592	-0.014956	0.011349	110.0
9	-0.070900	-0.044642	0.039062	-0.033214	-0.012577	-0.034508	-0.024993	-0.002592	0.067736	-0.013504	310.0

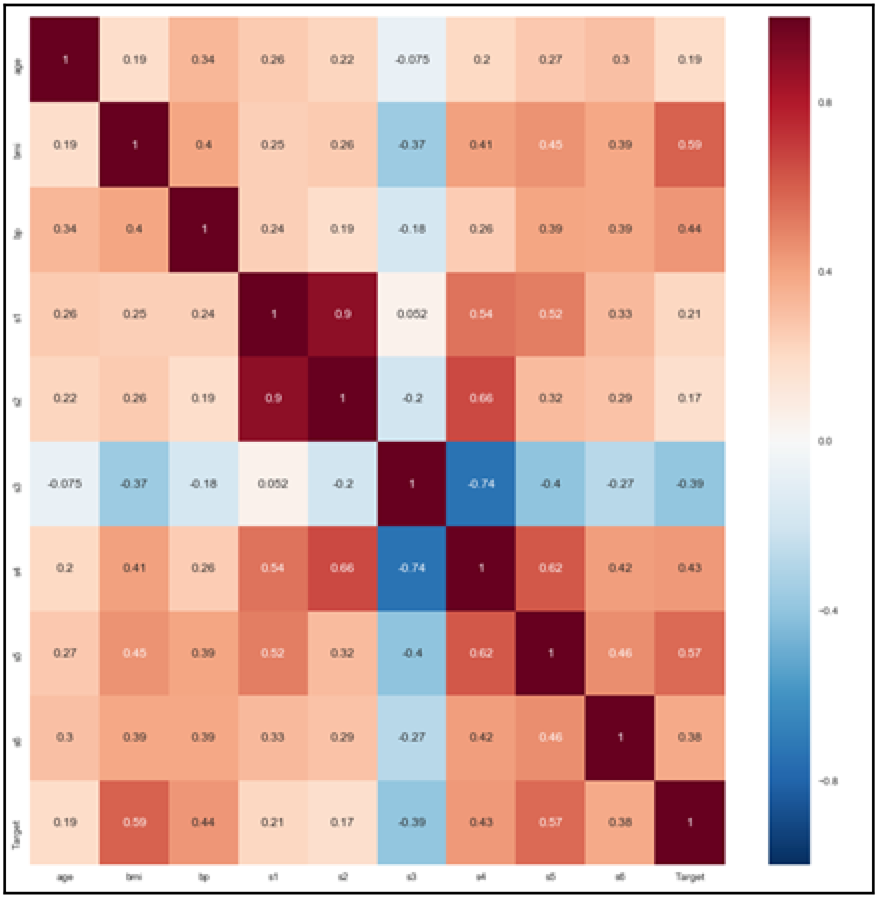
	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6	Target
count	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	442.000000
mean	-3.634285e-16	1.308343e-16	-8.045349e-16	1.281655e-16	-8.835316e-17	1.327024e-16	-4.574646e-16	3.777301e-16	-3.830854e-16	-3.412882e-16	152.133484
std	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	77.093005
min	-1.072256e-01	-4.464164e-02	-9.027530e-02	-1.123996e-01	-1.267807e-01	-1.156131e-01	-1.023071e-01	-7.639450e-02	-1.260974e-01	-1.377672e-01	25.000000
25%	-3.729927e-02	-4.464164e-02	-3.422907e-02	-3.665645e-02	-3.424784e-02	-3.035840e-02	-3.511716e-02	-3.949338e-02	-3.324879e-02	-3.317903e-02	87.000000
50%	5.383060e-03	-4.464164e-02	-7.283766e-03	-5.670611e-03	-4.320866e-03	-3.819065e-03	-6.584468e-03	-2.592262e-03	-1.947634e-03	-1.077698e-03	140.500000
75%	3.807591e-02	5.068012e-02	3.124802e-02	3.564384e-02	2.835801e-02	2.984439e-02	2.931150e-02	3.430886e-02	3.243323e-02	2.791705e-02	211.500000
max	1.107267e-01	5.068012e-02	1.705552e-01	1.320442e-01	1.539137e-01	1.987880e-01	1.811791e-01	1.852344e-01	1.335990e-01	1.356118e-01	346.000000

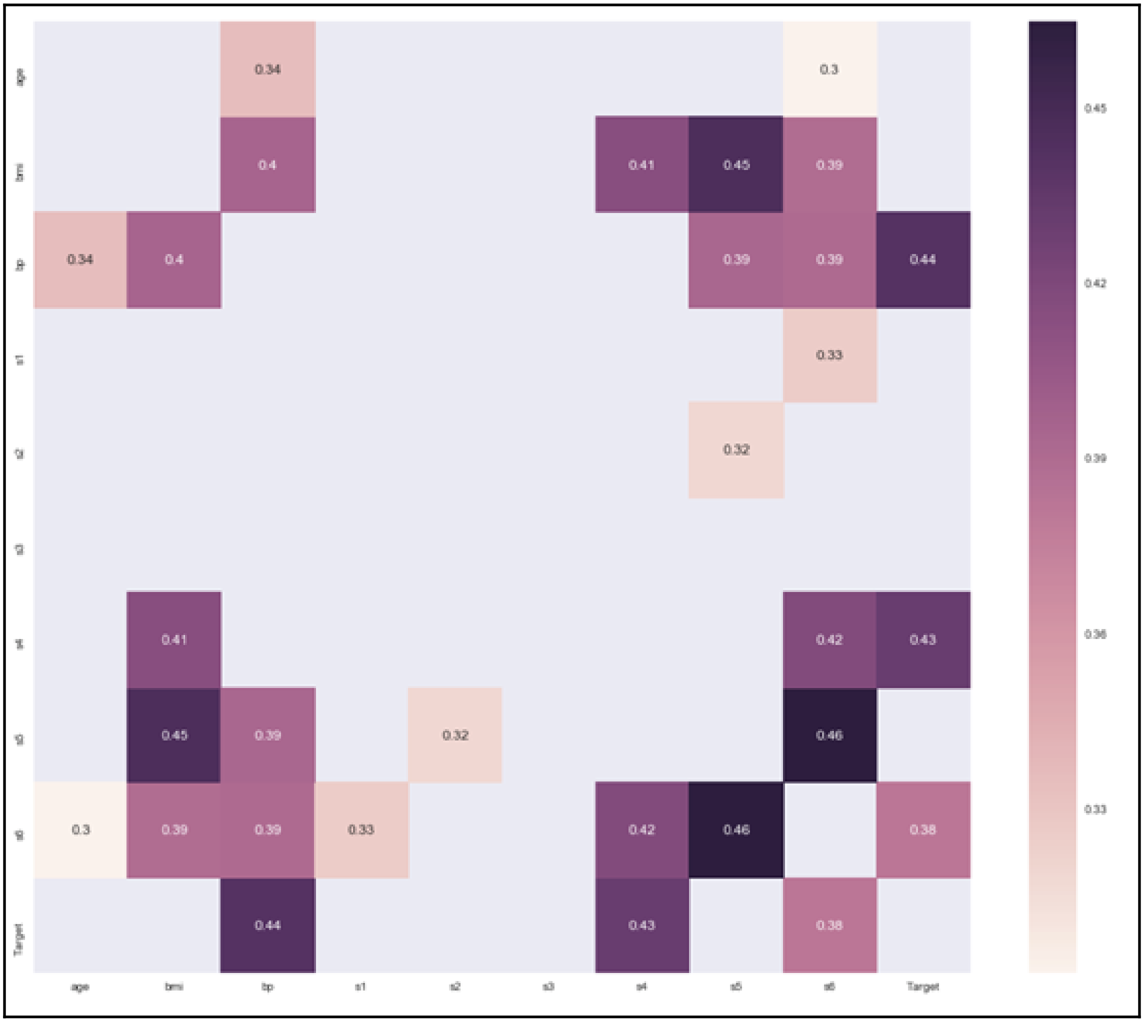
```
(array([38., 80., 68., 62., 50., 41., 38., 42., 17.,  6.]),
 array([ 25. ,  57.1,  89.2, 121.3, 153.4, 185.5, 217.6, 249.7, 281.8,
        313.9, 346. ]),
 <a list of 10 Patch objects>)
```

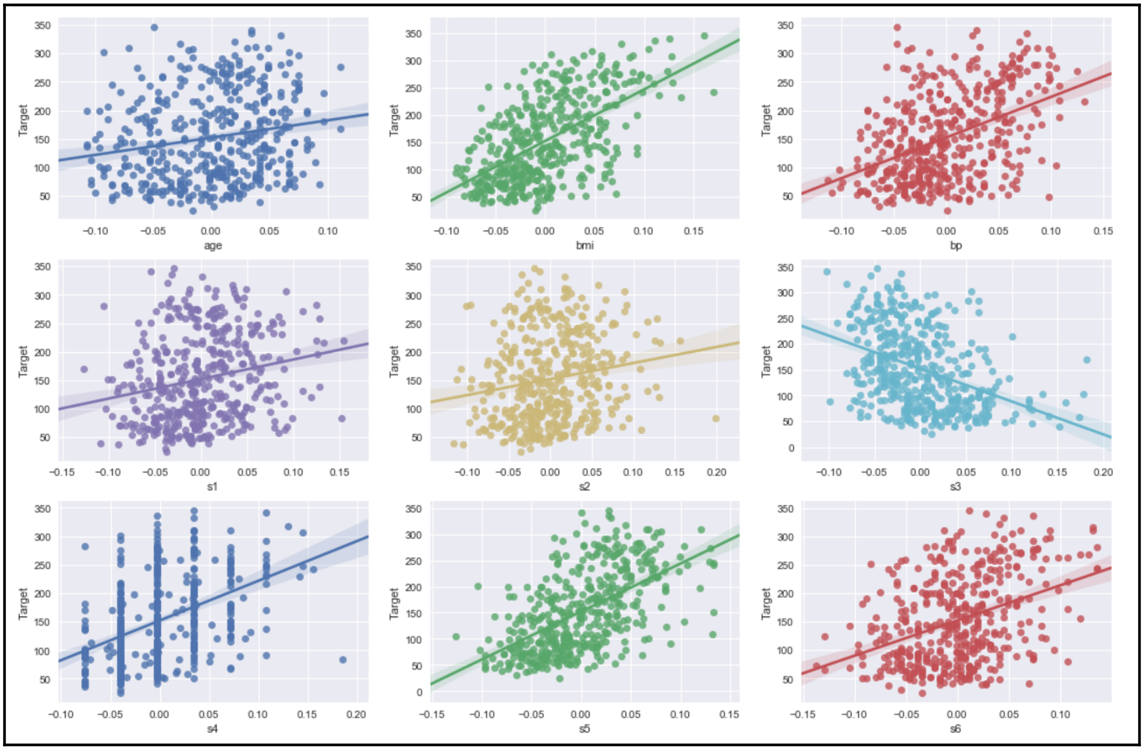



Target
age
bmi
bp
s1
s2
s3
s4
s5
s6

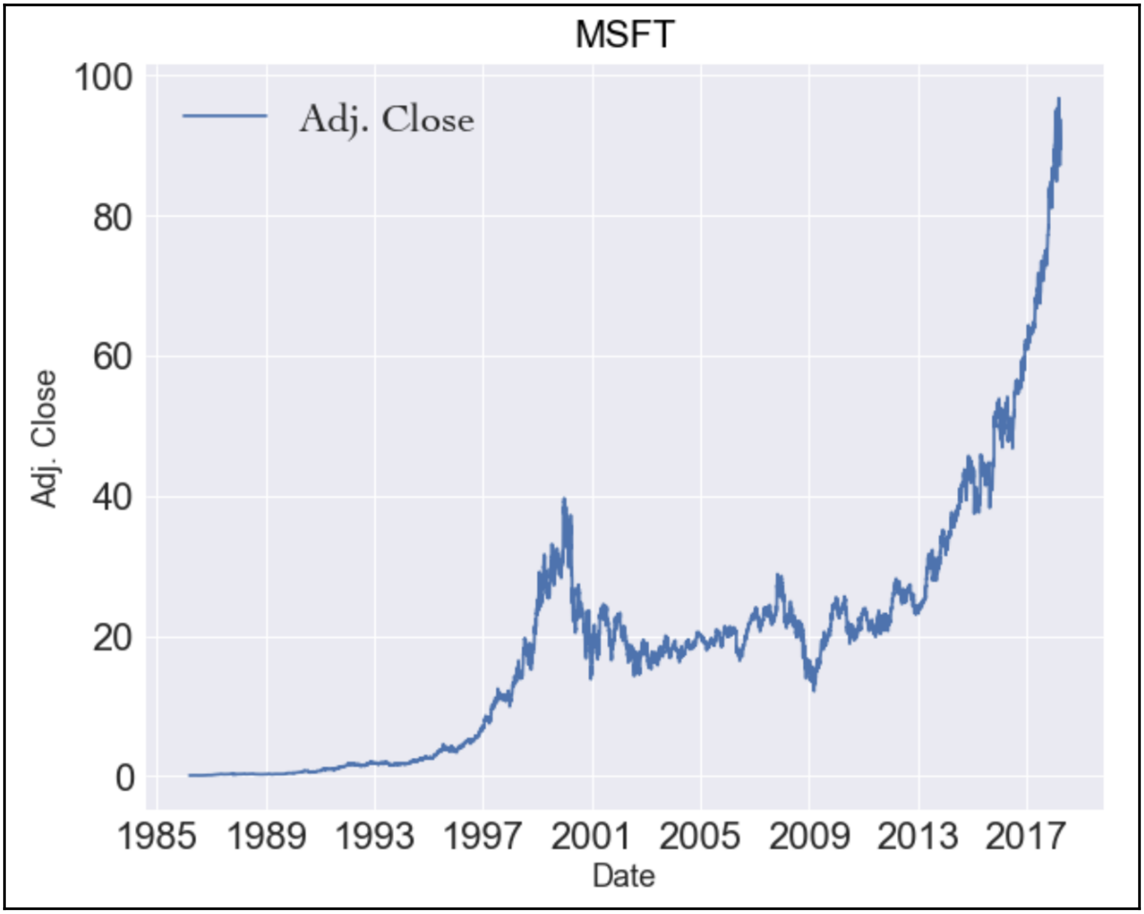
	age	bmi	bp	s1	s2	s3	s4	s5	s6	Target
age	1.000000	0.185085	0.335427	0.260061	0.219243	-0.075181	0.203841	0.270777	0.301731	0.187889
bmi	0.185085	1.000000	0.395415	0.249777	0.261170	-0.366811	0.413807	0.446159	0.388680	0.586450
bp	0.335427	0.395415	1.000000	0.242470	0.185558	-0.178761	0.257653	0.393478	0.390429	0.441484
s1	0.260061	0.249777	0.242470	1.000000	0.896663	0.051519	0.542207	0.515501	0.325717	0.212022
s2	0.219243	0.261170	0.185558	0.896663	1.000000	-0.196455	0.659817	0.318353	0.290600	0.174054
s3	-0.075181	-0.366811	-0.178761	0.051519	-0.196455	1.000000	-0.738493	-0.398577	-0.273697	-0.394789
s4	0.203841	0.413807	0.257653	0.542207	0.659817	-0.738493	1.000000	0.617857	0.417212	0.430453
s5	0.270777	0.446159	0.393478	0.515501	0.318353	-0.398577	0.617857	1.000000	0.464670	0.565883
s6	0.301731	0.388680	0.390429	0.325717	0.290600	-0.273697	0.417212	0.464670	1.000000	0.382483
Target	0.187889	0.586450	0.441484	0.212022	0.174054	-0.394789	0.430453	0.565883	0.382483	1.000000



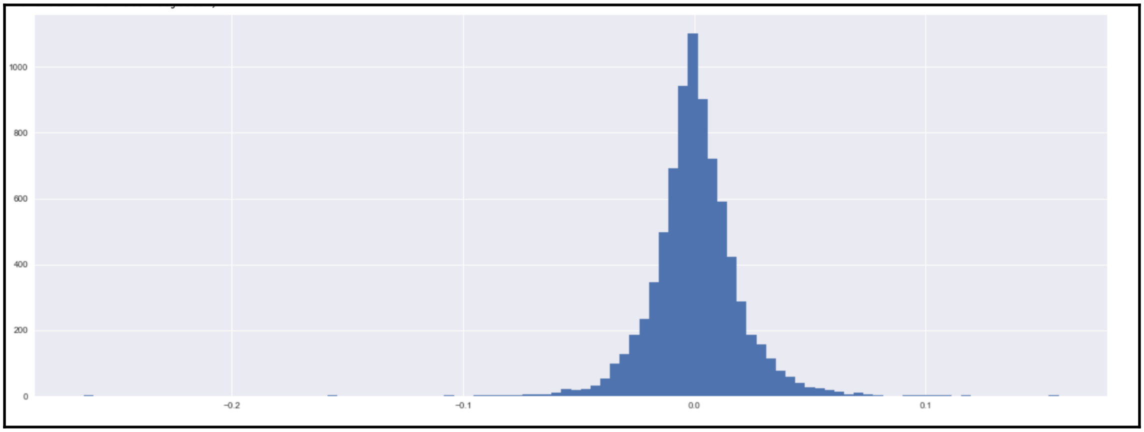

0.34
0.3
0.4
0.41
0.45
0.39
0.34
0.4
0.39
0.39
0.44
0.33
0.32
0.41
0.42
0.43
0.45
0.39
0.32
0.46
0.3
0.39
0.39
0.33
0.42
0.46
0.38
0.44
0.43
0.38
age
bmi
bp
s1
s2
s3
s4
s5
s6
Target
0.45
0.42
0.39
0.36
0.33



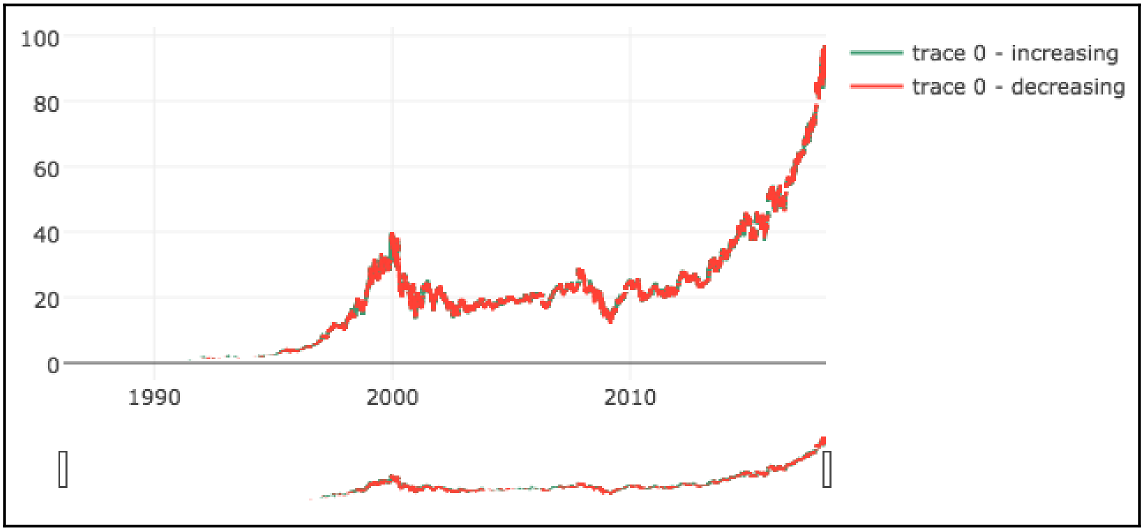
Date	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Low	Adj. Close	Adj. Volume
2018-03-21	92.930	94.050	92.21	92.48	23753263.0	0.0	1.0	92.930	94.050	92.21	92.48	23753263.0
2018-03-22	91.265	91.750	89.66	89.79	37578166.0	0.0	1.0	91.265	91.750	89.66	89.79	37578166.0
2018-03-23	89.500	90.460	87.08	87.18	42159397.0	0.0	1.0	89.500	90.460	87.08	87.18	42159397.0
2018-03-26	90.610	94.000	90.40	93.78	55031149.0	0.0	1.0	90.610	94.000	90.40	93.78	55031149.0
2018-03-27	94.940	95.139	88.51	89.47	53704562.0	0.0	1.0	94.940	95.139	88.51	89.47	53704562.0

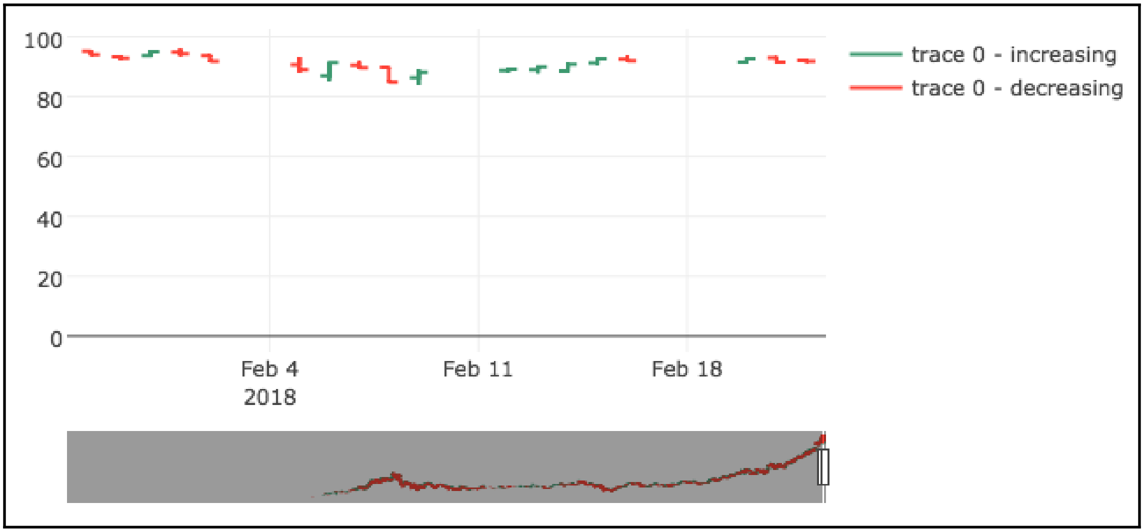


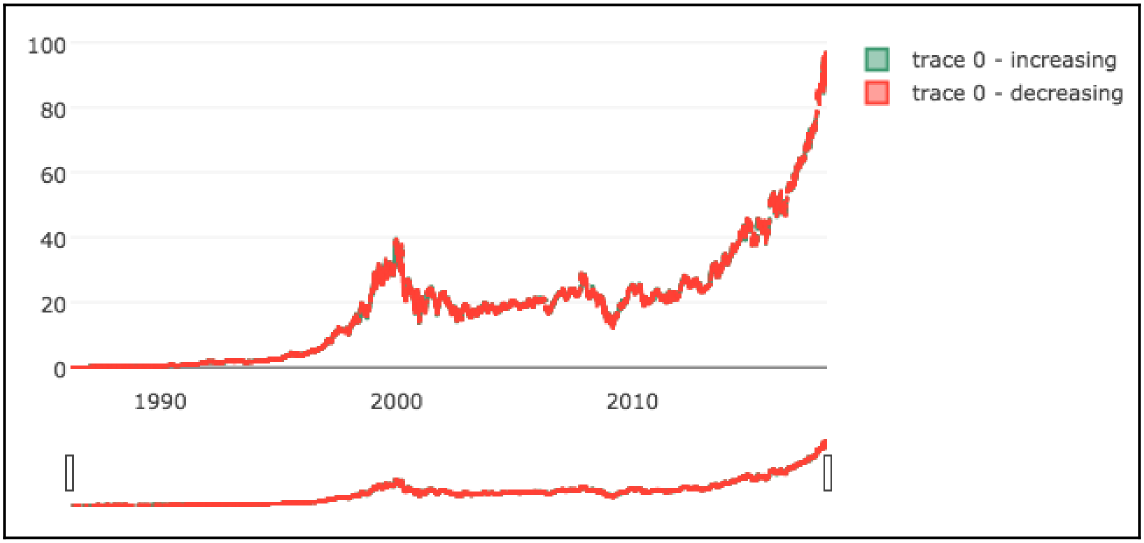
Date	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Low	Adj. Close	Adj. Volume	Daily Pct. Change
2018-03-14	95.120	95.410	93.50	93.85	31576898.0	0.0	1.0	95.120	95.410	93.50	93.85	31576898.0	-0.013352
2018-03-15	93.530	94.580	92.83	94.18	26279014.0	0.0	1.0	93.530	94.580	92.83	94.18	26279014.0	0.006950
2018-03-16	94.680	95.380	93.92	94.60	47329521.0	0.0	1.0	94.680	95.380	93.92	94.60	47329521.0	-0.000845
2018-03-19	93.740	93.900	92.11	92.89	31752589.0	0.0	1.0	93.740	93.900	92.11	92.89	31752589.0	-0.009068
2018-03-20	93.050	93.770	93.00	93.13	21787780.0	0.0	1.0	93.050	93.770	93.00	93.13	21787780.0	0.000860
2018-03-21	92.930	94.050	92.21	92.48	23753263.0	0.0	1.0	92.930	94.050	92.21	92.48	23753263.0	-0.004842
2018-03-22	91.265	91.750	89.66	89.79	37578166.0	0.0	1.0	91.265	91.750	89.66	89.79	37578166.0	-0.016162
2018-03-23	89.500	90.460	87.08	87.18	42159397.0	0.0	1.0	89.500	90.460	87.08	87.18	42159397.0	-0.025922
2018-03-26	90.610	94.000	90.40	93.78	55031149.0	0.0	1.0	90.610	94.000	90.40	93.78	55031149.0	0.034985
2018-03-27	94.940	95.139	88.51	89.47	53704562.0	0.0	1.0	94.940	95.139	88.51	89.47	53704562.0	-0.057615

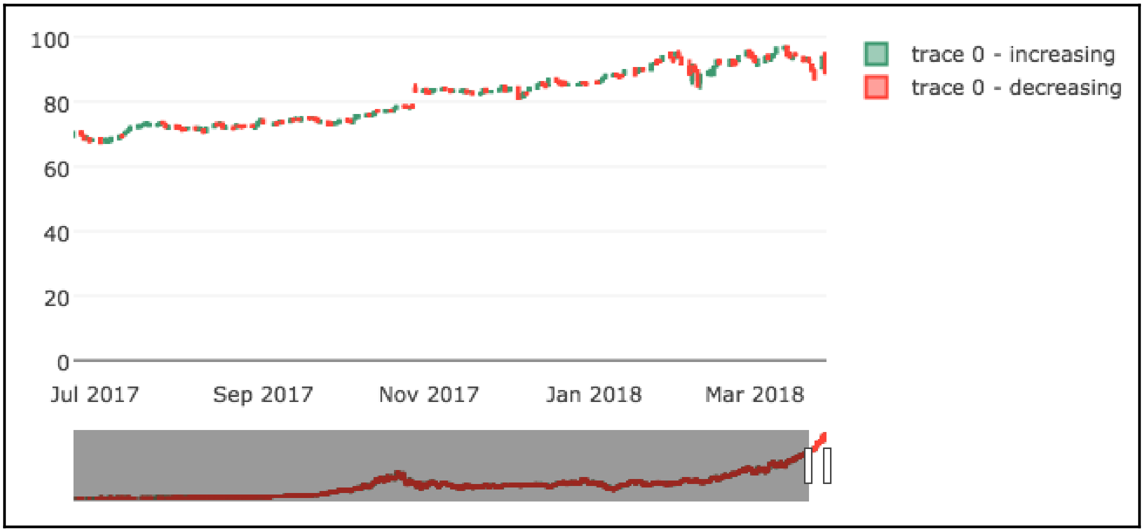

1000
800
600
400
200
0
-0.2
-0.1
0.0
0.1

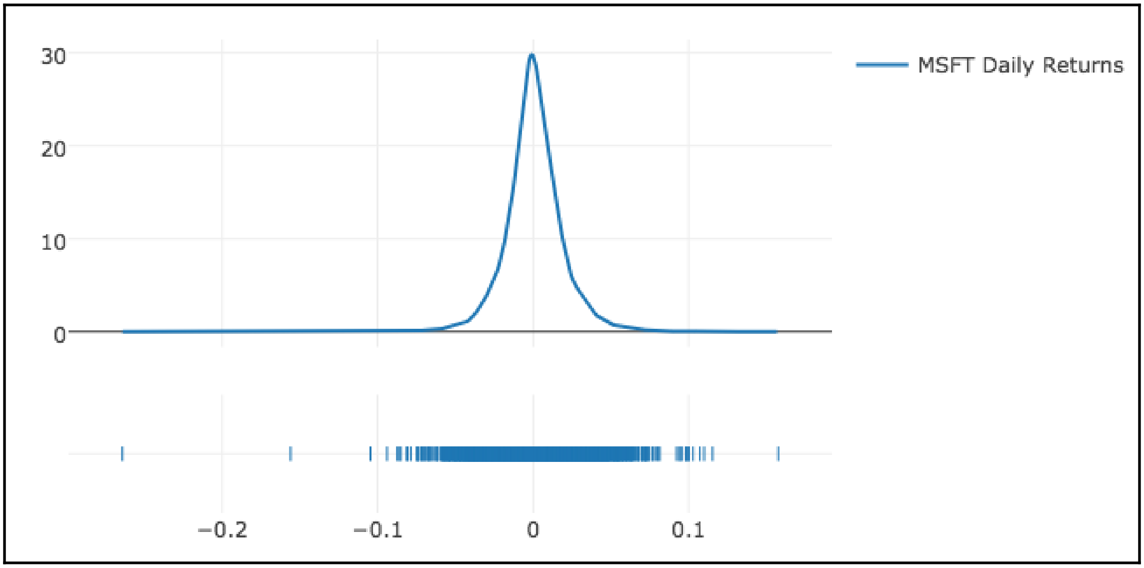

100
80
60
40
20
0
1990
1995
2000
2005
2010
2015


100
80
60
40
20
0
1990
2000
2010
trace 0 - increasing
trace 0 - decreasing

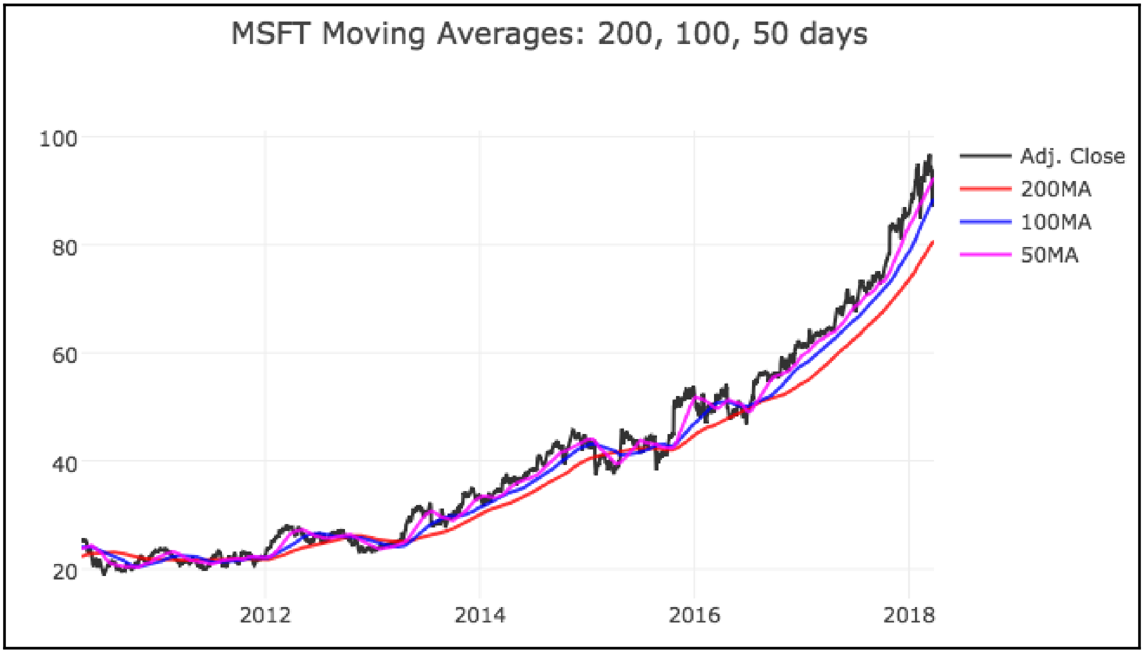

100
80
60
40
20
0
Feb 4
2018
Feb 11
Feb 18
trace 0 - increasing
trace 0 - decreasing


100
80
60
40
20
0
1990
2000
2010
trace 0 - increasing
trace 0 - decreasing

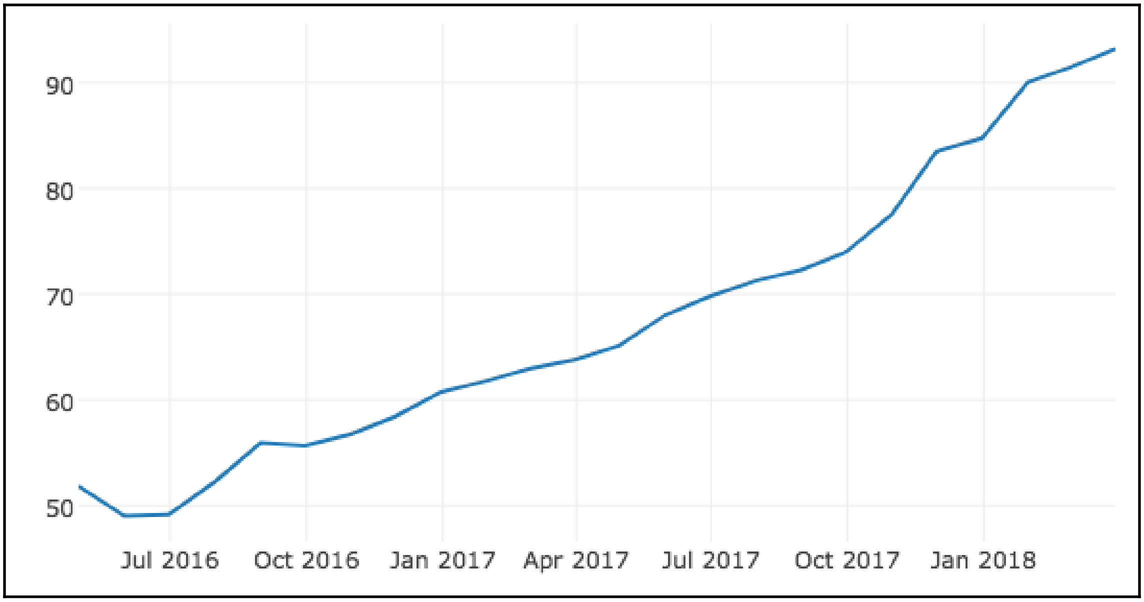

100
80
60
40
20
0
Jul 2017
Sep 2017
Nov 2017
Jan 2018
Mar 2018
trace 0 - increasing
trace 0 - decreasing

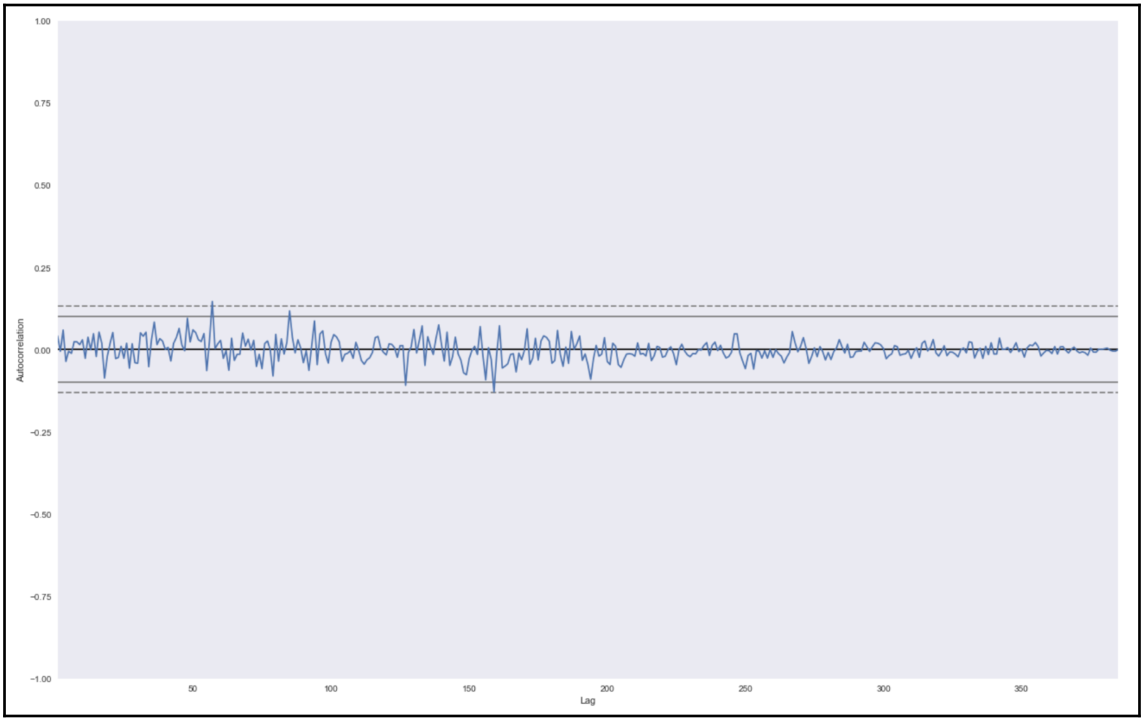


Date	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Low	Adj. Close	Adj. Volume	Daily Pct. Change	200MA	100MA	50MA
2018-03-14	95.120	95.410	93.50	93.85	31576898.0	0.0	1.0	95.120	95.410	93.50	93.85	31576898.0	-0.013352	79.764181	87.322623	91.4226
2018-03-15	93.530	94.580	92.83	94.18	26279014.0	0.0	1.0	93.530	94.580	92.83	94.18	26279014.0	0.006950	79.888837	87.492232	91.5872
2018-03-16	94.680	95.380	93.92	94.60	47329521.0	0.0	1.0	94.680	95.380	93.92	94.60	47329521.0	-0.000845	80.013416	87.663055	91.7522
2018-03-19	93.740	93.900	92.11	92.89	31752589.0	0.0	1.0	93.740	93.900	92.11	92.89	31752589.0	-0.009068	80.132266	87.807824	91.8678
2018-03-20	93.050	93.770	93.00	93.13	21787780.0	0.0	1.0	93.050	93.770	93.00	93.13	21787780.0	0.000860	80.251028	87.954794	91.9666
2018-03-21	92.930	94.050	92.21	92.48	23753263.0	0.0	1.0	92.930	94.050	92.21	92.48	23753263.0	-0.004842	80.358327	88.094965	92.0506
2018-03-22	91.265	91.750	89.66	89.79	37578166.0	0.0	1.0	91.265	91.750	89.66	89.79	37578166.0	-0.016162	80.449602	88.210525	92.0820
2018-03-23	89.500	90.460	87.08	87.18	42159397.0	0.0	1.0	89.500	90.460	87.08	87.18	42159397.0	-0.025922	80.526639	88.298691	92.0692
2018-03-26	90.610	94.000	90.40	93.78	55031149.0	0.0	1.0	90.610	94.000	90.40	93.78	55031149.0	0.034985	80.637320	88.402612	92.1832
2018-03-27	94.940	95.139	88.51	89.47	53704562.0	0.0	1.0	94.940	95.139	88.51	89.47	53704562.0	-0.057615	80.728653	88.462637	92.1810

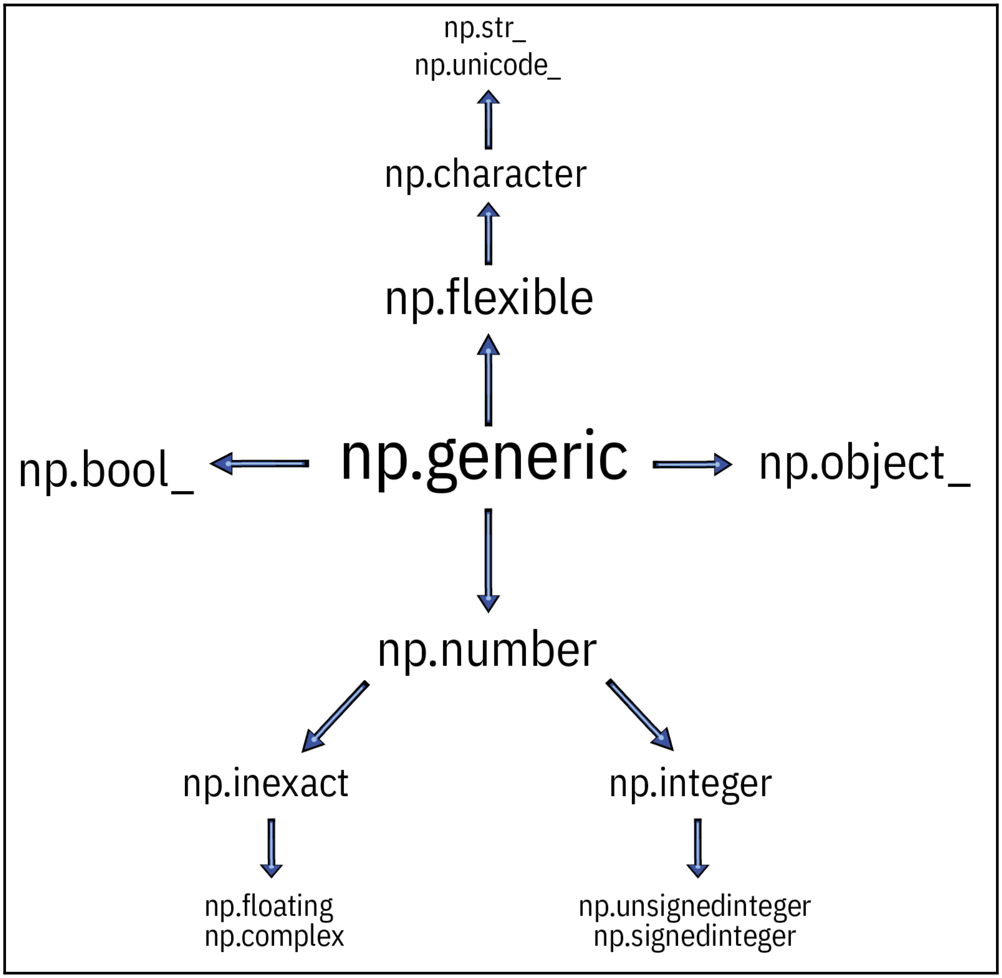


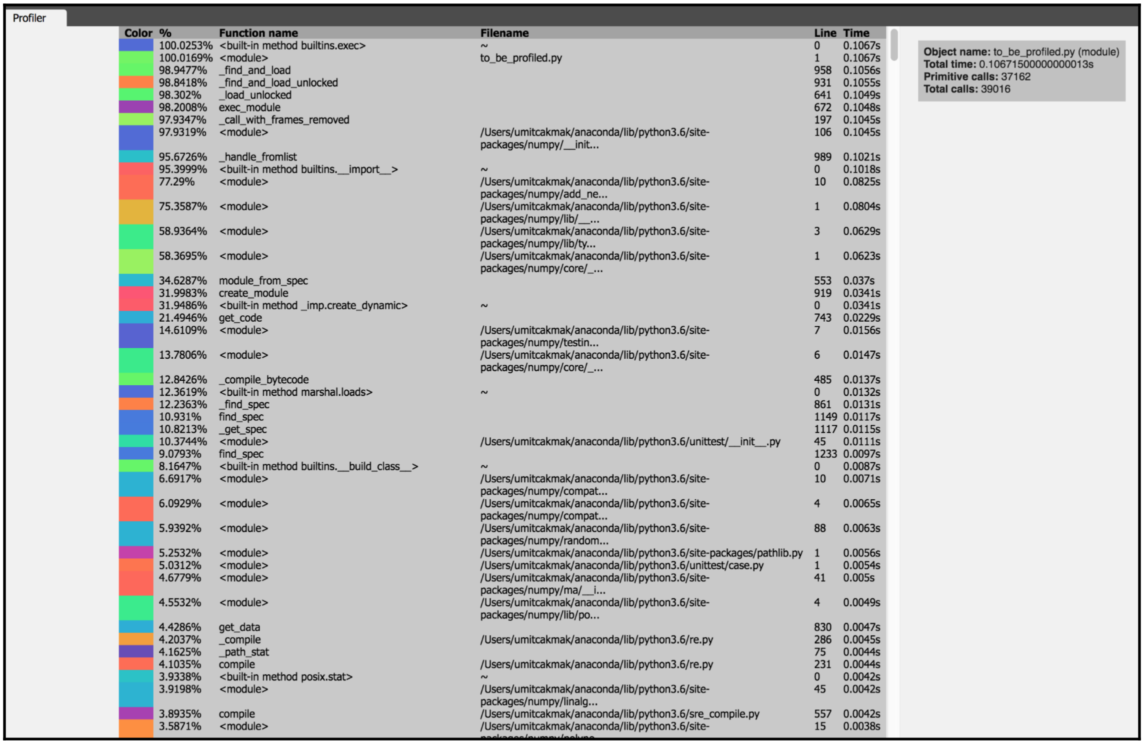
Date	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Low	Adj. Close	Adj. Volume	Daily Pct. Change	200MA	100MA	50MA
2017-06-30	70.561364	71.014600	69.835727	70.517955	2.773277e+07	0.000000	1.0	69.834054	70.282618	69.115897	69.791092	2.773277e+07	-0.000553	61.950990	65.477445	67.593782
2017-07-31	71.843250	72.412995	71.441000	72.012500	2.256239e+07	0.000000	1.0	71.102727	71.666599	70.704623	71.270232	2.256239e+07	0.002403	63.431438	66.971306	69.281458
2017-08-31	72.715652	73.196083	72.285187	72.816957	1.864639e+07	0.016957	1.0	72.183532	72.660475	71.756218	72.284124	1.864639e+07	0.001409	65.098648	68.785956	70.822208
2017-09-30	74.365500	74.786000	73.891000	74.344500	1.835672e+07	0.000000	1.0	73.990997	74.409380	73.518887	73.970103	1.835672e+07	-0.000265	66.700094	70.590506	72.419931
2017-10-31	77.889091	78.349318	77.529773	77.939545	2.002319e+07	0.000000	1.0	77.496844	77.954753	77.139335	77.547044	2.002319e+07	0.000765	68.223272	72.157040	73.922416
2017-11-30	83.620500	84.061610	83.124875	83.675500	1.980172e+07	0.021000	1.0	83.430357	83.870554	82.936030	83.485128	1.980172e+07	0.000679	70.262112	74.611141	77.520274
2017-12-31	84.836000	85.409915	84.163255	84.758500	2.237773e+07	0.000000	1.0	84.836000	85.409915	84.163255	84.758500	2.237773e+07	-0.000846	72.340131	77.291184	81.378112
2018-01-31	89.965952	90.657486	89.372143	90.074286	2.587511e+07	0.000000	1.0	89.965952	90.657486	89.372143	90.074286	2.587511e+07	0.001250	74.734287	80.352415	85.030938
2018-02-28	91.392105	92.764974	90.055832	91.413158	3.633093e+07	0.000000	1.0	91.392105	92.764974	90.055832	91.413158	3.633093e+07	0.000513	77.324649	83.868860	88.237253
2018-03-31	93.570263	94.471032	92.227684	93.169474	3.339462e+07	0.000000	1.0	93.570263	94.471032	92.227684	93.169474	3.339462e+07	-0.004179	79.714769	87.235899	91.248411


90
80
70
60
50
Jul 2016
Oct 2016
Jan 2017
Apr 2017
Jul 2017
Oct 2017
Jan 2018

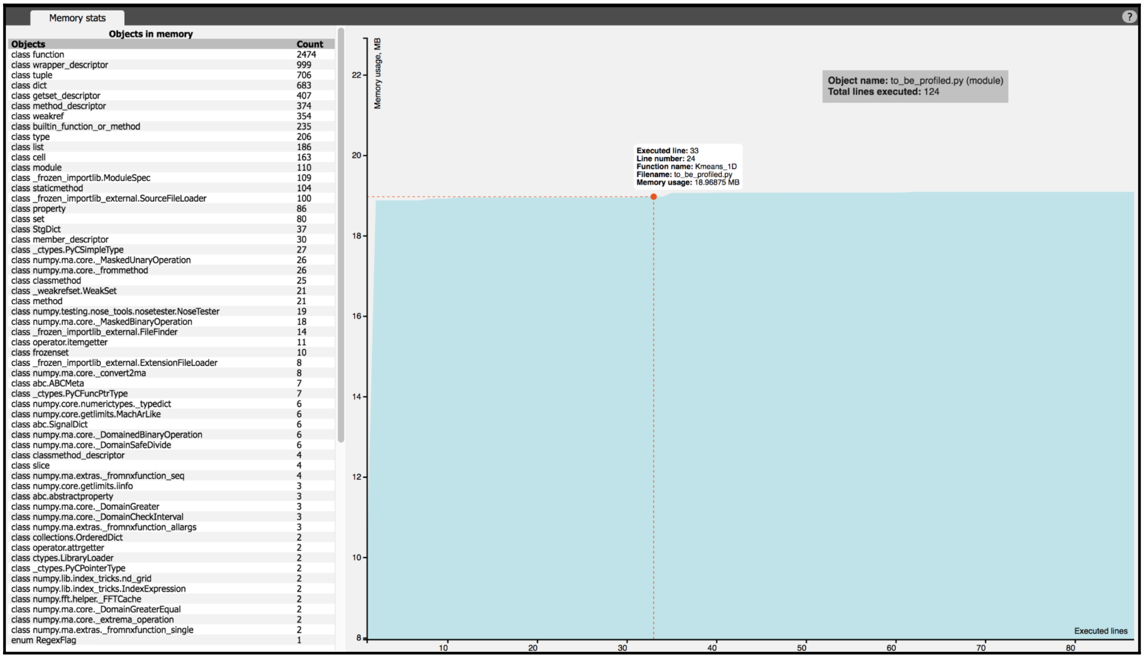

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
Autocorrelation
50
100
150
200
250
300
350
Lag

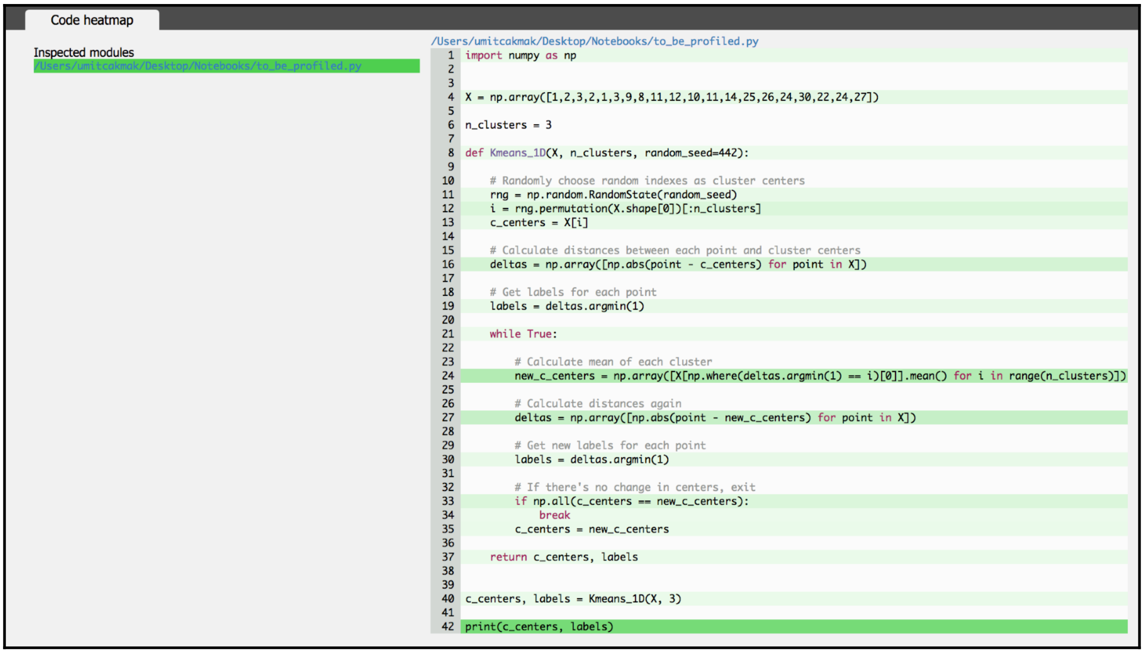
Chapter 7: Advanced Numpy

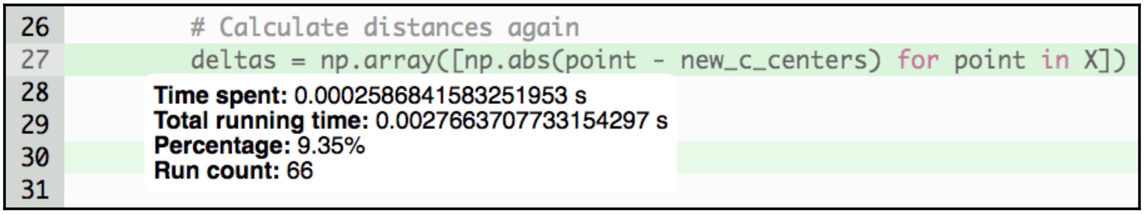




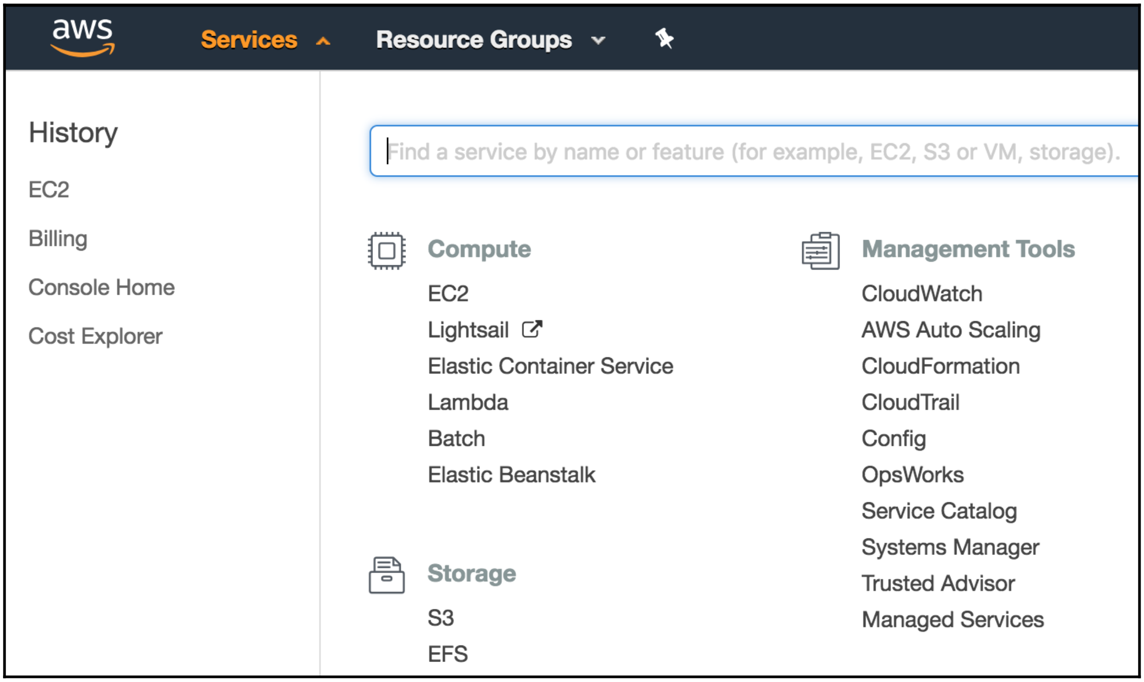
Color	%	Function name	Filename	Line	Time
	100.0253%	<built-in method builtins.exec>	~	0	0.1067s
	100.0169%	<module>	to_be_profiled.py	1	0.1067s
	98.9477%	_find_and_load		958	0.1056s
	98.8418%	_find_and_load_unlocked		931	0.1055s
	98.302%	_load_unlocked		641	0.1049s
	98.2008%	exec_module		672	0.1048s
	97.9347%	_call_with_frames_removed		197	0.1045s
	97.9319%	<module>	/Users/umitcakmak/anaconda/lib/python3.6/site-packages/numpy/__init...	106	0.1045s
	95.6726%	_handle_fromlist		989	0.1021s
	95.3999%	<built-in method builtins.__import__>	~	0	0.1018s
	77.29%	<module>	/Users/umitcakmak/anaconda/lib/python3.6/site-packages/numpy/add_ne...	10	0.0825s
	75.3587%	<module>	/Users/umitcakmak/anaconda/lib/python3.6/site-packages/numpy/lib/__...	1	0.0804s
	58.9364%	<module>	/Users/umitcakmak/anaconda/lib/python3.6/site-packages/numpy/lib/ty...	3	0.0629s
	58.3695%	<module>	/Users/umitcakmak/anaconda/lib/python3.6/site-packages/numpy/core/_...	1	0.0623s
	34.6287%	module_from_spec		553	0.037s
	31.9983%	create_module		919	0.0341s
	31.9486%	<built-in method _imp.create_dynamic>	~	0	0.0341s
	21.4946%	get_code		743	0.0229s
	14.6109%	<module>	/Users/umitcakmak/anaconda/lib/python3.6/site-packages/numpy/testin...	7	0.0156s
	13.7806%	<module>	/Users/umitcakmak/anaconda/lib/python3.6/site-packages/numpy/core/_...	6	0.0147s
	12.8426%	_compile_bytecode		485	0.0137s
	12.3619%	<built-in method marshal.loads>	~	0	0.0132s
	12.2363%	_find_spec		861	0.0131s
	10.931%	find_spec		1149	0.0117s
	10.8213%	_get_spec		1117	0.0115s
	10.3744%	<module>	/Users/umitcakmak/anaconda/lib/python3.6/unittest/__init__.py	45	0.0111s
	9.0793%	find_spec		1233	0.0097s
	8.1647%	<built-in method builtins.__build_class__>	~	0	0.0087s
	6.6917%	<module>	/Users/umitcakmak/anaconda/lib/python3.6/site-packages/numpy/compat...	10	0.0071s
	6.0929%	<module>	/Users/umitcakmak/anaconda/lib/python3.6/site-packages/numpy/compat...	4	0.0065s
	5.9392%	<module>	/Users/umitcakmak/anaconda/lib/python3.6/site-packages/numpy/random...	88	0.0063s
	5.2532%	<module>	/Users/umitcakmak/anaconda/lib/python3.6/site-packages/pathlib.py	1	0.0056s
	5.0312%	<module>	/Users/umitcakmak/anaconda/lib/python3.6/unittest/case.py	1	0.0054s
	4.6779%	<module>	/Users/umitcakmak/anaconda/lib/python3.6/site-packages/numpy/ma/__i...	41	0.005s
	4.5532%	<module>	/Users/umitcakmak/anaconda/lib/python3.6/site-packages/numpy/lib/po...	4	0.0049s
	4.4286%	get_data		830	0.0047s
	4.2037%	_compile	/Users/umitcakmak/anaconda/lib/python3.6/re.py	286	0.0045s
	4.1625%	_path_stat		75	0.0044s
	4.1035%	compile	/Users/umitcakmak/anaconda/lib/python3.6/re.py	231	0.0044s
	3.9338%	<built-in method posix.stat>	~	0	0.0042s
	3.9198%	<module>	/Users/umitcakmak/anaconda/lib/python3.6/site-packages/numpy/linalg...	45	0.0042s
	3.8935%	compile	/Users/umitcakmak/anaconda/lib/python3.6/sre_compile.py	557	0.0042s
	3.5871%	<module>	/Users/umitcakmak/anaconda/lib/python3.6/site-	15	0.0038s

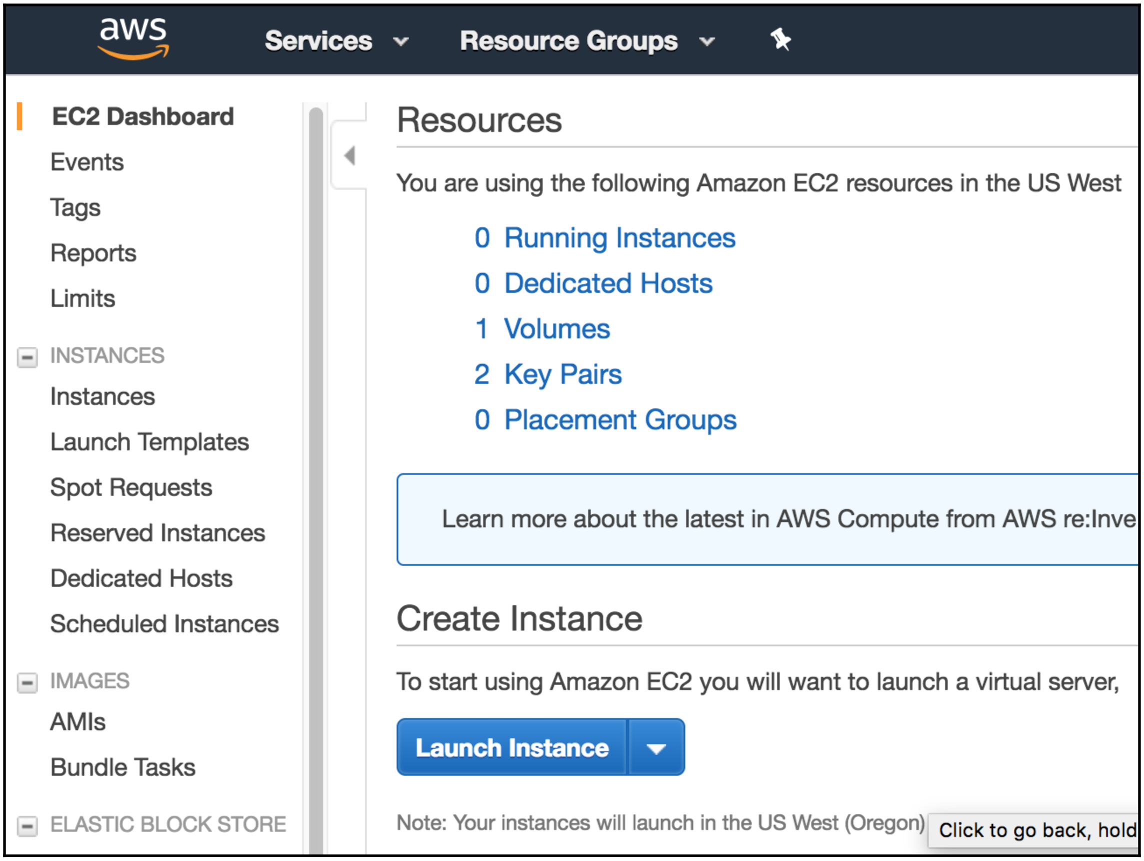

Memory stats
Objects in memory
Objects Count
class function 2474
class wrapper_descriptor 999
class tuple 706
class dict 683
class getset_descriptor 407
class method_descriptor 374
class weakref 354
class builtin_function_or_method 235
class type 206
class list 186
class cell 163
class module 110
class _frozen_importlib.ModuleSpec 109
class staticmethod 104
class _frozen_importlib_external.SourceFileLoader 100
class property 86
class set 80
class StgDict 37
class member_descriptor 30
class _ctypes.PyCSimpleType 27
class numpy.ma.core._MaskedUnaryOperation 26
class numpy.ma.core._frommethod 26
class classmethod 25
class _weakrefset.WeakSet 21
class method 21
class numpy.testing.nose_tools.nosetester.NoseTester 19
class numpy.ma.core._MaskedBinaryOperation 18
class _frozen_importlib_external.FileFinder 14
class operator.itemgetter 11
class frozenset 10
class _frozen_importlib_external.ExtensionFileLoader 8
class numpy.ma.core._convert2ma 8
class abc.ABCMeta 7
class _ctypes.PyCFuncPtrType 7
class numpy.core.numerictypes._typedict 6
class numpy.core.getlimits.MachArLike 6
class abc.SignalDict 6
class numpy.ma.core._DomainedBinaryOperation 6
class numpy.ma.core._DomainSafeDivide 6
class classmethod_descriptor 4
class slice 4
class numpy.ma.extras._fromnxfunction_seq 4
class numpy.core.getlimits.iinfo 3
class abc.abstractproperty 3
class numpy.ma.core._DomainGreater 3
class numpy.ma.core._DomainCheckInterval 3
class numpy.ma.extras._fromnxfunction_allargs 3
class collections.OrderedDict 2
class operator.attrgetter 2
class ctypes.LibraryLoader 2
class _ctypes.PyCPointerType 2
class numpy.lib.index_tricks.nd_grid 2
class numpy.lib.index_tricks.IndexExpression 2
class numpy.fft.helper._FFTCache 2
class numpy.ma.core._DomainGreaterEqual 2
class numpy.ma.core._extrema_operation 2
class numpy.ma.extras._fromnxfunction_single 2
enum RegexFlag 1
Memory usage, MB
Object name: to_be_profiled.py (module)
Total lines executed: 124
Executed line: 33
Line number: 24
Function name: Kmeans_1D
Filename: to_be_profiled.py
Memory usage: 18.96875 MB
Executed lines


Code heatmap
Inspected modules
/Users/umitcakmak/Desktop/Notebooks/to_be_profiled.py
/Users/umitcakmak/Desktop/Notebooks/to_be_profiled.py
import numpy as np
X = np.array([1,2,3,2,1,3,9,8,11,12,10,11,14,25,26,24,30,22,24,27])
n_clusters = 3
def Kmeans_1D(X, n_clusters, random_seed=442):
# Randomly choose random indexes as cluster centers
rng = np.random.RandomState(random_seed)
i = rng.permutation(X.shape[0])[:n_clusters]
c_centers = X[i]
# Calculate distances between each point and cluster centers
deltas = np.array([np.abs(point - c_centers) for point in X])
# Get labels for each point
labels = deltas.argmin(1)
while True:
# Calculate mean of each cluster
new_c_centers = np.array([X[np.where(deltas.argmin(1) == i)[0]].mean() for i in range(n_clusters)])
# Calculate distances again
deltas = np.array([np.abs(point - new_c_centers) for point in X])
# Get new labels for each point
labels = deltas.argmin(1)
# If there's no change in centers, exit
if np.all(c_centers == new_c_centers):
break
c_centers = new_c_centers
return c_centers, labels
c_centers, labels = Kmeans_1D(X, 3)
print(c_centers, labels)

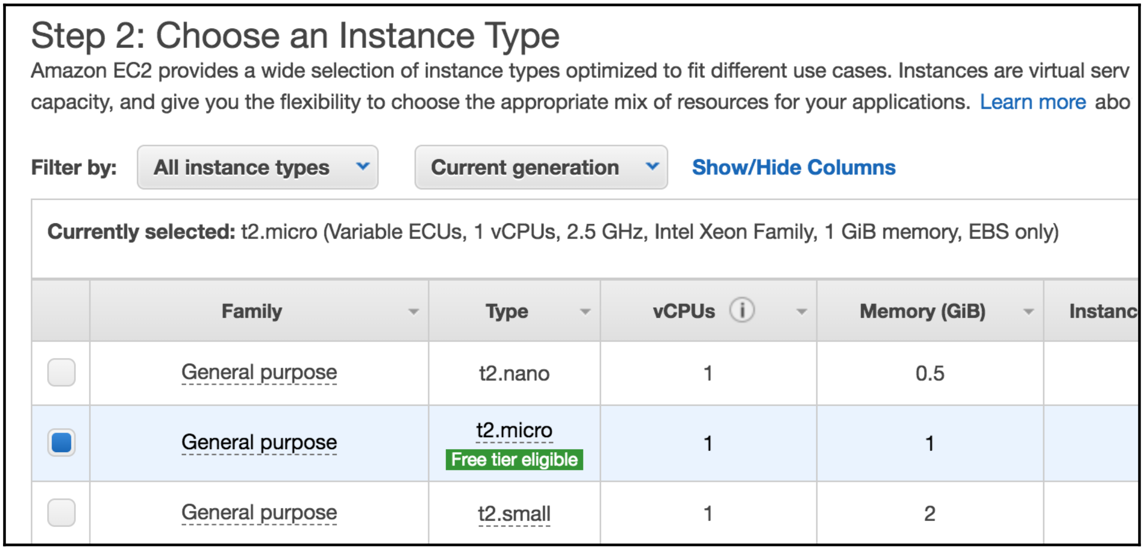

# Calculate distances again
deltas = np.array([np.abs(point - new_c_centers) for point in X])
Time spent: 0.0002586841583251953 s
Total running time: 0.0027663707733154297 s
Percentage: 9.35%
Run count: 66

Chapter 8: Overview of High-Performance Numerical Computing Libraries

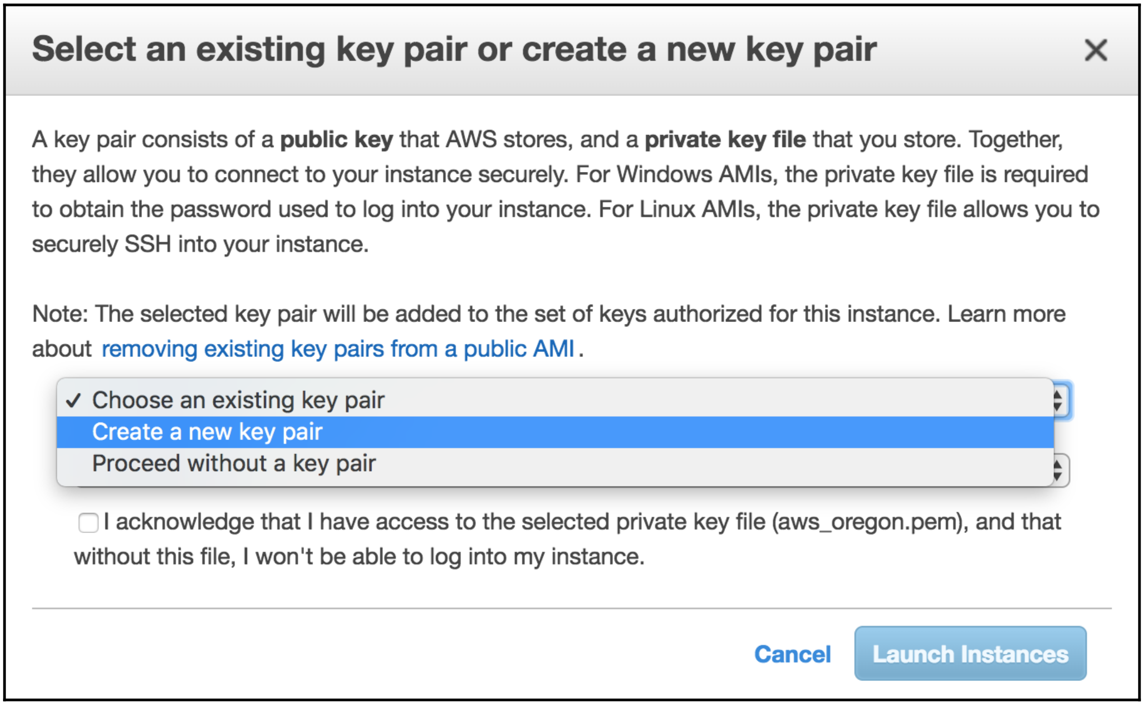


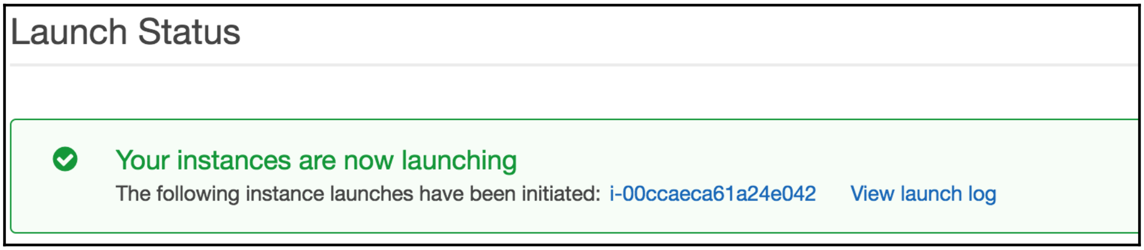

aws
Services
Resource Groups
EC2 Dashboard
Events
Tags
Reports
Limits
INSTANCES
Instances
Launch Templates
Spot Requests
Reserved Instances
Dedicated Hosts
Scheduled Instances
IMAGES
AMIs
Bundle Tasks
ELASTIC BLOCK STORE
Resources
You are using the following Amazon EC2 resources in the US West
0 Running Instances
0 Dedicated Hosts
1 Volumes
2 Key Pairs
0 Placement Groups
Learn more about the latest in AWS Compute from AWS re:Inve
Create Instance
To start using Amazon EC2 you will want to launch a virtual server,
Launch Instance
Note: Your instances will launch in the US West (Oregon)
Click to go back, hold

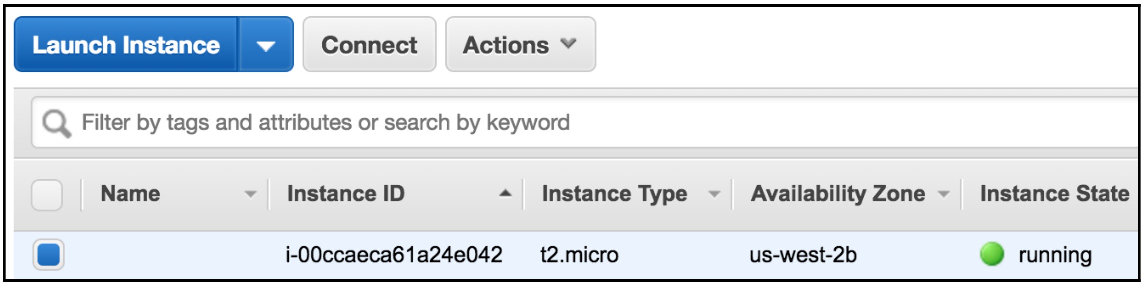

Ubuntu Server 16.04 LTS (HVM), SSD Volume Type - ami-db710fa3
Ubuntu Server 16.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).
Root device type: ebs Virtualization type: hvm ENA Enabled: Yes
Free tier eligible
Select
64-bit

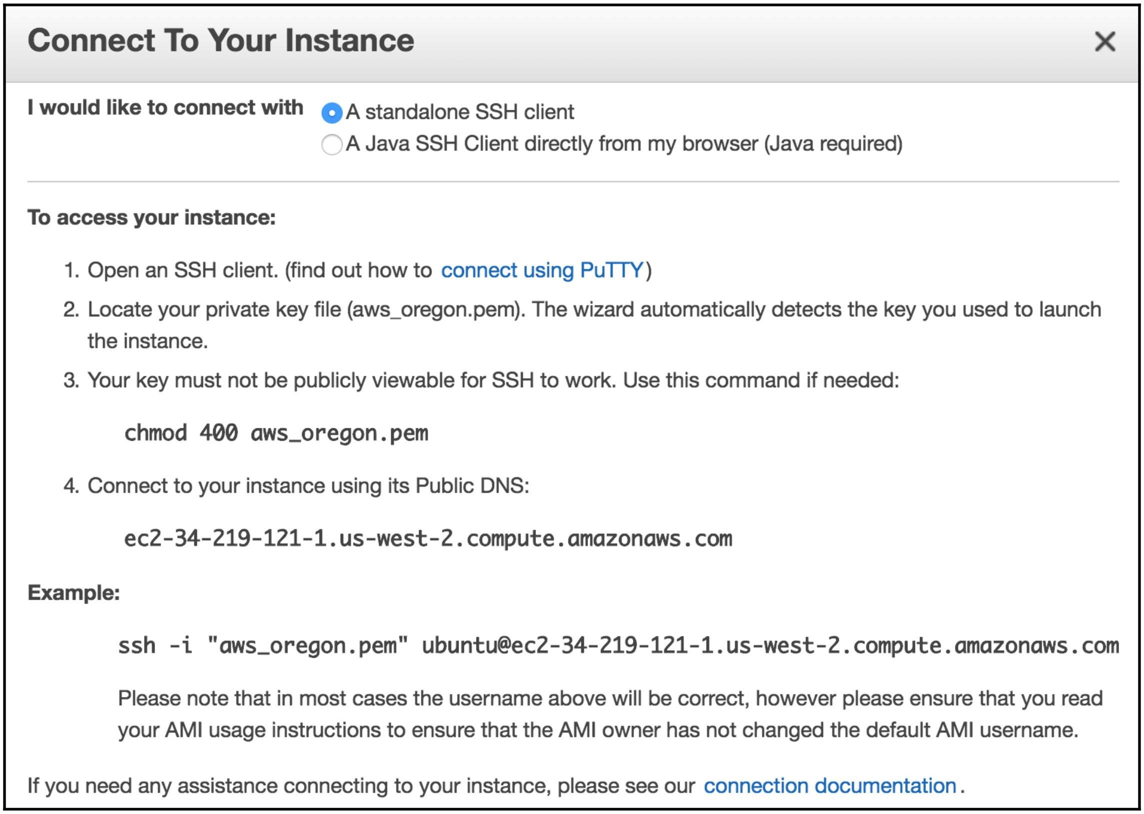

Step 2: Choose an Instance Type
Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual serv
capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more abo
Filter by:
All instance types
Current generation
Show/Hide Columns
Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)
Family
Type
vCPUs
Memory (GiB)
Instanc
General purpose
t2.nano
1
0.5
General purpose
t2.micro
Free tier eligible
1
1
General purpose
t2.small
1
2


Review and Launch


Select an existing key pair or create a new key pair
A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.
Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI .
✓ Choose an existing key pair
Create a new key pair
Proceed without a key pair
I acknowledge that I have access to the selected private key file (aws_oregon.pem), and that without this file, I won't be able to log into my instance.
Cancel
Launch Instances


Launch Status
Your instances are now launching
The following instance launches have been initiated: i-00ccaeca61a24e042
View launch log


Launch Instance
Connect
Actions
Filter by tags and attributes or search by keyword
Name
Instance ID
Instance Type
Availability Zone
Instance State
i-00ccaeca61a24e042
t2.micro
us-west-2b
running


Connect To Your Instance
I would like to connect with
A standalone SSH client
A Java SSH Client directly from my browser (Java required)
To access your instance:
1. Open an SSH client. (find out how to connect using PuTTY)
2. Locate your private key file (aws_oregon.pem). The wizard automatically detects the key you used to launch the instance.
3. Your key must not be publicly viewable for SSH to work. Use this command if needed:
chmod 400 aws_oregon.pem
4. Connect to your instance using its Public DNS:
ec2-34-219-121-1.us-west-2.compute.amazonaws.com
Example:
ssh -i "aws_oregon.pem" ubuntu@ec2-34-219-121-1.us-west-2.compute.amazonaws.com
Please note that in most cases the username above will be correct, however please ensure that you read your AMI usage instructions to ensure that the AMI owner has not changed the default AMI username.
If you need any assistance connecting to your instance, please see our connection documentation.

```
The authenticity of host 'ec2-34-219-121-1.us-west-2.compute.amazonaws.com (34.219.121.1)' can't be established.
ECDSA key fingerprint is SHA256:Mhxlf76E7CmSlNH52X4ls2EKeujAYYh4NETAfju9+cA.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-34-219-121-1.us-west-2.compute.amazonaws.com,34.219.121.1' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-1060-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-21-32:~$
```

```
ubuntu@ip-172-31-21-32:/$ sudo apt-get install python3-scipy
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  binutils cpp cpp-5 g++ g++-5 gcc gcc-5 libasan2 libatomic1 libblas-common libblas3 libc-dev-bin libc6-dev libcc1-0 libcilkrts5
  libgcc-5-dev libgfortran3 libgomp1 libisl15 libitm1 liblapack3 liblsan0 libmpc3 libmpx0 libquadmath0 libstdc++-5-dev libtsan0
  libubsan0 linux-libc-dev manpages-dev python3-decorator python3-numpy
Suggested packages:
  binutils-doc cpp-doc gcc-5-locales g++-multilib g++-5-multilib gcc-5-doc libstdc++6-5-dbg gcc-multilib make autoconf automake
  libtool flex bison gdb gcc-doc gcc-5-multilib libgcc1-dbg libgomp1-dbg libitm1-dbg libatomic1-dbg libasan2-dbg liblsan0-dbg
  libtsan0-dbg libubsan0-dbg libcilkrts5-dbg libmpx0-dbg libquadmath0-dbg glibc-doc libstdc++-5-doc gfortran python-numpy-doc
  python3-dev python3-nose python3-numpy-dbg python-scipy-doc
The following NEW packages will be installed:
  binutils cpp cpp-5 g++ g++-5 gcc gcc-5 libasan2 libatomic1 libblas-common libblas3 libc-dev-bin libc6-dev libcc1-0 libcilkrts5
  libgcc-5-dev libgfortran3 libgomp1 libisl15 libitm1 liblapack3 liblsan0 libmpc3 libmpx0 libquadmath0 libstdc++-5-dev libtsan0
  libubsan0 linux-libc-dev manpages-dev python3-decorator python3-numpy python3-scipy
0 upgraded, 33 newly installed, 0 to remove and 0 not upgraded.
Need to get 49.8 MB of archives.
After this operation, 190 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

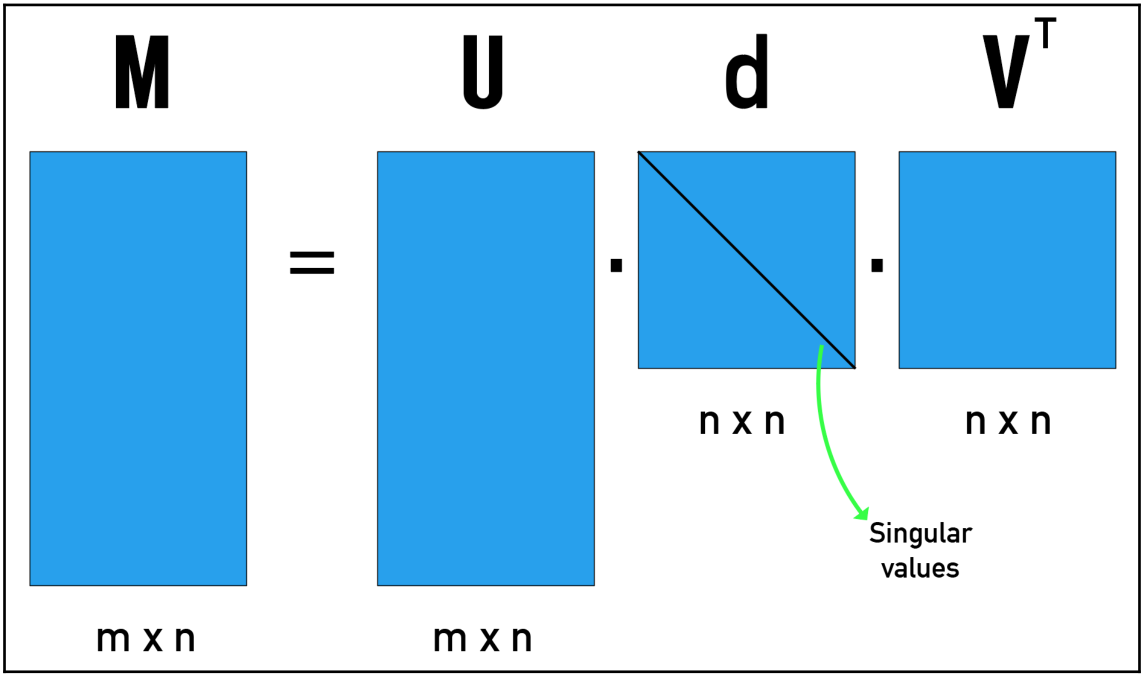
```
ubuntu@ip-172-31-21-32:~$ python3
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

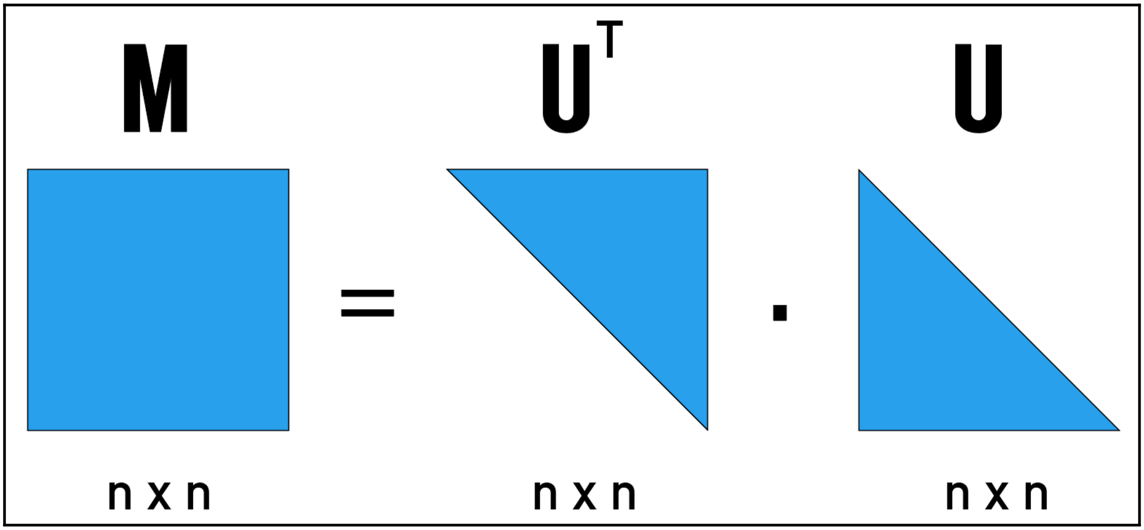
```
>>> import numpy as np
>>> np.show_config()
openblas_lapack_info:
  NOT AVAILABLE
atlas_3_10_blas_threads_info:
  NOT AVAILABLE
atlas_blas_info:
  NOT AVAILABLE
atlas_threads_info:
  NOT AVAILABLE
atlas_3_10_info:
  NOT AVAILABLE
lapack_info:
    language = f77
    libraries = ['lapack', 'lapack']
    library_dirs = ['/usr/lib']
atlas_blas_threads_info:
  NOT AVAILABLE
blas_info:
    language = c
    libraries = ['blas', 'blas']
    library_dirs = ['/usr/lib']
    define_macros = [('HAVE_CBLAS', None)]
lapack_opt_info:
    define_macros = [('NO_ATLAS_INFO', 1), ('HAVE_CBLAS', None)]
    libraries = ['lapack', 'lapack', 'blas', 'blas']
    library_dirs = ['/usr/lib']
    language = c
atlas_info:
  NOT AVAILABLE
openblas_info:
  NOT AVAILABLE
blas_opt_info:
    define_macros = [('NO_ATLAS_INFO', 1), ('HAVE_CBLAS', None)]
    libraries = ['blas', 'blas']
    library_dirs = ['/usr/lib']
    language = c
lapack_mkl_info:
  NOT AVAILABLE
atlas_3_10_blas_info:
  NOT AVAILABLE
mkl_info:
  NOT AVAILABLE
blas_mkl_info:
  NOT AVAILABLE
atlas_3_10_threads_info:
  NOT AVAILABLE
```

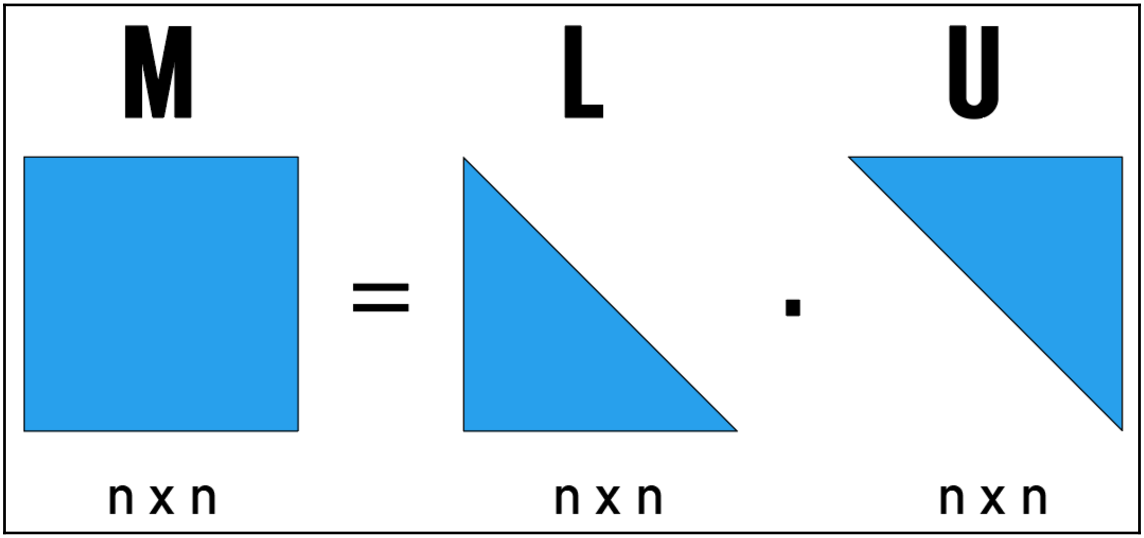
```
>>> import numpy as np
>>> np.show_config()
lapack_info:
  NOT AVAILABLE
openblas_lapack_info:
  NOT AVAILABLE
lapack_src_info:
  NOT AVAILABLE
lapack_opt_info:
  NOT AVAILABLE
atlas_3_10_threads_info:
  NOT AVAILABLE
blas_opt_info:
  NOT AVAILABLE
atlas_3_10_blas_info:
  NOT AVAILABLE
atlas_3_10_info:
  NOT AVAILABLE
atlas_info:
  NOT AVAILABLE
atlas_3_10_blas_threads_info:
  NOT AVAILABLE
blis_info:
  NOT AVAILABLE
blas_src_info:
  NOT AVAILABLE
openblas_clapack_info:
  NOT AVAILABLE
blas_mkl_info:
  NOT AVAILABLE
lapack_mkl_info:
  NOT AVAILABLE
blas_info:
  NOT AVAILABLE
atlas_threads_info:
  NOT AVAILABLE
openblas_info:
  NOT AVAILABLE
atlas_blas_threads_info:
  NOT AVAILABLE
accelerate_info:
  NOT AVAILABLE
atlas_blas_info:
  NOT AVAILABLE
```

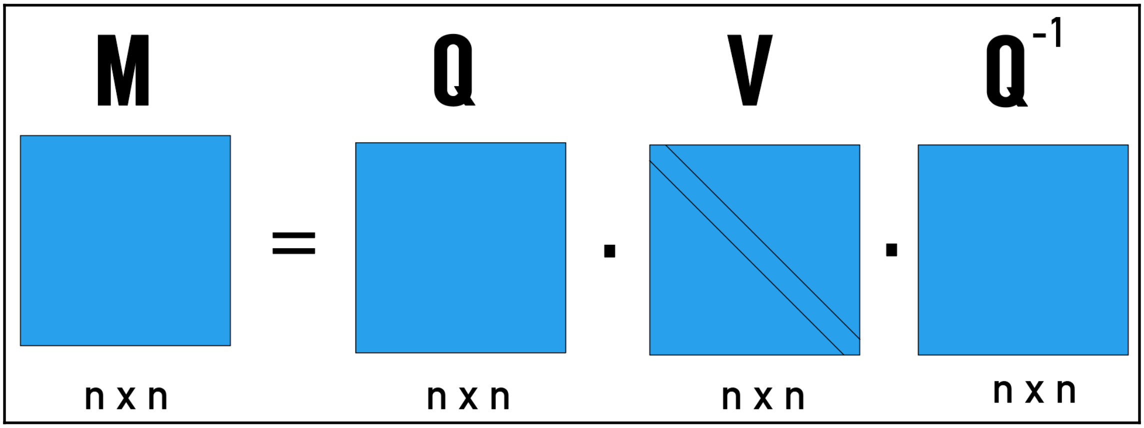
```
>>> import numpy as np
>>> np.show_config()
blas_mkl_info:
  NOT AVAILABLE
blis_info:
  NOT AVAILABLE
openblas_info:
  NOT AVAILABLE
atlas_3_10_blas_threads_info:
  NOT AVAILABLE
atlas_3_10_blas_info:
  NOT AVAILABLE
atlas_blas_threads_info:
  NOT AVAILABLE
atlas_blas_info:
  NOT AVAILABLE
blas_opt_info:
    extra_compile_args = ['-msse3', '-I/System/Library/Frameworks/vecLib.framework/Headers']
    extra_link_args = ['-Wl,-framework', '-Wl,Accelerate']
    define_macros = [('NO_ATLAS_INFO', 3), ('HAVE_CBLAS', None)]
lapack_mkl_info:
  NOT AVAILABLE
openblas_lapack_info:
  NOT AVAILABLE
openblas_clapack_info:
  NOT AVAILABLE
atlas_3_10_threads_info:
  NOT AVAILABLE
atlas_3_10_info:
  NOT AVAILABLE
atlas_threads_info:
  NOT AVAILABLE
atlas_info:
  NOT AVAILABLE
lapack_opt_info:
    extra_compile_args = ['-msse3']
    extra_link_args = ['-Wl,-framework', '-Wl,Accelerate']
    define_macros = [('NO_ATLAS_INFO', 3), ('HAVE_CBLAS', None)]
```

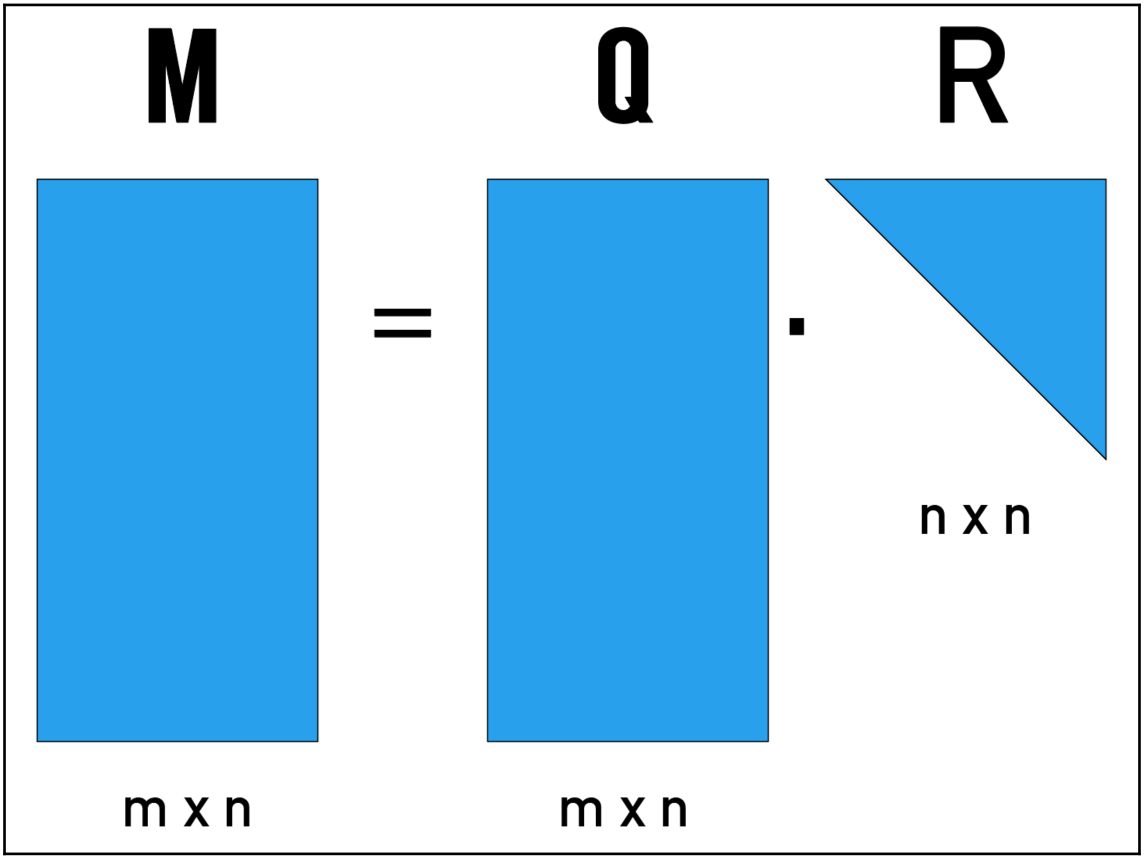
```
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> np.show_config()
blis_info:
  NOT AVAILABLE
lapack_mkl_info:
  NOT AVAILABLE
blas_mkl_info:
  NOT AVAILABLE
blas_opt_info:
    language = c
    libraries = ['openblas', 'openblas']
    define_macros = [('HAVE_CBLAS', None)]
    library_dirs = ['/usr/local/lib']
openblas_info:
    language = c
    libraries = ['openblas', 'openblas']
    define_macros = [('HAVE_CBLAS', None)]
    library_dirs = ['/usr/local/lib']
openblas_lapack_info:
    language = c
    libraries = ['openblas', 'openblas']
    define_macros = [('HAVE_CBLAS', None)]
    library_dirs = ['/usr/local/lib']
lapack_opt_info:
    language = c
    libraries = ['openblas', 'openblas']
    define_macros = [('HAVE_CBLAS', None)]
    library_dirs = ['/usr/local/lib']
```


M
=
U
.
d
.
V
T
m x n
m x n
n x n
n x n
Singular
values


M
=
U
T
.
U
n x n
n x n
n x n

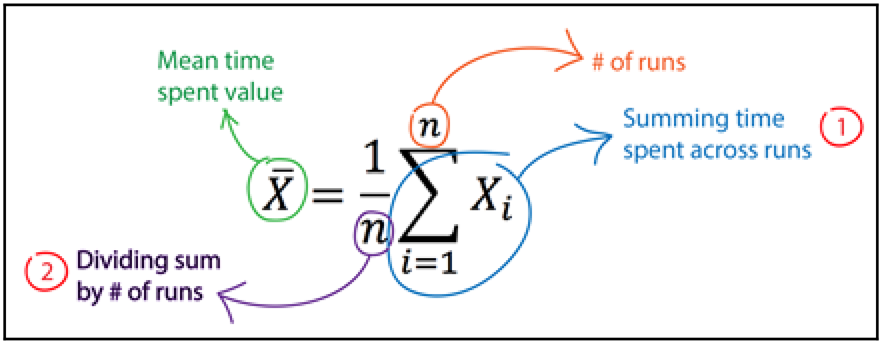

M
L
U
=
n x n
n x n
n x n


M
Q
V
Q-1
=
n x n
n x n
n x n
n x n


M
Q
R
=
·
n x n
m x n
m x n

Chapter 9: Performance Benchmarks

```
ubuntu@ip-172-31-21-32:~$ cd ~
ubuntu@ip-172-31-21-32:~$ mkdir py_scripts && cd py_scripts
```




```
ubuntu@ip-172-31-25-226:~/py_scripts$ vi linalg_benchmark.py
```

```
NumPy Configuration:
--------------------
""")
np.__config__.show()
:wq!
```

```
ubuntu@ip-172-31-25-226:~/py_scripts$ python3 linalg_benchmark.py
```

```
ubuntu@ip-172-31-22-134:~/py_scripts$ ~/anaconda3/bin/python linalg_benchmark.py
```

Operation	Mean	Std. Deviation
V-V Product	0.00000122	0.00000071
V-M Product	0.00000872	0.00000147
M-M Product	0.00074976	0.00001754
SV Decomp.	0.00644510	0.00009101
LU Decomp.	0.00042435	0.00001801
QR Decomp.	0.00134417	0.00003373
Cholesky D.	0.00001229	0.00000306
Eigval Dec.	0.01133923	0.00014564

Operation	Mean	Std. Deviation
V-V Product	0.00000169	0.00000104
V-M Product	0.00018053	0.00001345
M-M Product	0.09042594	0.00078627
SV Decomp.	1.72078687	2.11683465
LU Decomp.	0.36958391	0.05764444
QR Decomp.	1.64355660	0.26008436
Cholesky D.	0.00012395	0.00203646
Eigval Dec.	11.03387896	1.19246878

Operation	Mean	Std. Deviation
V-V Product	0.00000115	0.00000059
V-M Product	0.00000333	0.00000135
M-M Product	0.00009168	0.00000847
SV Decomp.	0.00507356	0.00005898
LU Decomp.	0.00016124	0.00001763
QR Decomp.	0.00065833	0.00001702
Cholesky D.	0.00001366	0.00000374
Eigval Dec.	0.03457905	0.00043139

Operation	Mean	Std. Deviation
V-V Product	0.00000124	0.00000078
V-M Product	0.00006752	0.00000487
M-M Product	0.00752822	0.00009364
SV Decomp.	0.13901888	0.00128025
LU Decomp.	0.00575469	0.00009780
QR Decomp.	0.02157722	0.00024035
Cholesky D.	0.00001288	0.00000212
Eigval Dec.	3.94406696	3.75736472

Operation	Mean	Std. Deviation
V-V Product	0.00000118	0.00000078
V-M Product	0.00000537	0.00001443
M-M Product	0.00029508	0.00011157
SV Decomp.	0.00475364	0.00025615
LU Decomp.	0.00015830	0.00000738
QR Decomp.	0.00093086	0.00004695
Cholesky D.	0.00001311	0.00000290
Eigval Dec.	0.01048062	0.00028431

Operation	Mean	Std. Deviation
V-V Product	0.00000168	0.00000054
V-M Product	0.00013248	0.00001036
M-M Product	0.02474427	0.00063530
SV Decomp.	0.22419701	0.00352764
LU Decomp.	0.00561713	0.00013463
QR Decomp.	0.05162554	0.00122877
Cholesky D.	0.00001262	0.00000260
Eigval Dec.	3.18629725	2.77181242

Operation	Mean	Std. Deviation
V-V Product	0.00000432	0.00031263
V-M Product	0.00000357	0.00005485
M-M Product	0.00007010	0.00035516
SV Decomp.	0.00241478	0.00065733
LU Decomp.	0.00015441	0.00008672
QR Decomp.	0.00055125	0.00030522
Cholesky D.	0.00001264	0.00003074
Eigval Dec.	0.00746131	0.00012120

Operation	Mean	Std. Deviation
V-V Product	0.00000140	0.00001808
V-M Product	0.00006262	0.00000957
M-M Product	0.00670626	0.00009224
SV Decomp.	0.09701678	0.00102559
LU Decomp.	0.00496843	0.00010792
QR Decomp.	0.01590121	0.00027027
Cholesky D.	0.00001278	0.00000220
Eigval Dec.	0.22408283	0.00155203

Index