W1-RY2G9P-B6_DataScience

March 24, 2019

```
In [47]: from IPython.display import Image
         Image('ok.png')
```

Out[47]:

FIRST ASSIGNMENT ON DATA SCIENCE		
Student's Code W1-RY2G9P-B6	AIMS African Institute for Mathematical Sciences CAMEROON	Deadline 03.19, 11:59pm
2018-2019		**2018-2019**
Lecturer:**Dr. Bubacarr Bah**		

1 Problem 1: Optimization

Typically, for many linear inverse problems the optimisation problem you want to solve is the following unconstrained optimisation problem:

$$\min_{\theta} J(\theta) := \frac{1}{2}||A\theta - y||_2^2$$

This optimization problem (2) is equivalent to the following Lagrangian formulation:

$$\min_{\theta} J(\theta) + \frac{\lambda}{2}||\theta||_2^2 \leq B \tag{1}$$

Show that the solution of the minimization problem 3 is given by:

$$\theta = (A^T A + \lambda I)^{-1} A^T y \tag{2}$$

where is the identity matrix.
ANSWER

$$\min_{\theta} J(\theta) := \frac{1}{2}||A\theta - y||_2^2 \tag{3}$$

We replace this expression inside equation (3) we get:

$$\begin{aligned}
L(\theta) =& \frac{1}{2}||A\theta - y||_2^2 + \frac{\lambda}{2}||\theta||_2^2 \\
=& \frac{1}{2}\left[(A\theta - y)^T(A\theta - y)\right] + \frac{1}{2}\lambda\theta\theta^T \\
=& \frac{1}{2}\left(A^T\theta^T A\theta - A^T\theta^T y - y^T A\theta + y^T y\right) + \frac{1}{2}\lambda\theta\theta^T \\
=& \frac{1}{2}\left(A^T\theta^T A\theta - 2A^T\theta^T y + y^T y\right) + \frac{1}{2}\lambda\theta\theta^T
\end{aligned}$$

Because,

$$yA^T\theta^T = y^T A\theta$$

Hence,

$$L(\theta) = \frac{1}{2}\left(A^T\theta^T A\theta - 2A^T\theta^T y + y^T y\right) + \frac{1}{2}\lambda\theta\theta^T$$

$$\begin{aligned}
\nabla_\theta L(\theta) =& \frac{1}{2}\left(2A^T\theta A - 2A^T y\right) + \frac{\lambda}{2} \times 2\theta \\
=& \frac{1}{2}\left[2\theta(A^T A + \lambda I) - 2A^T y\right]
\end{aligned}$$

When we set $\nabla_\theta L(\theta) = 0$ we have:

$$2\theta(A^T A + \lambda I) = 2A^T y$$

Hence,

$$\theta = (A^T A + \lambda I)^{-1} A^T y$$

\end{document}

2 Problem 2: Classification

Import the breast cancer dataset from scikit-learn and perform a classification on it using the KNeighborsClassifier (k-NN) for k = 1, 2, . . . , 10. For each value of k record the training and testing errors (using the in-built accuracy score of the k-NN). Then plot both sets of errors on one plot with the right legend. Based on this result what would you consider the optimal k for this problem?

```
In [1]: import pandas as pd
        import matplotlib
        import numpy as np
        import sklearn
        import matplotlib.pyplot as plt
```

```
In [2]: from sklearn.datasets import load_breast_cancer
        cancer = load_breast_cancer()
        print("cancer.keys(): \n{}".format(cancer.keys()))
```

```
cancer.keys():
dict_keys(['target_names', 'DESCR', 'feature_names', 'target', 'data'])
```

```
In [3]: X = cancer.data  #feature matrix
        y = cancer.target #response vector
```

```
In [4]: print(X.shape)
        print(y.shape)
```

```
(569, 30)
(569,)
```

```
In [5]: print("Feature names:\n{}".format(cancer.feature_names))
```

```
Feature names:
['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
```

```
In [6]: cancer.target_names
```

```
Out[6]: array(['malignant', 'benign'],
              dtype='<U9')
```

```
In [7]: df = pd.DataFrame(X, columns=cancer.feature_names)
        df.head()
```

Out[7]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	\
0	17.99	10.38	122.80	1001.0	0.11840	
1	20.57	17.77	132.90	1326.0	0.08474	
2	19.69	21.25	130.00	1203.0	0.10960	
3	11.42	20.38	77.58	386.1	0.14250	
4	20.29	14.34	135.10	1297.0	0.10030	

	mean compactness	mean concavity	mean concave points	mean symmetry	\
0	0.27760	0.3001	0.14710	0.2419	
1	0.07864	0.0869	0.07017	0.1812	
2	0.15990	0.1974	0.12790	0.2069	
3	0.28390	0.2414	0.10520	0.2597	
4	0.13280	0.1980	0.10430	0.1809	

```
   mean fractal dimension          ...            worst radius  \
0                 0.07871          ...                   25.38
1                 0.05667          ...                   24.99
2                 0.05999          ...                   23.57
3                 0.09744          ...                   14.91
4                 0.05883          ...                   22.54

   worst texture  worst perimeter  worst area  worst smoothness  \
0          17.33           184.60      2019.0            0.1622
1          23.41           158.80      1956.0            0.1238
2          25.53           152.50      1709.0            0.1444
3          26.50            98.87       567.7            0.2098
4          16.67           152.20      1575.0            0.1374

   worst compactness  worst concavity  worst concave points  worst symmetry
0             0.6656           0.7119                0.2654          0.4601
1             0.1866           0.2416                0.1860          0.2750
2             0.4245           0.4504                0.2430          0.3613
3             0.8663           0.6869                0.2575          0.6638
4             0.2050           0.4000                0.1625          0.2364

   worst fractal dimension
0                  0.11890
1                  0.08902
2                  0.08758
3                  0.17300
4                  0.07678

[5 rows x 30 columns]
```

2.1 Exploratory Analysis

```
In [8]: y
```

```
Out[8]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
               1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1,
               1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0,
               1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1,
               1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1,
               0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
               0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1,
               0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1,
               0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0,
               0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1,
               1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
```

```
1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0,
1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1,
1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1,
0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1,
1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1,
1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1])
```

In [9]:
```
df['y']=y
df.head()
```

Out[9]:
```
   mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
0        17.99         10.38          122.80     1001.0          0.11840
1        20.57         17.77          132.90     1326.0          0.08474
2        19.69         21.25          130.00     1203.0          0.10960
3        11.42         20.38           77.58      386.1          0.14250
4        20.29         14.34          135.10     1297.0          0.10030

   mean compactness  mean concavity  mean concave points  mean symmetry  \
0           0.27760          0.3001              0.14710         0.2419
1           0.07864          0.0869              0.07017         0.1812
2           0.15990          0.1974              0.12790         0.2069
3           0.28390          0.2414              0.10520         0.2597
4           0.13280          0.1980              0.10430         0.1809

   mean fractal dimension ...  worst texture  worst perimeter  worst area
0                 0.07871 ...          17.33           184.60      2019.0
1                 0.05667 ...          23.41           158.80      1956.0
2                 0.05999 ...          25.53           152.50      1709.0
3                 0.09744 ...          26.50            98.87       567.7
4                 0.05883 ...          16.67           152.20      1575.0

   worst smoothness  worst compactness  worst concavity  worst concave poin
0            0.1622             0.6656           0.7119                0.26
1            0.1238             0.1866           0.2416                0.18
2            0.1444             0.4245           0.4504                0.24
3            0.2098             0.8663           0.6869                0.25
4            0.1374             0.2050           0.4000                0.16

   worst symmetry  worst fractal dimension  y
0          0.4601                  0.11890  0
1          0.2750                  0.08902  0
2          0.3613                  0.08758  0
```

```
3          0.6638                  0.17300  0
4          0.2364                  0.07678  0

[5 rows x 31 columns]
```

In [10]: df.tail()

```
Out[10]:     mean radius  mean texture  mean perimeter  mean area  mean smoothness
        564        21.56         22.39          142.00     1479.0          0.11100
        565        20.13         28.25          131.20     1261.0          0.09780
        566        16.60         28.08          108.30      858.1          0.08455
        567        20.60         29.33          140.10     1265.0          0.11780
        568         7.76         24.54           47.92      181.0          0.05263

             mean compactness  mean concavity  mean concave points  mean symmetry
        564           0.11590         0.24390              0.13890         0.1726
        565           0.10340         0.14400              0.09791         0.1752
        566           0.10230         0.09251              0.05302         0.1590
        567           0.27700         0.35140              0.15200         0.2397
        568           0.04362         0.00000              0.00000         0.1587

             mean fractal dimension ...  worst texture  worst perimeter  worst are
        564                 0.05623 ...          26.40           166.10      2027.
        565                 0.05533 ...          38.25           155.00      1731.
        566                 0.05648 ...          34.12           126.70      1124.
        567                 0.07016 ...          39.42           184.60      1821.
        568                 0.05884 ...          30.37            59.16       268.

             worst smoothness  worst compactness  worst concavity  \
        564           0.14100            0.21130           0.4107
        565           0.11660            0.19220           0.3215
        566           0.11390            0.30940           0.3403
        567           0.16500            0.86810           0.9387
        568           0.08996            0.06444           0.0000

             worst concave points  worst symmetry  worst fractal dimension  y
        564                0.2216          0.2060                  0.07115  0
        565                0.1628          0.2572                  0.06637  0
        566                0.1418          0.2218                  0.07820  0
        567                0.2650          0.4087                  0.12400  0
        568                0.0000          0.2871                  0.07039  1

        [5 rows x 31 columns]
```

In [11]: df.sample(8)

```
Out[11]:     mean radius  mean texture  mean perimeter  mean area  mean smoothness
        106        11.64         18.33           75.17      412.5          0.11420
        130        12.19         13.29           79.08      455.8          0.10660
```

```
195       12.91         16.33           82.53      516.4            0.07941
335       17.06         21.00          111.80      918.6            0.11190
239       17.46         39.28          113.40      920.6            0.09812
438       13.85         19.60           88.68      592.6            0.08684
267       13.59         21.84           87.16      561.0            0.07956
352       25.73         17.46          174.20     2010.0            0.11490

     mean compactness  mean concavity  mean concave points  mean symmetry
106           0.10170         0.07070              0.03485         0.1801
130           0.09509         0.02855              0.02882         0.1880
195           0.05366         0.03873              0.02377         0.1829
335           0.10560         0.15080              0.09934         0.1727
239           0.12980         0.14170              0.08811         0.1809
438           0.06330         0.01342              0.02293         0.1555
267           0.08259         0.04072              0.02142         0.1635
352           0.23630         0.33680              0.19130         0.1956

     mean fractal dimension ...  worst texture  worst perimeter  worst are
106                 0.06520 ...          29.26            85.51       521.
130                 0.06471 ...          17.81            91.38       545.
195                 0.05667 ...          22.00            90.81       600.
335                 0.06071 ...          33.15           143.20      1362.
239                 0.05966 ...          44.87           141.20      1408.
438                 0.05673 ...          28.01           100.90       749.
267                 0.05859 ...          30.04            97.66       661.
352                 0.06121 ...          23.58           229.30      3234.

     worst smoothness  worst compactness  worst concavity  \
106            0.1688             0.2660          0.28730
130            0.1427             0.2585          0.09915
195            0.1097             0.1506          0.17640
335            0.1449             0.2053          0.39200
239            0.1365             0.3735          0.32410
438            0.1118             0.1141          0.04753
267            0.1005             0.1730          0.14530
352            0.1530             0.5937          0.64510

     worst concave points  worst symmetry  worst fractal dimension  y
106               0.12180          0.2806                  0.09097  1
130               0.08187          0.3469                  0.09241  1
195               0.08235          0.3024                  0.06949  1
335               0.18270          0.2623                  0.07599  0
239               0.20660          0.2853                  0.08496  0
438               0.05890          0.2513                  0.06911  1
267               0.06189          0.2446                  0.07024  1
352               0.27560          0.3690                  0.08815  0

[8 rows x 31 columns]
```

In [14]:
```
df.describe()
```

Out[14]:
```
       mean radius  mean texture  mean perimeter    mean area  \
count   569.000000    569.000000      569.000000   569.000000
mean     14.127292     19.289649       91.969033   654.889104
std       3.524049      4.301036       24.298981   351.914129
min       6.981000      9.710000       43.790000   143.500000
25%      11.700000     16.170000       75.170000   420.300000
50%      13.370000     18.840000       86.240000   551.100000
75%      15.780000     21.800000      104.100000   782.700000
max      28.110000     39.280000      188.500000  2501.000000

       mean smoothness  mean compactness  mean concavity  mean concave poi
count       569.000000        569.000000      569.000000            569.000
mean          0.096360          0.104341        0.088799              0.048
std           0.014064          0.052813        0.079720              0.038
min           0.052630          0.019380        0.000000              0.000
25%           0.086370          0.064920        0.029560              0.020
50%           0.095870          0.092630        0.061540              0.033
75%           0.105300          0.130400        0.130700              0.074
max           0.163400          0.345400        0.426800              0.201

       mean symmetry  mean fractal dimension     ...      worst texture  \
count     569.000000              569.000000     ...         569.000000
mean        0.181162                0.062798     ...          25.677223
std         0.027414                0.007060     ...           6.146258
min         0.106000                0.049960     ...          12.020000
25%         0.161900                0.057700     ...          21.080000
50%         0.179200                0.061540     ...          25.410000
75%         0.195700                0.066120     ...          29.720000
max         0.304000                0.097440     ...          49.540000

       worst perimeter   worst area  worst smoothness  worst compactness
count       569.000000   569.000000        569.000000         569.000000
mean        107.261213   880.583128          0.132369           0.254265
std          33.602542   569.356993          0.022832           0.157336
min          50.410000   185.200000          0.071170           0.027290
25%          84.110000   515.300000          0.116600           0.147200
50%          97.660000   686.500000          0.131300           0.211900
75%         125.400000  1084.000000          0.146000           0.339100
max         251.200000  4254.000000          0.222600           1.058000

       worst concavity  worst concave points  worst symmetry  \
count       569.000000            569.000000      569.000000
mean          0.272188              0.114606        0.290076
std           0.208624              0.065732        0.061867
min           0.000000              0.000000        0.156500
25%           0.114500              0.064930        0.250400
```

```
50%                0.226700                 0.099930          0.282200
75%                0.382900                 0.161400          0.317900
max                1.252000                 0.291000          0.663800

       worst fractal dimension            y
count               569.000000   569.000000
mean                  0.083946     0.627417
std                   0.018061     0.483918
min                   0.055040     0.000000
25%                   0.071460     0.000000
50%                   0.080040     1.000000
75%                   0.092080     1.000000
max                   0.207500     1.000000

[8 rows x 31 columns]
```

2.2 Visualization

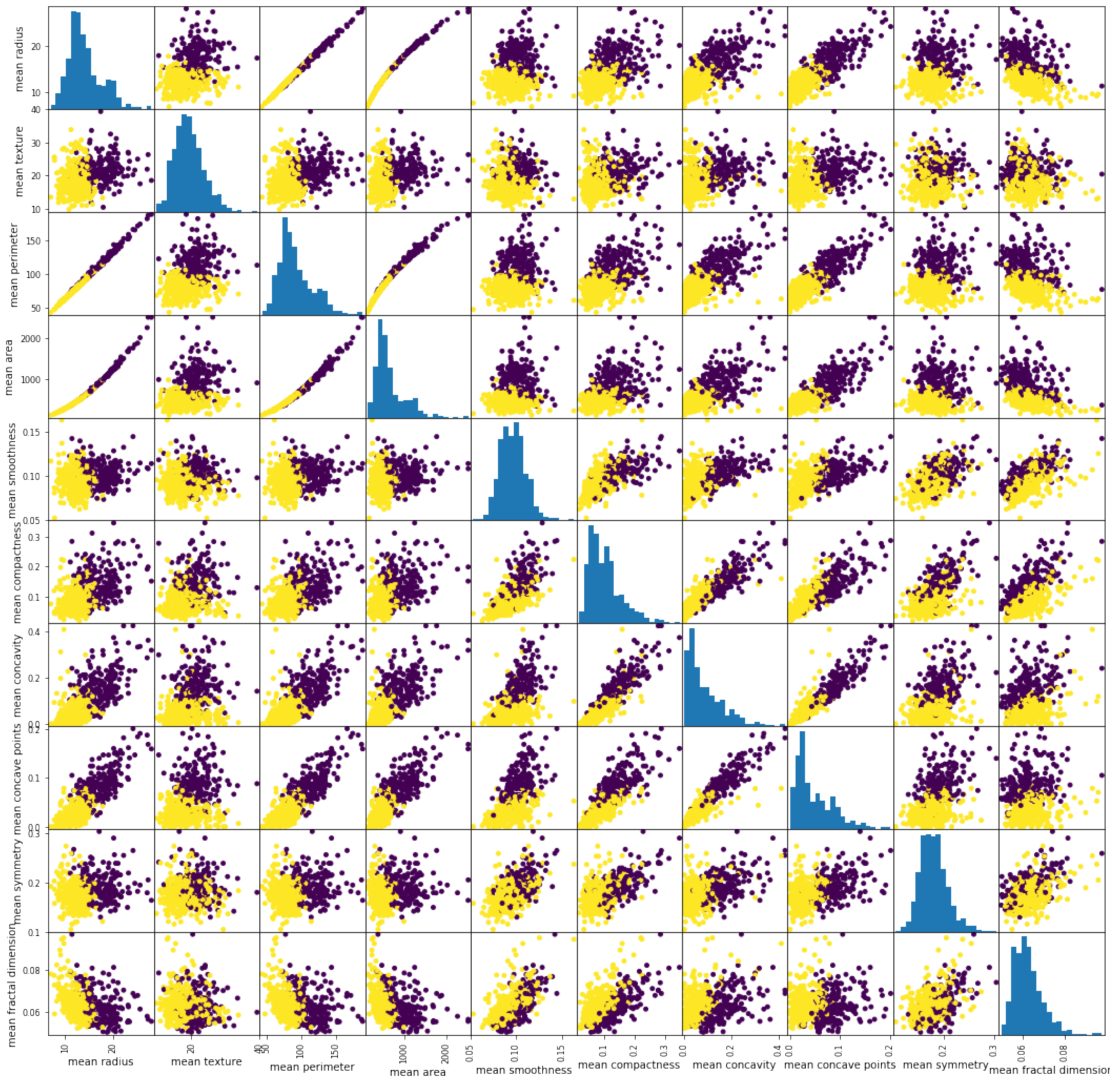
In [15]:
```
#visualisation
%pylab inline
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

pd.scatter_matrix(df[['mean radius', 'mean texture', 'mean perimeter',
'mean area','mean smoothness', 'mean compactness', 'mean concavity','mean
'mean fractal dimension']] , hist_kwds={'bins':20} , c = df['y'] , s = 60
```

```
Populating the interactive namespace from numpy and matplotlib
```

Out[15]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f622176a2e8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6221468f98>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f62214339b0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6221403780>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f62213c7ac8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f622139c780>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f62212e7ac8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f62212c4400>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6221290d68>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f622123ccc0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f62211aa908>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f62211b99b0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6221147a58>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f622110acc0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6221061ba8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f622102bcc0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6221009748>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6220fd2f60>,
```

```
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f6220f81eb8>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f6220eee898>],
[<matplotlib.axes._subplots.AxesSubplot object at 0x7f6220efc518>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f6220e89b38>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f622125e5f8>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f6220e1b4e0>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f6220ddd828>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f6220d34630>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f6220cff828>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f6220cdc2b0>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f6220c259b0>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f6220c557b8>],
[<matplotlib.axes._subplots.AxesSubplot object at 0x7f6220bc1470>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f6220bcd5c0>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f6220ade5c0>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f6220aa3908>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f6220a78710>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f6220a44908>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f62209a1390>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f622096ca90>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f6220998898>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f6220907550>],
[<matplotlib.axes._subplots.AxesSubplot object at 0x7f62209136a0>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f62208226a0>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f6220c67940>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f62207b6c18>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f622077d438>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f6220757e80>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f62206a45c0>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f62206d13c8>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f6220641080>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f622064a1d0>],
[<matplotlib.axes._subplots.AxesSubplot object at 0x7f62205db1d0>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f6220520518>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f62204f6320>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f62204c0518>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f622049af60>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f62203e86a0>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f62204144a8>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f6220385160>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f62203902b0>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f622029e2b0>],
[<matplotlib.axes._subplots.AxesSubplot object at 0x7f62202665f8>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f622023c400>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f62201f3198>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f62201ce278>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f6220193518>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f62200eb550>,
```

```
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f62200bc908>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f622006a5f8>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f62200583c8>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f621ffe3518>],
[<matplotlib.axes._subplots.AxesSubplot object at 0x7f621ff71518>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f621ff39860>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f621ff0e668>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f621fed9860>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f621fe362e8>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f621fe009e8>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f621fdad7f0>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f621fd9b4a8>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f621fd255f8>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f621fcb75f8>],
[<matplotlib.axes._subplots.AxesSubplot object at 0x7f621fc7d940>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f621fc53748>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f621fc1c940>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f621fb7a3c8>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f621fb43ac8>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f621faf08d0>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f621facf9e8>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f621fa9eb70>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f621f9e6898>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f621f9acb00>],
[<matplotlib.axes._subplots.AxesSubplot object at 0x7f621f9808d0>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f621f94cb00>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f621f8ab588>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f621f874c88>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f621f822a90>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f621f810748>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f621f81c898>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f621f72b898>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f621f6f2be0>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f621f6c79e8>]]
```



2.3 Model, Trainning and Testing

In [16]:
```python
#importing the model
from sklearn.neighbors import KNeighborsClassifier
```

In [17]:
```python
#instantiating the model
knn= KNeighborsClassifier( n_neighbors=1)
```

In [18]:
```python
#fitting the model with the data
knn.fit(X, y)
```

Out[18]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=1, p=2,
           weights='uniform')
```

2.3.1 creation of a training set and a testing set

```
In [20]: from sklearn.model_selection import train_test_split
         knn = KNeighborsClassifier(n_neighbors=1)

In [21]: X_train, X_test, y_train, y_test = train_test_split(X, y,random_state=42,

In [114]: print(X_train.shape)
          print(X_test.shape)

(426, 30)
(143, 30)
```

2.3.2 we instantiate a k-NN class and fit with our training set.

```
In [22]: knn.fit(X_train,y_train)

Out[22]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                    metric_params=None, n_jobs=1, n_neighbors=1, p=2,
                    weights='uniform')

In [32]: y_pred  = knn.predict(X_test)
         y_pred

Out[32]: array([1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0
                1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1
                1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1
                1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1
                1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1
                0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1
                1, 0, 1, 0, 1])
```

2.3.3 we make a prediction based on the test set

```
In [23]: print("prediction(Test set): {}".format(knn.predict(X_test)))
         # We calculate the predictions for y_test with the clf model

prediction(Test set): [1 0 0 1 1 0 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 1 1 1 1 1 1 0
 1 0 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 1 1 1 0 0 1 1 1 0 0 1 1 0 0 1 1
 1 1 1 1 1 1 0 1 1 0 0 0 0 0 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1 1 0
 1 0 0 1 0 1 1 1 0 1 1 1 0 1 0 0 1 1 0 0 0 0 1 0 0 1 1 1 0 1 0 1]
```

2.3.4 we evaluate the accuracy of the model by comparing the predictions with “correct answers”.

```
In [24]: print("test accuracy: {:.2f}".format(knn.score(X_test, y_test)))

test accuracy: 0.93
```

```
In [25]: print("training accuracy: {:.2f}".format(knn.score(X_train, y_train)))
```

```
training accuracy: 1.00
```

2.3.5 The model has an test accuracy of 93% and 100% for training accuracy with n_neighbors=1.

```
In [36]: import numpy as np
         print("fraction of correct examples")
         print(np.sum(y_pred == y_test) / float(len(y_test)))
```

```
fraction of correct examples
0.93006993007
```

```
In [37]: from sklearn import metrics
         metrics.accuracy_score(y_test , y_pred)
```

```
Out[37]: 0.93006993006993011
```

```
In [38]: #allow us to have the point that belong to predict and actual
         metrics.confusion_matrix(y_test , y_pred)
```

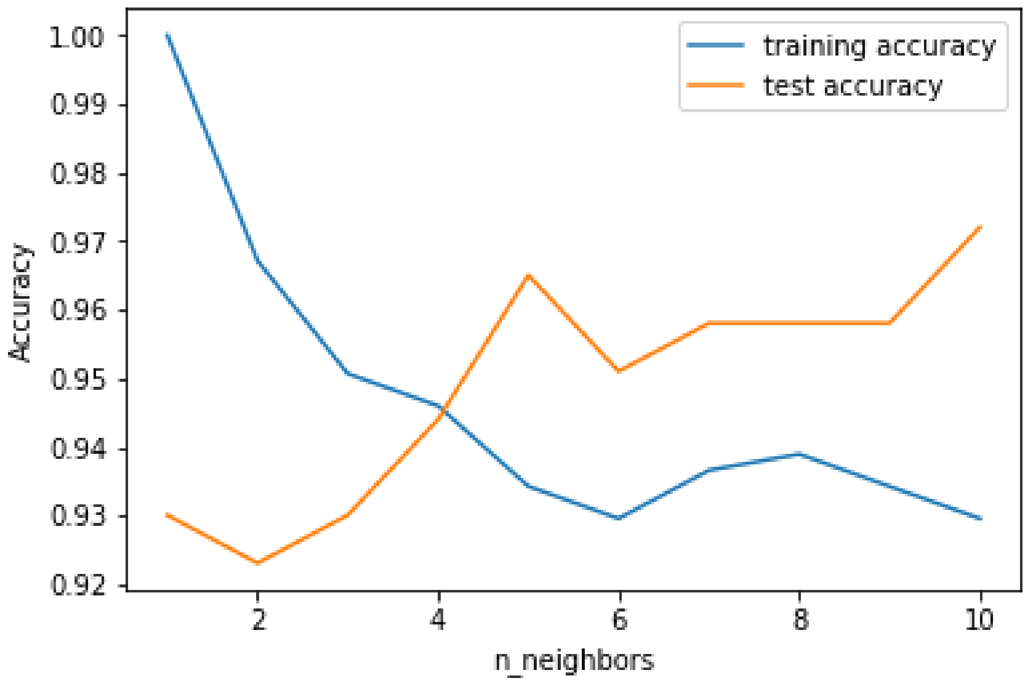
```
Out[38]: array([[48,  6],
                [ 4, 85]])
```

```
In [44]: %pylab inline
         from sklearn.datasets import load_breast_cancer

         cancer = load_breast_cancer()
         X_train, X_test, y_train, y_test = train_test_split(
             cancer.data, cancer.target,
             random_state=42, test_size =0.25)
         # Create training and testing datasets

         training_accuracy = []
         test_accuracy = []
         # try n_neighbors from 1 to 10 (k=1,2..10)
         neighbors_settings = range(1, 11)

         for n_neighbors in neighbors_settings:
             # import the model and instantiate
             knn = KNeighborsClassifier(n_neighbors=n_neighbors)
             knn.fit(X_train, y_train)
             # record training accuracy
             training_accuracy.append(knn.score(X_train, y_train))
             # record generalization accuracy
             test_accuracy.append(knn.score(X_test, y_test))
```

```
plt.plot(neighbors_settings,
         training_accuracy, label="training accuracy")
plt.plot(neighbors_settings,
        test_accuracy, label="test accuracy")
plt.ylabel("Accuracy")
plt.xlabel("n_neighbors")
plt.legend()
```

```
Populating the interactive namespace from numpy and matplotlib
```

Out[44]: <matplotlib.legend.Legend at 0x7f62210b4ef0>



In [45]:
```
training_accuracy #trainng of neighbors from 1 to 10
```

Out[45]:
```
[1.0,
 0.9671361502347417 5,
 0.9507042253521126 3,
 0.946009389671361 5,
 0.9342723004694836 2,
 0.9295774647887323 8,
 0.9366197183098591 3,
 0.9389671361502347 5,
 0.9342723004694836 2,
 0.9295774647887323 8]
```

```
In [46]: test_accuracy #test of neighbors from 1 to 10

Out[46]: [0.9300699300699301,
          0.9230769230769231,
          0.9300699300699301,
          0.9440559440559441,
          0.965034965034965,
          0.951048951048951,
          0.958041958041958,
          0.958041958041958,
          0.958041958041958,
          0.972027972027972]
```

2.4 Conclusion

the test set is best around k=5 the pic.The two curves intersect at k=4. Another thing to be noted is that since kNN models is the most complex when k=1, the trends of the two lines are flipped compared to standard complexity-accuracy chart for models.

```
In [ ]:
```