from sklearn.datasets import load\_breast\_cancer  
cancer = load\_breast\_cancer()  
print("cancer.keys(): \n{}".format(cancer.keys()))  
print(cancer.data)  
print(cancer.feature\_names)

cancer.keys():   
dict\_keys(['target', 'target\_names', 'feature\_names', 'DESCR', 'data'])  
[[ 1.79900000e+01 1.03800000e+01 1.22800000e+02 ..., 2.65400000e-01  
 4.60100000e-01 1.18900000e-01]  
 [ 2.05700000e+01 1.77700000e+01 1.32900000e+02 ..., 1.86000000e-01  
 2.75000000e-01 8.90200000e-02]  
 [ 1.96900000e+01 2.12500000e+01 1.30000000e+02 ..., 2.43000000e-01  
 3.61300000e-01 8.75800000e-02]  
 ...,   
 [ 1.66000000e+01 2.80800000e+01 1.08300000e+02 ..., 1.41800000e-01  
 2.21800000e-01 7.82000000e-02]  
 [ 2.06000000e+01 2.93300000e+01 1.40100000e+02 ..., 2.65000000e-01  
 4.08700000e-01 1.24000000e-01]  
 [ 7.76000000e+00 2.45400000e+01 4.79200000e+01 ..., 0.00000000e+00  
 2.87100000e-01 7.03900000e-02]]  
['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']

cancer.data[1]

array([ 2.05700000e+01, 1.77700000e+01, 1.32900000e+02,  
 1.32600000e+03, 8.47400000e-02, 7.86400000e-02,  
 8.69000000e-02, 7.01700000e-02, 1.81200000e-01,  
 5.66700000e-02, 5.43500000e-01, 7.33900000e-01,  
 3.39800000e+00, 7.40800000e+01, 5.22500000e-03,  
 1.30800000e-02, 1.86000000e-02, 1.34000000e-02,  
 1.38900000e-02, 3.53200000e-03, 2.49900000e+01,  
 2.34100000e+01, 1.58800000e+02, 1.95600000e+03,  
 1.23800000e-01, 1.86600000e-01, 2.41600000e-01,  
 1.86000000e-01, 2.75000000e-01, 8.90200000e-02])

cancer.target\_names

array(['malignant', 'benign'],   
 dtype='<U9')

cancer.data.shape

(569, 30)

cancer.target.shape

(569,)

x = cancer.data  
y = cancer.target

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n\_neighbors=1)

knn.fit(x, y)

KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski',  
 metric\_params=None, n\_jobs=1, n\_neighbors=1, p=2,  
 weights='uniform')

x\_new = cancer.data[8], cancer.data[9]

knn.predict(x\_new)

array([0, 0])

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n\_neighbors=5)

knn.fit(x, y)

KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski',  
 metric\_params=None, n\_jobs=1, n\_neighbors=5, p=2,  
 weights='uniform')

y\_pred = knn.predict(x)

from sklearn import metrics  
print(metrics.accuracy\_score(y, y\_pred))

0.947275922671

from sklearn.neighbors import KNeighborsClassifier  
knn = KNeighborsClassifier(n\_neighbors=1)  
knn.fit(x, y)  
y\_pred = knn.predict(x)  
print(metrics.accuracy\_score(y, y\_pred))

1.0

from sklearn.linear\_model import LogisticRegression  
logreg = LogisticRegression()  
logreg.fit(x, y)  
logreg.predict(x\_new)  
y\_pred = logreg.predict(x)

from sklearn import metrics  
print(metrics.accuracy\_score(y, y\_pred))

0.959578207381

from sklearn.cross\_validation import train\_test\_split  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.4, random\_state=4)

C:\Users\aluno.unilasalle\Downloads\WinPython-64bit-3.5.3.1Qt5\python-3.5.3.amd64\lib\site-packages\sklearn\cross\_validation.py:44: DeprecationWarning: 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 module will be removed in 0.20.  
 "This module will be removed in 0.20.", DeprecationWarning)

logreg = LogisticRegression()  
logreg.fit(X\_train, y\_train)  
y\_pred = logreg.predict(X\_test)

print(metrics.accuracy\_score(y\_test, y\_pred))

0.907894736842

knn = KNeighborsClassifier(n\_neighbors=5)  
knn.fit(X\_train, y\_train)  
y\_pred = knn.predict(X\_test)  
print(metrics.accuracy\_score(y\_test, y\_pred))

0.907894736842

knn = KNeighborsClassifier(n\_neighbors=1)  
knn.fit(X\_train, y\_train)  
y\_pred = knn.predict(X\_test)  
print(metrics.accuracy\_score(y\_test, y\_pred))

0.90350877193

k\_range = list(range(1, 26))  
scores = []  
for k in k\_range:  
 knn = KNeighborsClassifier(n\_neighbors=k)  
 knn.fit(X\_train, y\_train)  
 y\_pred = knn.predict(X\_test)  
 scores.append(metrics.accuracy\_score(y\_test, y\_pred))  
  
import matplotlib.pyplot as plt  
  
%matplotlib inline  
  
plt.plot(k\_range, scores)  
plt.xlabel('Valor de K para o KNN')  
plt.ylabel('Testando Acurácia')

<matplotlib.text.Text at 0xb548748>

