Applied Machine Learning, Module 1: A simple classification task

Import required modules and load data file

In [1]:

%matplotlib notebook

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.model\_selection import train\_test\_split

fruits = pd.read\_table('readonly/fruit\_data\_with\_colors.txt')

In [2]:

fruits.head()

Out[2]:

fruit\_label fruit\_name fruit\_subtype mass width height color\_score

0 1 apple granny\_smith 192 8.4 7.3 0.55

1 1 apple granny\_smith 180 8.0 6.8 0.59

2 1 apple granny\_smith 176 7.4 7.2 0.60

3 2 mandarin mandarin 86 6.2 4.7 0.80

4 2 mandarin mandarin 84 6.0 4.6 0.79

In [3]:

# create a mapping from fruit label value to fruit name to make results easier to interpret

lookup\_fruit\_name = dict(zip(fruits.fruit\_label.unique(), fruits.fruit\_name.unique()))

lookup\_fruit\_name

Out[3]:

{1: 'apple', 2: 'mandarin', 3: 'orange', 4: 'lemon'}

The file contains the mass, height, and width of a selection of oranges, lemons and apples. The heights were measured along the core of the fruit. The widths were the widest width perpendicular to the height.

Examining the data

In [4]:

# plotting a scatter matrix

from matplotlib import cm

X = fruits[['height', 'width', 'mass', 'color\_score']]

y = fruits['fruit\_label']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=0)

cmap = cm.get\_cmap('gnuplot')

scatter = pd.scatter\_matrix(X\_train, c= y\_train, marker = 'o', s=40, hist\_kwds={'bins':15}, figsize=(9,9), cmap=cmap)

<IPython.core.display.Javascript object>

In [5]:

# plotting a 3D scatter plot

from mpl\_toolkits.mplot3d import Axes3D

fig = plt.figure()

ax = fig.add\_subplot(111, projection = '3d')

ax.scatter(X\_train['width'], X\_train['height'], X\_train['color\_score'], c = y\_train, marker = 'o', s=100)

ax.set\_xlabel('width')

ax.set\_ylabel('height')

ax.set\_zlabel('color\_score')

plt.show()

<IPython.core.display.Javascript object>

Create train-test split

In [6]:

# For this example, we use the mass, width, and height features of each fruit instance

X = fruits[['mass', 'width', 'height']]

y = fruits['fruit\_label']

# default is 75% / 25% train-test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=0)

Create classifier object

In [7]:

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n\_neighbors = 5)

Train the classifier (fit the estimator) using the training data

In [8]:

knn.fit(X\_train, y\_train)

Out[8]:

KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski',

metric\_params=None, n\_jobs=1, n\_neighbors=5, p=2,

weights='uniform')

Estimate the accuracy of the classifier on future data, using the test data

In [9]:

knn.score(X\_test, y\_test)

Out[9]:

0.53333333333333333

Use the trained k-NN classifier model to classify new, previously unseen objects

In [10]:

# first example: a small fruit with mass 20g, width 4.3 cm, height 5.5 cm

fruit\_prediction = knn.predict([[20, 4.3, 5.5]])

lookup\_fruit\_name[fruit\_prediction[0]]

Out[10]:

'mandarin'

In [11]:

# second example: a larger, elongated fruit with mass 100g, width 6.3 cm, height 8.5 cm

fruit\_prediction = knn.predict([[100, 6.3, 8.5]])

lookup\_fruit\_name[fruit\_prediction[0]]

Out[11]:

'lemon'

Plot the decision boundaries of the k-NN classifier

In [12]:

from adspy\_shared\_utilities import plot\_fruit\_knn

plot\_fruit\_knn(X\_train, y\_train, 5, 'uniform') # we choose 5 nearest neighbors

<IPython.core.display.Javascript object>

How sensitive is k-NN classification accuracy to the choice of the 'k' parameter?

In [13]:

k\_range = range(1,20)

scores = []

for k in k\_range:

knn = KNeighborsClassifier(n\_neighbors = k)

knn.fit(X\_train, y\_train)

scores.append(knn.score(X\_test, y\_test))

plt.figure()

plt.xlabel('k')

plt.ylabel('accuracy')

plt.scatter(k\_range, scores)

plt.xticks([0,5,10,15,20]);

<IPython.core.display.Javascript object>

How sensitive is k-NN classification accuracy to the train/test split proportion?

In [14]:

t = [0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2]

knn = KNeighborsClassifier(n\_neighbors = 5)

plt.figure()

for s in t:

scores = []

for i in range(1,1000):

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 1-s)

knn.fit(X\_train, y\_train)

scores.append(knn.score(X\_test, y\_test))

plt.plot(s, np.mean(scores), 'bo')

plt.xlabel('Training set proportion (%)')

plt.ylabel('accuracy');

<IPython.core.display.Javascript object>