Module-1

July 3, 2020

You are currently looking at ***version 1.0*** *of this notebook. To download notebooks and datafiles, as well as get help on Jupyter notebooks in the Coursera platform, visit the Jupyter Notebook FAQ course resource.*

0.1 Applied Machine Learning, Module 1: A simple classification task

0.1.1 Import required modules and load data file

[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')
```

[2]:
```
fruits.head()
```

[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

[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
```

[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.

0.1.2 Examining the data

[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>
```

```
<IPython.core.display.HTML object>
```

[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>
```

```
<IPython.core.display.HTML object>
```

0.1.3 Create train-test split

[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)
```

0.1.4 Create classifier object

[7]:
```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors = 5)
```

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

[8]:
```
knn.fit(X_train, y_train)
```

[8]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform')
```

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

[9]:
```
knn.score(X_test, y_test)
```

[9]:
```
0.5333333333333333
```

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

[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]]
```

[10]:
```
'mandarin'
```

[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]]
```

[11]:
```
'lemon'
```

0.1.8 Plot the decision boundaries of the k-NN classifier

[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>
```

```
<IPython.core.display.HTML object>
```

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

[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>
```

```
<IPython.core.display.HTML object>
```

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

[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>
```

```
<IPython.core.display.HTML object>
```