Module 1

January 31, 2019

---

*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

```
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 eas
        lookup_fruit_name = dict(zip(fruits.fruit_label.unique(), fruits.fruit_name
        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.

0.1.2 Examining the data

```
In [ ]: # 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_k

In [ ]: # 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 =
        ax.set_xlabel('width')
        ax.set_ylabel('height')
        ax.set_zlabel('color_score')
        plt.show()
```

0.1.3 Create train-test split

```
In [9]: # For this example, we use the mass, width, and height features of each fru
        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

```
In [28]: from sklearn.neighbors import KNeighborsClassifier

         knn = KNeighborsClassifier(n_neighbors = 5)

In [29]: from sklearn.neighbors import KNeighborsClassifier
         knn=KNeighborsClassifier(n_neighbors =5)

In [30]: knn.fit(X_train,y_train)

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

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

```
In [31]: knn.score(X_test,y_test)

Out[31]: 0.53333333333333333

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

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

```
In [22]: knn.score(X_test, y_test)

Out[22]: 0.53333333333333333

In [32]: fruit_prediction=knn.predict([[20,5.5,6.5]])
         lookup_fruit_name[fruit_prediction[0]]


        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-32-7a2d3c8f2ef8> in <module>()
          1 fruit_prediction=knn.predict([[20,5.5,6.5]])
    ----> 2 lookup_fruit_name[fruit_prediction[0]]


        NameError: name 'lookup_fruit_name' is not defined
```

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

```
In [33]: # 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]]


        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-33-f5a9b6377eb6> in <module>()
          1 # first example: a small fruit with mass 20g, width 4.3 cm, height 5.5
          2 fruit_prediction = knn.predict([[20, 4.3, 5.5]])
    ----> 3 lookup_fruit_name[fruit_prediction[0]]


        NameError: name 'lookup_fruit_name' is not defined
```

```
In [27]: # second example: a larger, elongated fruit with mass 100g, width 6.3 cm,
         fruit_prediction = knn.predict([[100, 6.3, 8.5]])
         lookup_fruit_name[fruit_prediction[0]]
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-27-9f516b6e2570> in <module>()
  1 # second example: a larger, elongated fruit with mass 100g, width 6.3 c
  2 fruit_prediction = knn.predict([[100, 6.3, 8.5]])
----> 3 lookup_fruit_name[fruit_prediction[0]]


NameError: name 'lookup_fruit_name' is not defined
```

```
In [ ]:
```

0.1.8 Plot the decision boundaries of the k-NN classifier

```
In [ ]: from adspy_shared_utilities import plot_fruit_knn

        plot_fruit_knn(X_train, y_train, 5, 'uniform')   # we choose 5 nearest neig
```

```
In [1]: from adspy_shared_utilities import plot_fruit_knn
        plot_fruit_knn(X_train,y_train,10,'uniform')
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-1-17e1a76bf3ea> in <module>()
  1 from adspy_shared_utilities import plot_fruit_knn
----> 2 plot_fruit_knn(X_train,y_train,10,'uniform')


NameError: name 'X_train' is not defined
```

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

```
In [ ]: 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]);
```

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

```
In [ ]: 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
                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');
```

```
In [30]: %Arpit style

ERROR:root:Line magic function `%Arpit` not found.
```

```
In [2]: import matplotlib as plt
        import pandas as pd
        import numpy as np
        from sklearn.model_selection import train_test_split
```

```
In [7]: fruits=pd.read_table('fruit_data_with_colors.txt')
```

```
In [12]: fruits.head()
```

Out[12]:

	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 [10]: fruits.shape

Out[10]: (59, 7)

In [15]: a=dict(zip(fruits.fruit_label.unique(),fruits.fruit_name.unique()))

In [16]: a

Out[16]: {1: 'apple', 2: 'mandarin', 3: 'orange', 4: 'lemon'}

In [24]: x=fruits[['mass','width','height','color_score']]
         y=fruits['fruit_label']
         x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=0)

In [27]: x_train.head()

Out[27]:     mass  width  height  color_score
         42   154    7.2     7.2         0.82
         48   174    7.3    10.1         0.72
         7     76    5.8     4.0         0.81
         14   152    7.6     7.3         0.69
         32   164    7.2     7.0         0.80

In [28]: x_test.head()

Out[28]:     mass  width  height  color_score
         26   362    9.6     9.2         0.74
         35   150    7.1     7.9         0.75
         43   194    7.2    10.3         0.70
         28   140    6.7     7.1         0.72
         11   172    7.1     7.6         0.92

In [ ]:
```