17.1) Implementing the Classifier

Vitor Kamada

December 2019

Reference

Tables, Graphics, and Figures from

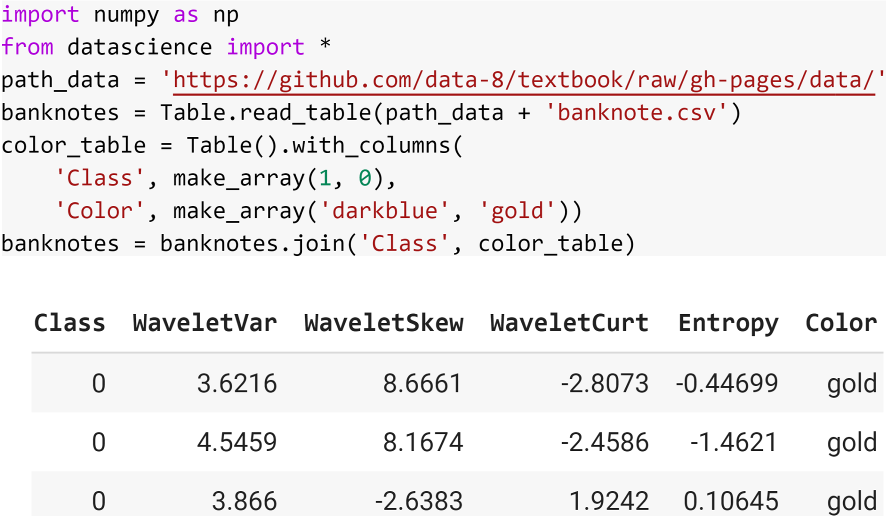
Computational and Inferential Thinking: The Foundations of Data Science

Adhikari & DeNero (2019): Ch 17.4 Implementing the Classifier

17.5 The Accuracy of the Classifier

https://www.inferentialthinking.com/

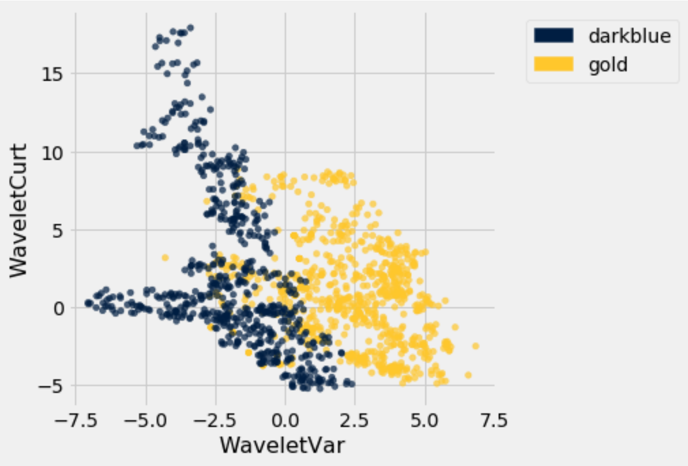
Predicting if a banknote is counterfeit?



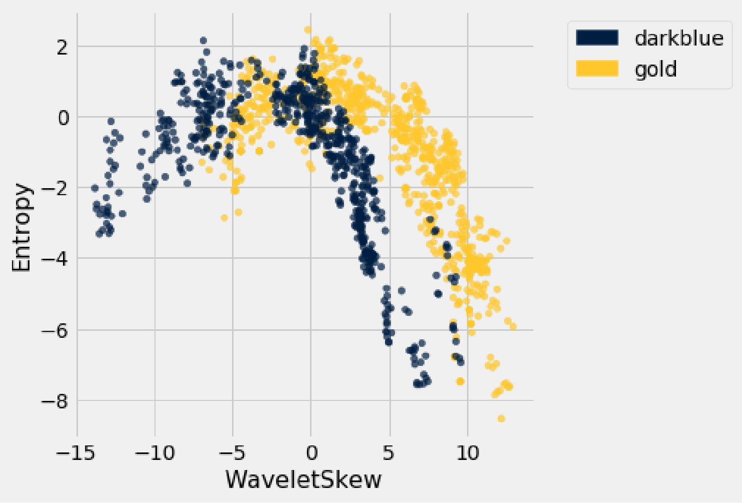
```
import numpy as np
from datascience import *
path_data = 'https://github.com/data-8/textbook/raw/gh-pages/data/'
banknotes = Table.read_table(path_data + 'banknote.csv')
color_table = Table().with_columns(
    'Class', make_array(1, 0),
    'Color', make_array('darkblue', 'gold'))
banknotes = banknotes.join('Class', color_table)
```

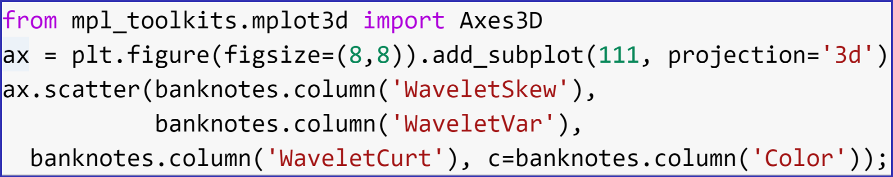
Class	WaveletVar	WaveletSkew	WaveletCurt	Entropy	Color
0	3.6216	8.6661	-2.8073	-0.44699	gold
0	4.5459	8.1674	-2.4586	-1.4621	gold
0	3.866	-2.6383	1.9242	0.10645	gold

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
banknotes.scatter('WaveletVar', 'WaveletCurt', colors='Color')
```

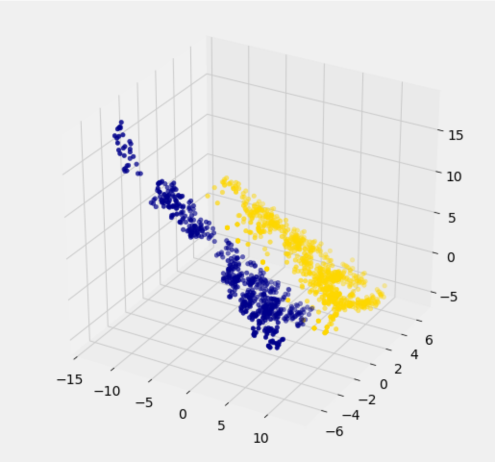


```
banknotes.scatter('WaveletSkew', 'Entropy', colors='Color')
```





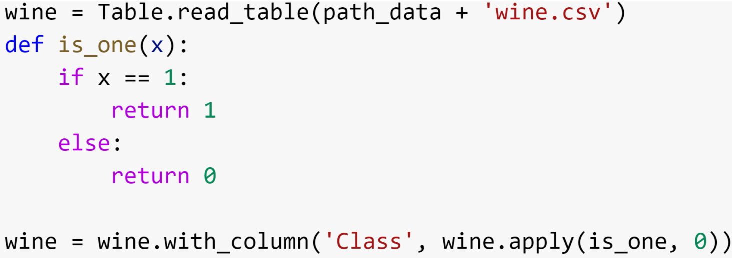
```
from mpl_toolkits.mplot3d import Axes3D
ax = plt.figure(figsize=(8,8)).add_subplot(111, projection='3d')
ax.scatter(banknotes.column('WaveletSkew'),
           banknotes.column('WaveletVar'),
  banknotes.column('WaveletCurt'), c=banknotes.column('Color'));
```



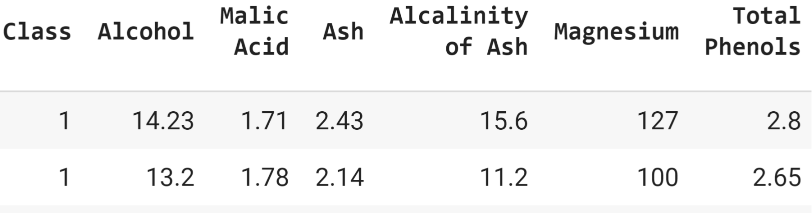
Chemical composition of 178 different Italian wines

There are three classes

Class 1 apart from the other two

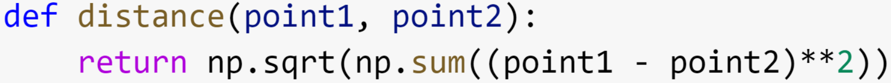


```python
wine = Table.read_table(path_data + 'wine.csv')
def is_one(x):
    if x == 1:
        return 1
    else:
        return 0

wine = wine.with_column('Class', wine.apply(is_one, 0))
```

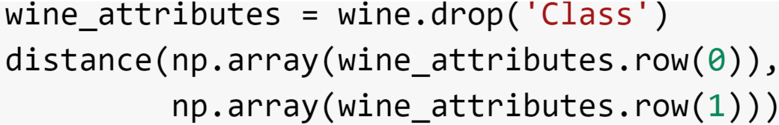


Class	Alcohol	Malic Acid	Ash	Alcalinity of Ash	Magnesium	Total Phenols
1	14.23	1.71	2.43	15.6	127	2.8
1	13.2	1.78	2.14	11.2	100	2.65

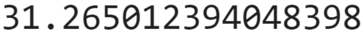
$D = \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2 + (z_0 - z_1)^2}$



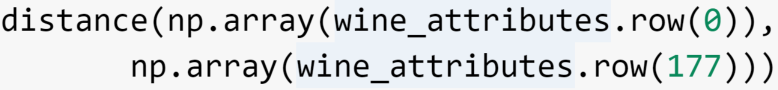
```
def distance(point1, point2):
    return np.sqrt(np.sum((point1 - point2)**2))
```



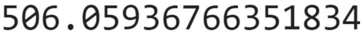
```
wine_attributes = wine.drop('Class')
distance(np.array(wine_attributes.row(0)),
         np.array(wine_attributes.row(1)))
```



```
31.265012394048398
```

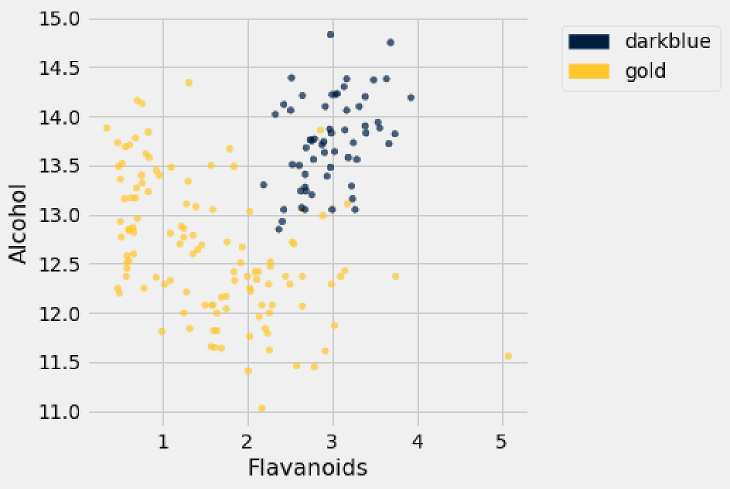


```
distance(np.array(wine_attributes.row(0)),
         np.array(wine_attributes.row(177)))
```

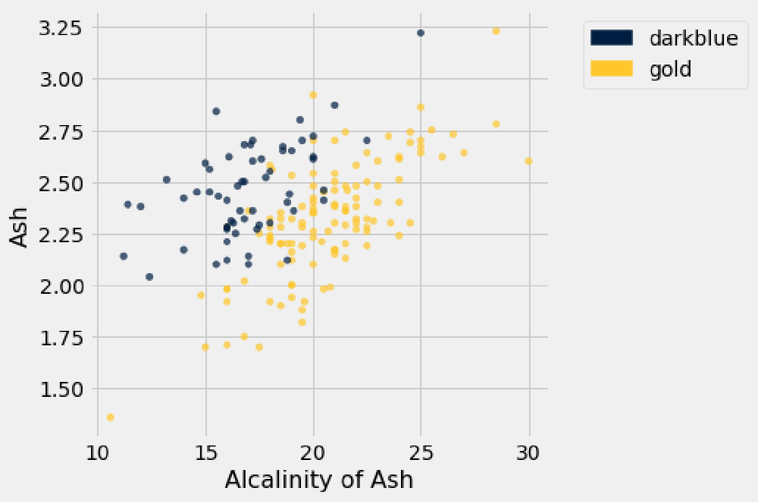


```
506.0593676351834
```

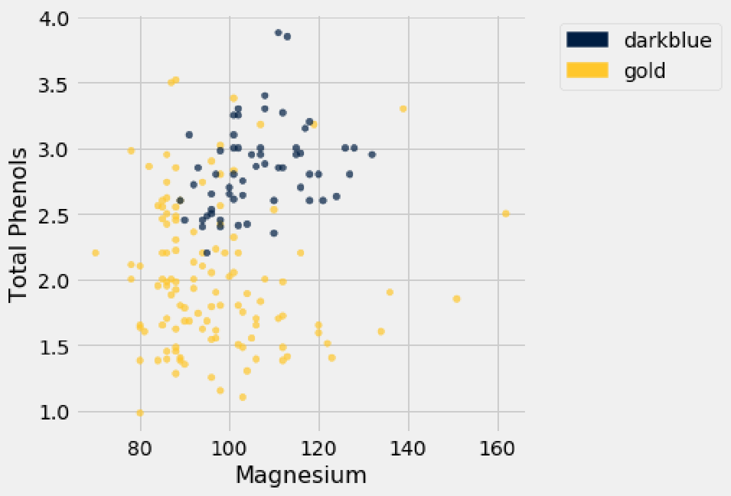
```
wine_with_colors = wine.join('Class', color_table)
wine_with_colors.scatter('Flavanoids', 'Alcohol', colors='Color')
```



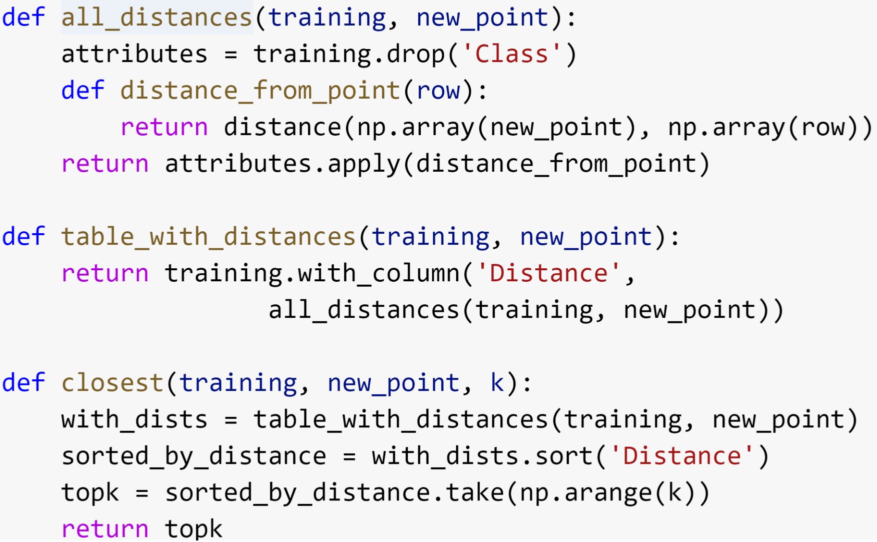
```
wine_with_colors.scatter('Alcalinity of Ash', 'Ash', colors='Color')
```



```
wine_with_colors.scatter('Magnesium', 'Total Phenols', colors='Color')
```



Functions

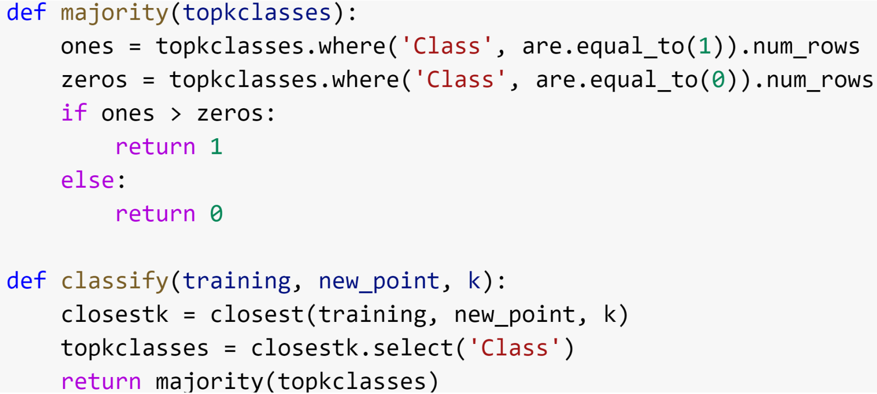


```python
def all_distances(training, new_point):
    attributes = training.drop('Class')
    def distance_from_point(row):
        return distance(np.array(new_point), np.array(row))
    return attributes.apply(distance_from_point)

def table_with_distances(training, new_point):
    return training.with_column('Distance',
                      all_distances(training, new_point))

def closest(training, new_point, k):
    with_dists = table_with_distances(training, new_point)
    sorted_by_distance = with_dists.sort('Distance')
    topk = sorted_by_distance.take(np.arange(k))
    return topk
```

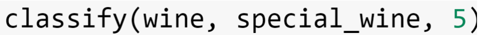
```python
special_wine = wine.drop('Class').row(0)
closest(wine, special_wine, 5)
```

Color Intensity	Hue	OD280/OD315 of diulted wines	Proline	Distance
5.64	1.04	3.92	1065	0
5.85	0.92	3.2	1060	10.3928
5.24	0.87	3.33	1080	22.3407
6.2	1.07	2.75	1060	24.7602
4.9	1.04	3.44	1065	25.0947

Classify Wine

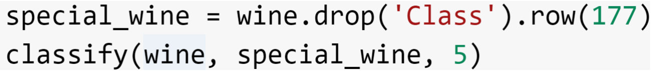


```
def majority(topkclasses):
    ones = topkclasses.where('Class', are.equal_to(1)).num_rows
    zeros = topkclasses.where('Class', are.equal_to(0)).num_rows
    if ones > zeros:
        return 1
    else:
        return 0

def classify(training, new_point, k):
    closestk = closest(training, new_point, k)
    topkclasses = closestk.select('Class')
    return majority(topkclasses)
```



```
classify(wine, special_wine, 5)
```

1, $y_0 = 1$



```
special_wine = wine.drop('Class').row(177)
classify(wine, special_wine, 5)
```

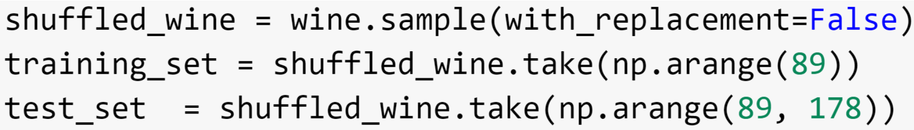
0, $y_{178} = 0$

Functions for Measuring the Accuracy

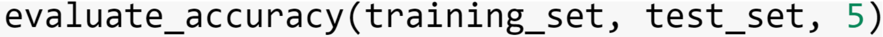
```python
def count_zero(array):
    """Counts the number of 0's in an array"""
    return len(array) - np.count_nonzero(array)

def count_equal(array1, array2):
    """Takes two numerical arrays of equal length
    and counts the indices where the two are equal"""
    return count_zero(array1 - array2)

def evaluate_accuracy(training, test, k):
    test_attributes = test.drop('Class')
    def classify_testrow(row):
        return classify(training, row, k)
    c = test_attributes.apply(classify_testrow)
    return count_equal(c, test.column('Class')) / test.num_rows
```

The Accuracy of the Classifier

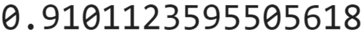
k-nearest neighbor classifier



```
shuffled_wine = wine.sample(with_replacement=False)
training_set = shuffled_wine.take(np.arange(89))
test_set  = shuffled_wine.take(np.arange(89, 178))
```



```
evaluate_accuracy(training_set, test_set, 5)
```



```
0.9101123595505618
```