School of Computing and Information Systems
The University of Melbourne
COMP30027
MACHINE LEARNING (Semester 1, 2019)

Practical exercises: Week 3

Today[1], we will be working with a slightly modified version of the (very famous) *Iris* dataset, first discussed in:

> Fisher, R.A. (1936) The Use of Multiple Measurements in Taxonomic Problems, *Annual Eugenics* 7(II), pp. 179–188.

This dataset (along with many others) is freely downloadable from the UCI Machine Learning Repository (`https://archive.ics.uci.edu/ml/datasets/Iris`). It also comes bundled with `scikit-learn`, but we will not be using that version today.

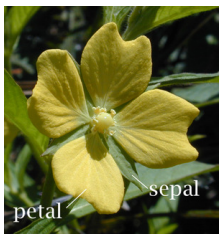
Before we start doing anything, let's frame our problem:

- Let's say that we would like to build a system that can automatically predict the species of a flower, based on a photograph of that flower.
- The data that we have collected at this point is a description of three kinds of *Iris* flowers: *Iris setosa*, *Iris versicolor*, and *Iris virginica*. (In the future, we hope to examine other species/genera of flowers.) We believe that the sepals and petals represent one possible source of useful information to help us solve this problem.

I. setosa

I. versicolor

I. virginica



Sepals & Petals

- The data was collected by field researchers measuring certain properties of the sepals and petals on a sample of the three kinds of flower. (If some of this information proves useful, then we plan on incorporating image–processing software so that this can be discerned automatically.)

1. Begin by visually inspecting the raw data[2] contained within the (textual) file `iris-data-dos.csv` (for Windows) or `iris-data-nix.csv` (for Mac/*nix):
   (a) How many instances are there? How many attributes? The instances have been labelled with class information — so this dataset is suitable for **supervised** machine learning — how many classes are there?
   (b) Confirm your observations by writing a Python script to count the instances, and keep track of how many instances of each class there are. (You might find the `string.split()` method useful.) You should notice a couple of problems with the class labels; make a copy of the CSV file called `iris-clean.csv`, and edit the erroneous class labels with their (likely) correct values.

[1] The dataset and images, along with some of the problems in this worksheet have been adapted from work by Randal Olson (`http://www.randalolson.com/`) which we use under its Creative Commons license (`https://creativecommons.org/licenses/by/4.0/`).

[2] Note that "inspecting the data" is mostly ineffectual for very large datasets. In fact, this is one of the main motivators for Machine Learning! `:-)`

2. Before we can think about building a classifier, we should double–check that the data is formatted correctly. Confirm that the first line of the dataset is a header, which describes the expected format.

   (a) Write a function `check-csv()` which returns `True` if every instance has the same number of fields (in this case, attribute values separated by commas), and `False` otherwise.

   (b) Write a function which, for each attribute, prints the number of instances where the attribute value is numeric[3]. What do you observe?

   (c) For the instances with non-numeric attribute values, the values appear to be missing. We can't know what the true values of these attribute were supposed to have been, but leaving them unknown might cause problems for our model. Use the `matplotlib` library to plot a histogram for the data as follows:

```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> f = open("./iris-data-nix.csv", 'r') # This line is platform-specific
>>> petal_width = []
>>> for line in f.readlines()[1:]:
...     pw = line.split(",")[3]
...     if not pw == "NA":
...         petal_width.append(float(pw))
>>> f.close()
>>> plt.hist(np.array(petal_width))
>>> plt.show()
```

   One possible approach toward removing missing values is **mean imputation**: replacing a missing attribute value with the mean of the observed values for that attribute. Why is this a bad idea, based on the data you observe? Is there anything else we could do, so as to make mean imputation a plausible strategy here?

3. Use the method above to plot the histograms for the other attributes. You should observe some clear **outliers** in the attribute values. Take a look at the raw data, and see if you can guess what could have caused the outliers. Modify your `iris-clean.csv` copy of the data, to fix the outlying attribute values.

4. Let's attempt to visualise some relationships in the data.

   (a) Make a **scatter-plot** of the sepal length vs. sepal width by using the following:

```
>>> plt.scatter(np.array(sepal_length),np.array(sepal_width))
>>> plt.show()
```

   It is suggestive, but without a clear indication of the classes of the instances, we can't really be certain of any patterns in the data.

   (b) The scatter function takes an optional argument for colouring points, based on a list of strings (like `red` or `blue`). Choose some suitable colours for `colour`, and replot as follows:

```
>>> plt.scatter(np.array(sepal_length),np.array(sepal_width),c=colour)
>>> plt.show()
```

   (c) Try making scatter-plots for different pairs of attributes. Do you notice anything that might suggest that one class is distinguishable from the others? How might this information be utilised by the Naive Bayes algorithm? What about other supervised machine learning methods?

[3]Python 3 has an `isnumeric()` string method, but Python 2 doesn't — if you are using the latter, regular expressions are recommended; for example: `if re.match(r'^[0-9.]+$', string):`