Machine Learning Workshop – Session 1

Machine learning course outline/overview:

Day 1:
- Session 1: Python review
- Session 2: Machine learning basics with KNN

Day 2:
- Session 1: Preprocessing data for machine learning
- Session 2: Machine learning model tuning & evaluation

Day 3:
- Session 1: Naïve Bayes & Random Forest
- Session 2: Regression & unsupervised Learning

Day 4:
- Session 1: Neural Network
- Session 2: Deep Learning using Keras & Tensorflow

Python Review:

Live Coding basic python functions (Improvisational):

- Conditionals (if, elif, else) (also explain indenting for code blocks)
- Lists: initializing, appending, adding, extending
- for/while loops
- writing a function (def myFunction():)
- Importing (show numpy and pandas)
- Using methods from a library using the dot (.) operator, also explain "tabbing" through methods
- Googling a method to understand what it does
- **parameters/arguments** and **return values**

Working with Numpy Arrays:

Initializing a numpy array from a list:

```
np.array([2,4,6,8])
```

Which is the same as:

```
a = [2,4,6,8]
a
np.array(a)
```

Two-Dimensional Example:

```
np.array([[2,4,6,8],
          [1,3,5,7]])
```

Initialize a numpy array using ones:

```
m  = np.ones((10, 5))
m
```

We can check the shape using np.shape:

```
np.shape(m)
```

Result isn't stored in m:

```
m*5
m
```

The following will store the new values in m:

```
m = m*5
m
```

Numpy can represent more than just matrices (i.e., tensors):
Tensors are a representation of n-dimensional coordinate space.

```
np.zeros((10, 5, 7))
```

The above has 10 units for dimension 1, 5 units for dimension 2, and 7 units for dimension 3.

Accessing/Manipulating specific values within a numpy array:

```
m = np.array([[2,4,6,8],
              [1,3,5,7],
              [9,10,11,12],
              [0,0,0,0]])
m
```

Selecting a value from a single cell:

```
m[0,0]
```

Assigning a single unit of a matrix/tensor:

```
m[0,0] = 100
m
```

We can specify a range of all rows in the first column using the following :

```
m[:,0]
```

Specify all values from the first column, starting with the 2nd row (index = 1):

```
m[1:,0]
```

All values less than the third index:

```
m[:3,0]
```

Or indices greater than or equal to 1 but strictly less than 3:

```
m[1:3,0]
```

This can be applied for each dimension:

```
m[1:3, 1:3]
```

Using selection to assign values:

```
m[1:3,:] = 100
m
```

All rows between 1 and 3 are now assigned to 100.

Selecting using an index list:

```
m[[0,3], :]
```

Creating filters:

```
hundredfilter = m[:, 0] == 100
m[hundredfilter,:]
```

Combining filters using & and |:

```
fourzerofilter= (m[:, 0] == 4) | (m[:, 0] == 0)
m[fourzerofilter,:]
```

The **|** is the logical **OR** operator
The **&** is the logical **AND** operator

Other useful functions:
np.sum(nparray ,axis=0)
np.mean(nparray ,axis=1)

Reading data from a file using pandas:

```
import pandas as pd
mydata = pd.read_csv("RandomData.txt", sep="\t", header=None)
mydata
```

Using the read_csv method from pandas.
sep - Separating is the column delimiter of the file. Usually this is with a tab ("\t") or a comma (,)
header - Parameter to specify the column names or which row contains the column names (default = "infer").

When reading a file, it's important to identify:

1) The columns
2) The file delimiter. Such as a comma (",") **e.g.:** "1,2,3" or tab ("\t") **e.g.:** "1 2 3"

We can obtain the underlying numpy array by accessing the values variable of the DataFrame:

```
mydatavals = mydata.values
mydatavals
```

Plotting data using matplotlib:

Importing the plotting package:

```
import matplotlib.pyplot as plt
```

Setting up example data:

```
x = [5,10,15,20]
y = [10,5, 20,15]
```

Generating a scatterplot:

```
plt.scatter(x,y)
```

Generating a bar plot:

```
plt.bar([1,2,3,4], x)
```

Generating a line plot:

```
plt.plot(y)
```

Generating a boxplot:

```
plt.boxplot((x,y))
```

Look for further documentation using google!

Exercise:

1. Read contents of "RandomData.txt" into a variable as a numpy array.
2. Separate the data into two numpy arrays, where:
    a. one array holds data from columns 1-10 and
    b. the other array holds columns 11-20.
3. Create two new numpy arrays for each selected array in 2 selecting:
    a. the top 50 rows for one numpy array and
    b. all the remaining rows for the other numpy array.

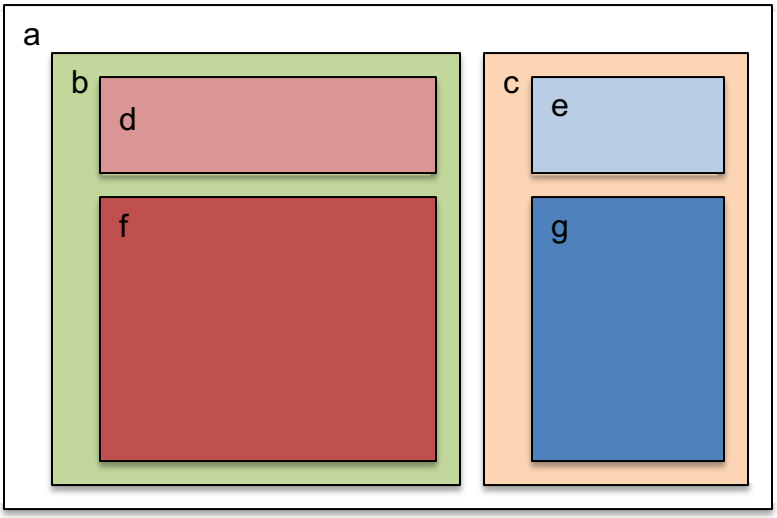
The starting size of the array is 500 rows by 20 columns.
Use np.shape to confirm that the array selection matches the dimensions.

In total you should have 7 numpy arrays assigned to different variables.

a. One numpy array for the entire dataset

b. One numpy array containing ALL rows with columns 1-10.
c. One numpy array containing ALL rows with columns 11-20.

d. One numpy array containing the top 50 rows for columns 1-10.
e. One numpy array containing the top 50 rows for columns 11-20.
f. One numpy array containing the remaining rows for columns 1-10.
g. One numpy array containing the remaining rows for columns 11-20.



Additional Exercise: Select a column and generate any type of plot from this data.