GETTING STARTED WITH THE ASSIGNMENT IN Python

We start with reading the credit scoring data from the lectures in Python.
The data can be found here.
Suppose you saved the data to a file "credit.txt" in the directory "dm"
on the C drive. To read it into Python type (">>>" denotes the prompt):

```
>>> import numpy as np
>>> credit_data = np.genfromtxt('C:/dm/credit.txt', delimiter=',', skip_header=True)
```

To display its value, just type its name at the command line:

```
>>> credit_data
array([[22.,  0.,  0., 28.,  1.,  0.],
       [46.,  0.,  1., 32.,  0.,  0.],
       [24.,  1.,  1., 24.,  1.,  0.],
       [25.,  0.,  0., 27.,  1.,  0.],
       [29.,  1.,  1., 32.,  0.,  0.],
       [45.,  1.,  1., 30.,  0.,  1.],
       [63.,  1.,  1., 58.,  1.,  1.],
       [36.,  1.,  0., 52.,  1.,  1.],
       [23.,  0.,  1., 40.,  0.,  1.],
       [50.,  1.,  1., 28.,  0.,  1.]])
```

"credit_data" is now a 2d NumPy array. Each rows represent a record and the columns represent the data attributes.

Select the first row of credit_data:

```
>>> credit_data[0]
array([22.,  0.,  0., 28.,  1.,  0.])
```

Select the fourth column of credit_data:

```
>>> credit_data[:,3]
array([28., 32., 24., 27., 32., 30., 58., 52., 40., 28.])
```

Select the element in row 4, column 0:

```
>>> credit_data[4,0]
29.0
```

Give the distinct values of income, sorted from low to high:

```
>>> np.sort(np.unique(credit_data[:,3]))
array([24., 27., 28., 30., 32., 40., 52., 58.])
```

Add all the entries of the sixth column:

```
>>> np.sum(credit_data[:,5])
5.0
```

Add the entries of each column of credit_data:

```
>>> credit_data.sum(axis=0)
array([363.,   6.,   7., 351.,   5.,   5.])
```

Add the entries of each row:

```
>>> credit_data.sum(axis=1)
array([ 51.,  79.,  51.,  53.,  63.,  78., 125.,  91.,  65.,  81.])
```

Select all rows where the first column is bigger than 27:

```
>>> credit_data[credit_data[:,0] > 27]
array([[46.,  0.,  1., 32.,  0.,  0.],
       [29.,  1.,  1., 32.,  0.,  0.],
       [45.,  1.,  1., 30.,  0.,  1.],
       [63.,  1.,  1., 58.,  1.,  1.],
       [36.,  1.,  0., 52.,  1.,  1.],
       [50.,  1.,  1., 28.,  0.,  1.]])
```

Construct a vector "x" with the numbers 2, 5, 10 in that order:

```
>>> x = np.array([2, 5, 10])
>>> x
array([ 2,  5, 10])
```

Construct a vector consisting of the numbers 0 through 9:

```
>>> np.arange(0, 10)
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

Select the *row numbers* of the rows where the first column of credit_data is bigger than 27:

```
>>> np.arange(0, 10)[credit_data[:,0] > 27]
array([1, 4, 5, 6, 7, 9])
```

Draw a random sample of size 5 from the numbers 1 through 10 (without replacement):

```
>>> index = np.random.choice(np.arange(0, 10), size=5, replace=False)
>>> index
array([5, 7, 1, 3, 8])
```

Select the corresponding rows:

```
>>> train = credit_data[index,]
>>> train
array([[45.,  1.,  1., 30.,  0.,  1.],
       [36.,  1.,  0., 52.,  1.,  1.],
       [46.,  0.,  1., 32.,  0.,  0.],
       [25.,  0.,  0., 27.,  1.,  0.],
       [23.,  0.,  1., 40.,  0.,  1.]])
```

Select all rows with row number not in "index":
(This does not delete any rows from the original credit_data.)

```
>>> test = np.delete(credit_data, index, axis=0)
>>> test
array([[22.,  0.,  0., 28.,  1.,  0.],
       [24.,  1.,  1., 24.,  1.,  0.],
       [29.,  1.,  1., 32.,  0.,  0.],
       [63.,  1.,  1., 58.,  1.,  1.],
       [50.,  1.,  1., 28.,  0.,  1.]])
```

Consult the help page of the function "np.random.choice"

```
>>> help(np.random.choice)
```

Practice exercise 1

Assume we have a classification problem with only 2 classes that are labeled 0 and 1 respectively.
Write a function that computes the impurity of a vector (of arbitrary length) of class labels.
Use the gini-index as impurity measure. Do not use a loop structure in your function,
this is not necessary.

Example:

```
>>> array=np.array([1,0,1,1,1,0,0,1,1,0,1])
>>> array
array([1,0,1,1,1,0,0,1,1,0,1])

>>> impurity(array)
0.23140495867768596
```

Practice exercise 2

Write a function "bestsplit(x,y)" that computes the best split value on a numeric attribute x.
Here x is a vector of numeric values, and y is the vector of class labels (assume there
are only two classes, coded as 0 and 1). x and y must be of the same length: y[i] is the class
label of the i-th observation, and x[i] is the corresponding value of attribute x.
Only consider splits of type "x <= c" where "c" is the average of two consecutive values of x in the sorted order.
So one child contains all elements with "x <= c" and the other child contains all elements with "x > c".
The best split is the split that achieves the highest impurity reduction.

Example (best split on income):

```
>>> bestsplit(credit_data[:,3],credit_data[:,5])
```

```
  36
```

Hint: Clever use of "subscripting" (selecting elements of vectors and matrices) is important. For example, y[x > 29] produces a vector with all elements of y whose corresponding x-element (that is the element of x with the same index) is bigger than 29. More formally: y[x > 29] = {y[i]: x[i] > 29}. The result is a vector, not a set, i.e. duplicate values may occur. Just try it!

Hint: Example of how to determine candidate split points

```
>>> income_sorted = np.sort(np.unique(credit_data[:,3]))
>>> income_sorted
  array([24, 27, 28, 30, 32, 40, 52, 58])

>>> income_splitpoints = (income_sorted[0:7]+income_sorted[1:8])/2
>>> income_splitpoints
  array([25.5, 27.5, 29. , 31. , 36. , 46. , 55. ])
```

Note: use the "brute force" approach, i.e. don't implement the "segment borders" algorithm.