Data Analytics

Lab 6

Working with CSV data files

A comma-separated values (CSV) file is a delimited text file that uses a comma to separate values. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. A CSV file typically stores tabular data (numbers and text) in plain text, in which case each line will have the same number of fields. (Wikipedia)

Data analysis becomes easy when you have data files to analyze. The numpy library of pythons provides functions that helps you to read from and write into the files. Considering the example from last lab, if you have the climate data of 10000 regions, how will you find, which region is best suitable for yielding apples.

```
In [3]: import numpy as np

In [4]: climate_data = np.genfromtxt('climate_data.csv', delimiter=',', skip_header=1)

In [5]: climate_data

Out[5]: array([[25., 76., 99.],
               [39., 65., 70.],
               [59., 45., 77.],
               ...,
               [99., 62., 58.],
               [70., 71., 91.],
               [92., 39., 76.]])

In [6]: climate_data.shape

Out[6]: (10000, 3)

In [7]: weights = np.array([0.3, 0.2, 0.5])
```

We can now perform a matrix multiplication using the @ operator to predict the yield of apples for the entire dataset using a given set of weights.

```
In [8]: yields = climate_data @ weights

In [9]: yields

Out[9]: array([72.2, 59.7, 65.2, ..., 71.1, 80.7, 73.4])
```

Let's add the yields to climate_data as a fourth column using the np.concatenate function.

```
In [11]: climate_results = np.concatenate((climate_data, yields.reshape(10000, 1)), axis=1)

In [12]: climate_results

Out[12]: array([[25. , 76. , 99. , 72.2],
                [39. , 65. , 70. , 59.7],
                [59. , 45. , 77. , 65.2],
                ...,
                [99. , 62. , 58. , 71.1],
                [70. , 71. , 91. , 80.7],
                [92. , 39. , 76. , 73.4]])
```

Tasks:

1. Replace all odd numbers in the given array with -1
   task1 = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
2. Find the positions of:
   1. elements in x where its value is more than its corresponding element in y, and
   2. elements in x where its value is equals to its corresponding element in y.

      x = np.array([21, 64, 86, 22, 74, 55, 81, 79, 90, 89])
      y = np.array([21, 7, 3, 45, 10, 29, 55, 4, 37, 18])
3. Generate a 2-D array of random numbers of size (1000,3). Find the mean of three random numbers in each row and store the mean of each row in a fourth column of that row. Save the result in a csv file by the name “mean.csv” on your local disk with headers as “num1, num2, num3, average”. Also find which row has the highest mean of all.