Department of Computational and Data Sciences
DS226: Introduction to Computing for Artificial Intelligence and Machine Learning
October 13, 2022

Assignment 2

Instructions:

- **Submit a typed report in PDF format on Moodle. Use LaTeXpreferably. Handwritten reports will not be accepted.**
- **Solutions should be in the same order as the questions.**
- **Discussion is encouraged, but answers should be your own. Do not copy answers. Plagiarism will be penalised severely.**
- **For late submissions, the following late submission policy:**

Delay	% of Credit
0-72 hours	**50**
72 hours - 1 week	**25**
Beyond 1 week	**No credit**

Deadline: 27th October, 11:59 AM
Total Credits: 100

Question 1. (30)

Take any dataset from https://data.gov.in/ and produce visualisations using matplotlib, pandas,seaborn and numpy.
Consider yourself as an independent evaluator presenting the findings based on the data to Government authorities who are going to make decisions based on the data(visualisations) that you present. The objective here is to create a **Data Story** and to make sure each plot conveys some inference (can also be some hand-wavy inference) related to the data. Feel free to use more plots than the ones mentioned below if you feel that it is meaningful and helps in explaining your data more concretely.

1) Submit one jupyter notebook with all code plots and any comments/notes you have to make (comments can be made via markdown in jupyter)
2) You are required to do at least the following
   - Scatter plot using matplotlib
   - Line plot
   - Bar plot or dot plot
   - Box plot or violin plot
   - Pair plot using seaborn (make sure you justify the plots that you choose to show in the pairplot)
3) For each plot write what inferences you can draw from the plot.
4) Each plot should be legible and must be properly labelled.
5) Finally, a single jupyter notebook is required for this question. Once you save your jupyter notebook make sure all the outputs are seen when it is reopened (without the need for compiling it again).

Question 2. (30)

Consider a vector $\mathbf{v} \in \mathbb{R}^n$,

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

stored as a 1-dimensional array. Recall that the sum of all elements of this vector, $\sum_{i=1}^{n} v_i$, can be computed with a parallel implementation using OpenMP

```
#pragma omp parallel for default(shared) reduction(+: sum_arr)
    for (i = 0; i <= ARRAY_SIZE; i++)
        sum_arr += arr[i]; // Sum up the array
```

Now, consider the dot product between two vectors, $\mathbf{v}, \mathbf{w} \in \mathbb{R}^n$,

$$\begin{aligned} \mathbf{v} \cdot \mathbf{w} &= \mathbf{v}^T \mathbf{w} \\ &= \begin{bmatrix} v_1 & v_2 \dots & v_n \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} \\ &= \sum_{i=1}^{n} v_i w_i \end{aligned}$$

The dot product is used widely to calculate vector projections, vector decompositions and to determine orthogonality. In the context of neural networks, the dot product can be used to calculated the weighted sum of inputs to a neuron.

I. **Write a routine to calculate the dot product of two 1-dimensional arrays with double values. The routine should be an efficient, parallel implementation in C using OpenMP.**

Another commonly used linear algebra routine is *DAXPY*. For two vectors $\mathbf{v}, \mathbf{w} \in \mathbb{R}^n$ and the scalar $\alpha \in \mathbb{R}$, *DAXPY* performs the following operation

$$\mathbf{v} \leftarrow \mathbf{v} + \alpha \mathbf{w}$$

Assume that $\mathbf{v}$ and $\mathbf{w}$ are both stored as 1-dimensional arrays of double values and α is a double constant.

II. **Write a parallel implementation of DAXPY using OpenMP in C. The routine should take two 1-dimensional arrays with double values and a double scalar and perform the DAXPY operation mentioned above.**

Next, we consider the L_2 norm of the difference between two vectors $\mathbf{v}, \mathbf{w} \in \mathbb{R}^n$,

$$\|\mathbf{v} - \mathbf{w}\|_{L_2} = \sqrt{\sum_{i=1}^{n} (v_i - w_i)^2}$$

This can be used to calculate the error between a predicted and observed value, and finds extensive usage in ML algorithms. Assume $\mathbf{v}$ and $\mathbf{w}$ to be two 1-dimensional arrays with double values.

III. **Write a routine to calculate the L_2 norm error between two 1-dimensional arrays with double values. Use the parallel dot product and DAXPY functions you have implemented earlier to build this routine.**

Lastly, recall that a matrix-vector product appears in the solution of the normal equation for a linear regression problem. For a given matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and a vector $\mathbf{x} \in \mathbb{R}^n$, the matrix-vector product is given

by

$$\mathbf{Ax} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{bmatrix}$$

Assume **A** to be stored as a 1-dimensional array of double values with length $m \times n$ in the row-major form and **x** to be stored as a 1-dimensional array of length n

IV. **Write a routine to compute the matrix-vector product Ax. Use the parallel dot product function you have implemented earlier to build this routine.**

Question 3. (40)

This is a programming question to recreate a very simplified version of the classic Snake game. In this simplified version, you have a snake with a goal to move around and eat pieces of food in the game area. Once the game starts, the snake moves continuously in the user input direction. Each time snake eats food, it grows i.e. its length increases. You lose the game if the snake runs into itself.

Setup:

The essential objects for building the game are,

1) Screen or the game window
2) Snake Head
3) Snake Body
4) Food pieces

You have been provided template for the program written in Python. The template takes care of the game window and other GUI implementations using `turtle`. You have to write the main game logic components like movement of snake's head, random food placement in game area and consumption condition for food, increase of snake's length after successful consumption of food piece and the check for the game end condition using **OOP concepts in Python**. Rest instructions are given in the template. Each function is accompanied by a set of comments that will tell you what the function implements, what it accepts as parameters and what it returns. You are given the function definitions in the template.

Objectives:

1) Your code must run after you submit it.
2) Food should appear at random places inside the game window.
3) Snake length should increase by one unit after consuming the food piece.
4) Game should end if the snake runs into itself.