# CSCI316 – Big Data Mining Techniques and Implementation
# Assignment 2
# Autumn 2023

**10 Marks**
**Deadline: Refer to the submission link of this assignment on Moodle**

**Two tasks are included in this laboratory. The specification of each task starts in a separate page.**

**You must implement and run all your Python code in Jupyter Notebook.** ***The deliverables include one Jupyter Notebook source file (with .ipybn extension) and one PDF document for each task.***

**To generate a PDF file for a notebook source file, you can either (i) use the Web browser's PDF printing function, or (ii) click "File" on top of the notebook, choose "Download as" and then "PDF via LaTex".**

**The submitted source file(s) and PDF document(s) must show that all of your code has been executed successfully. Otherwise, they will not be assessed.**

***This is an individual assessment. Plagiarism of any part of this assessment will result in having 0 mark for this assessment and for all students involved.***

**The correctness of your implementation and the clearness of your explanations will be assessed.**

# Task 1

(4 marks)
**Dataset**: kddcup.data_10_percent
Source: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
(Note. The data is in the file kddcup.data_10_percent.gz.)

The kddcup.data_10_percent dataset is a small subset of a larger benchmark problem. The benchmark problem aims at assisting researchers to develop methods that can reliably detect network intrusions in real time. A very brief description of the data, the names of attributes and classes, and more can be found on the corresponding source webpage. The classes are DOS (Denial of Service), Probe, R2L (Root 2 Local), U2R (User 2 Root) and Normal.

**Objective**
Implement a Naïve Bayesian classifier from scratch in Python.

**Task requirements**
(1) Use stratified sampling to separate the data into two subsets: ~70% for training and ~30% for test.
(2) The Naïve Bayesian classifier must implement techniques to overcome the *numerical underflows* and *zero counts*.
(3) To handle the continuous feature, you can use either binning or a continuous distribution (such as the Gaussian distribution).
(4) No ML library can be used in this task. The implementation must be developed from scratch. However, scientific computing and data analytics libraries such as NumPy and Pandas are allowed.

**Deliverables**
- A Jupiter Notebook source file named `your_name_task1.ipybn` which contains your implementation source code in Python
- A PDF document named `your_name_task1.pdf` which is generated from your Jupiter Notebook source file, and which includes clear and accurate explanation of your implementation and results.

# Task 2

(3 marks)
**Dataset**: Webpage links (gr0. California)
Source: https://www.cs.cornell.edu/courses/cs685/2002fa/data/gr0.California

**Dataset information**
This dataset contains about 9664 webpages and their hyperlinks. The records with the "n" flag contain (unique) IDs and URLs of webpages. Those with the "e" flag represent hyperlinks between webpages. Essentially, this data set models a Web.

**Objective**
Performance explorative analysis of this data set with Apache Spark. You must load the data into Spark and use Spark DataFrame, Pandas on Spark and RDD to compute the required outputs. (Your implementation must only include operations (i.e., transformations and actions) on the three data structures in Spark.)

The following definitions are used in this task: The *out-degree* of a webpage is the number of hyperlinks which that webpage has. The *in-degree* of a webpage is the number of webpages which have a hyperlink to that webpage. For example, if a webpage A contains two hyperlinks in total, then the out-degree of A is 2; if in total there are four webpages (say, A, B, C and D) containing a hyperlink to A, then the in-degree of A is 4. (Note that a webpage might have a hyperlink to itself.)

**Takes requirements**
Return the following outputs:

(1) The webpage(s) with the largest out-degree;
(2) The webpage(s) with the largest in-degree;
(3) The average out-degree;
(4) The average in-degree;
(5) The number of webpages whose out-degree is 0.

**Deliverables**
- A Jupiter Notebook source file named `your_name_task2.ipybn` which contains your implementation source code in Python
- A PDF document named `your_name_task2.pdf` which is generated from your Jupiter Notebook source file, and which includes clear and accurate explanation of your implementation and results.

# Task 3

(3 marks)
**Dataset**: Webpage links (gr0. California)
Source: https://www.cs.cornell.edu/courses/cs685/2002fa/data/gr0.California

**Objective**
The objective of this task is to implement a PageRank algorithm to rank the webpages in the dataset.

**Takes requirements**

(1) Generate a $5 \times 5$ block matrix for the transition matrix of the web graph, and a $5 \times 1$ block matrix for the initial rank vector (which is a uniform vector).

(2) Implement the sparse matrix formulation of the PageRank algorithm. Use a teleport probability $\beta = 0.85$. The computation of the (evolving) rank vector is iterated 10 times. Then, return the ranking scores of the first 20 webpages (i.e., webpages from ID 0 to ID 19).

Note. A block matrix in this task refers to the data structure of pyspark.mllib.linalg.distributed.BlockMatrix. The BlockMatrix supports matrix addition and multiplication (see its API doc). However, you may need to convert a BlockMatrix to other Spark data structures (such as RDD) to perform other necessary operations.

**Deliverables**
- A Jupiter Notebook source file named `your_name_task3.ipybn` which contains your implementation source code in Python
- A PDF document named `your_name_task3.pdf` which is generated from your Jupiter Notebook source file, and which includes clear and accurate explanation of your implementation and results.

--End--