CSCI316 – Big Data Mining Techniques and Implementation
Assignment 1

10 Marks
Deadline: Refer to the Assignment1 submission link on Moodle

<u>Three (3) tasks</u> are included in this assignment. 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.*

Note: 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".

All results of your implementation must be reproducible from your submitted Jupyter notebook source files. In addition, the submission must include all execution outputs as well as clear explanation of your implementation algorithms (e.g., in the Markdown format or as comments in your Python codes).

Submission must be done online by using the submission link associated with assignment 1 for this subject on MOODLE. The size limit for all submitted materials is 20MB. DO NOT submit a zip file.

Submissions made after the due time will be assessed as late submissions. Late submissions are counted in full day increments (i.e. 1 minute late counts as a 1 day late submission). There is a 25% penalty for each day after the due date including weekends. The submission site closes four days after the due date. No submission will be accepted after the submission site has closed.

This is an <u>individual assignment</u>. Plagiarism of any part of the assignment will result in having 0 mark for the assignment and for all students involved.

Marking guidelines

Code: Your Python code will be assessed. The computers in the lab define the standard environment for code development and code execution. Note that the correctness, completeness, efficiency, and results of your executed code will be assessed. Thus, code that produces no useful outputs will receive zero marks. This also means that code that does not run on a computer in the lab would be awarded zero marks or code where none of the core functions produce correct results would be awarded zero marks.

Presentation and explanation: The correctness, completeness and clearness of your answers will be assessed.

Task1

Dataset: MNIST
Source: http://yann.lecun.com/exdb/mnist/

Dataset information

The MNIST dataset includes a set of 70,000 small images of digits handwritten by high school students and employees of the US Census Bureau. Each image has 784 features, because each image is 28×28 pixels, and each feature simply represents one pixel's intensity, from 0 (white) to 255 (black). Each image is labelled with the digit it represents. The MNIST dataset is already split into a training set (the first 60,000 images) and a test set (the last 10,000 images).

Objective

The objective of this task is to develop a small end-to-end data mining project by using the machine learning library Scikit-Learn in Python. The output of the project is a classification model that classifies the handwritten images in the MNIST dataset.

Dataset download

The MNIST dataset can be continently downloaded by using Scikit-Learn:

```
from sklearn.datasets import fetch_openml
your_data_home = '<your directory>'
mnist = fetch_openml('mnist_784', version=1, data_home=your_data_home)
```

Image viewing

To view the images, reshape it to a 28×28 array, and display it using Matplotlib's imshow() function:

```
import matplotlib as mpl
import matplotlib.pyplot as plt
X = mnist["data"]
some_digit = X[0]
some_digit_image = some_digit.reshape(28, 28)
plt.imshow(some_digit_image, cmap = mpl.cm.binary, interpolation="nearest")
plt.axis("off")
plt.show()
```

Task requirements

(1)   Main steps of the project are (a) "discover and visualise the data", (b) "prepare the data for machine learning algorithms", (c) "select and train models", (d) "fine-tune the models" and (e) "evaluate the outcomes". You can structure the project in your own way. Some steps can be performed more than once.
(2)   Explanation of each step together with the Python codes must be included.
(3)   In the steps (c) and (d), you must work with at least three models.
(4)   Use the first 60,000 images as the training dataset and 10,000 as the test set.

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 presents clear and accurate explanation of your implementation and results. A poor presentation of results gains less marks in this task.

Task2

Dataset: MAGIC Gamma Telescope Dataset
Source: https://archive.ics.uci.edu/ml/datasets/MAGIC+Gamma+Telescope
(The same dataset is also included in the datasets folder of this assignment)

Dataset information
The data are Monte-Carlo generated to simulate registration of high energy gamma particles in a ground-based atmospheric Cherenkov gamma telescope using the imaging technique. The dataset contains 19,020 records. The records have 10 feature attributes and one class attribute.

Objective
Train decision trees based on the MAGIC Gamma Telescope Dataset to predict whether the collected patterns are caused by the primary gamma (signal) or hadron (background), and compared the performance (e.g., accuracy) of the classifiers.

Task requirements
(1)   Implement a decision tree (DT) classifier by using one the information gain as the split criteria. Note that you can use either binary-split or multiple-split; you can also use any reasonable pre-processing method and any reasonable early stopping criteria (e.g. pre-pruned parameters) but need to explain your reasons.
It is recommended that each classifier includes a tree induction function, a classification function and any other auxiliary functions.
(2)   Implement a 10-fold cross-validation to evaluate the model.
.

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 presents clear and accurate explanation of your implementation and results. A poor presentation of results gains less marks in this task.

Task 3

Dataset: Files "wordsList" and "classList"
(available in the datasets folder of this assignment on Moodle)

Dataset information
The wordsList file contains 72 pre-processed emails. Each line is a list of words extracted from each email. The classList file contains the class labels that indicating whether the emails are ordinary or adverts (0 for ordinary and 1 for adverts).

Objective
Develop a Naïve Bayesian classifier as an email filter in Python. Namely, the classifier predicts whether emails are ordinary or adverts.

Task requirements
(1)   Use stratified sampling to select 66 out of 72 lines for training and the remaining 6 lines for test. Return the classification probabilities of these 6 records.
(2)   The Naïve Bayesian classifier must account for *multiple occurrences* of words and implements techniques to overcome the *numerical underflows* and *zero counts*.
(3)   No ML library can be used in this task. The implementation must be developed from scratch. However, scientific computing libraries such as NumPy and SciPy are allowed.

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 presents clear and accurate explanation of your implementation and results. A poor presentation of results gains less marks in this task.