Part 2 Coursework 1 (20 marks)

Make sure you justify your answers with technical evidence - when in doubt, give details! Remember, any external material used **must be cited**.

Q1. Classifiers: Behaviour (8 marks)

Note: this is the same assignment #2 from the Data Mining course available here.

Start with the *genes-leukemia.csv* dataset used in the lab (on Blackboard). As a predictor use **TREATMENT_RESPONSE**, which has values *Success* , *Failure*, or "?" (missing).

Examine the record where TREAMENT_RESPONSE is non-missing

Q1.1. Count the number of such records, and describe these records using other sample fields (*Year from XXXX to YYYY, Gender = X etc.*) (1 mark)

Q1.2. Explain why it is not correct to build predictive models for TREAMENT_RESPONSE using records where it is missing? (1 mark)

Select only the records with non-missing TREATMENT_RESPONSE. Keep SNUM (sample number) but remove sample fields that are all the same or missing. Call the reduced dataset *genes-reduced.csv*

Q1.3. Which sample fields should you keep? (1 mark)

Build a J48 model using 10-fold cross validation

Q1.4. Show a diagram of the tree and computed the expected error rate (1 mark)

Q1.5. What are the important variables and their relative importance (according to J48)? (1 mark)

Q1.6. Remove the top predictor and re-run J48. What do you get and why? Show the new tree and error rates. (1 mark)

Q1.7. Based on the results from Q1.6 and Q1.4, do you think the tree that you found with the original data is significant? Justify your answer with a thorough comparison (accuracy/error rates, efficiency at predicting, ROC area, structure simplicity, and why this is better in terms of computational advantages, readability etc.) (2 marks)

Q2. Classifiers: Accuracy (4 marks)

Q2.1. Compare *ZeroR* and *OneR* against J48 on multiple datasets in terms of accuracy (2 marks)

- ○ Start the Experimenter
- ○ Add datasets: iris, breast-cancer, credit-g, diabetes, glass, ionosphere, segmentchallenge
- ○ Add classifiers: J48, ZeroR, OneR (in this order)
- ○ Leave settings as 10-fold cross-validation and 10 repetitions
- ○ DO NOT write the results to a file
- ○ Run the experiment and analyse the results
- ○ Show the results table produced by Weka and discuss how ZeroR and OneR compare against J48 on the different datasets used.

Q2.2. Compare OneR against ZeroR in terms of accuracy (2 marks)

- ○ Go back to the Experimenter
- ○ On the 'Analyse' tab find the 'Test base' option and select OneR
- ○ Now the other two classifiers will be compared against OneR ○ Click 'Perform test'

- ○ Show the results table produced by Weka and discuss how OneR compares against ZeroR on the different datasets

Q3. Classifiers: Training Time Comparison (4 marks)

Generate artificial datasets of different sizes:

- Open Weka GUI chooser - click on Explorer
- Under the 'Preprocess' tab click the generate button.
- Click 'Choose' and select classifiers>classification>LED24.
- Once LED24 is selected click the choose button to configure parameters of the generator.
- In the 'num'Examples' field insert 100000.
- Click 'Generate'. It may take a few seconds to generate the file.
- You have just generated an artificial file for classification with 100K instances.
- Click the 'Save' button and save the file on your disk under the name *led100K.arff*
- Repeat process and generate datasets for 200000, 300000, 400000, 500000 instances with names *led200K.arff, led300K.arff, led400K.arff*, and *led500K.arff* respectively.
- Close the Explorer.

Run the classifiers on all the datasets:

- Start the Experimenter; click 'New'
- For experiment type choose 'Train/Test Percentage Split' (data randomized). DO NOT choose cross-validation otherwise the run time will be prohibitive.
- Choose 1 for the number of repetitions
- Add the algorithms: J48, *NaiveBayesSimple*
- Add the five datasets generated in the previous step
- For the results destination choose *csv* and the name & destination of the output file
- Run the experiment (this may take a few minutes)
- Examine the file with the results

Q3.1. Plot the training time for each classifier (*Elasped_time_training* column from file) against the data size. Explain what you observe and your understanding in terms of training time and data size (include a graph). Consider algorithm implementation and potential stochasticity in running times. (2 marks)

Q3.2. What do you think would happen if we continue increasing the number of instances? Which of the algorithms would be more suitable for a very large number of instances and why? Consider the algorithms' complexity and how they scale. (2 marks)

Q4. Classifiers: Memory Usage Comparison (4 marks)

- Copy the 'J48MemTest.jar' file into the directory that contains the datasets created in Q3. (This 'J48MemTest.jar' file creates a classification model using J48 and measures the memory consumption during that process.)
- Open the terminal and move to the directory containing your datasets.
- Run the following command to get the memory usage for the dataset 'led24_100_000.arff':

  java -jar J48MemTest.jar led24_100_000.arff
- Record the memory usage (in MB)
- Repeat the experiment for the remaining datasets (from 100K-500K)

Q4.1. Plot memory usage of J48 against the data size (i.e. number of instances used). Explain the memory usage (include graph in your answer) (4 marks)

Further Reading: In the "big data" context, an issue is that many algorithms try to fit all data in memory to create the classification model. For example, try to generate 1M instances of

the LED24 dataset and run J48; the algorithm crashes (out of memory – using Weka's default 1GB memory settings). A solution is to use incremental (aka updatable) algorithms that process one instance at a time rather than loading the entire dataset in memory. You can play with this using Weka's command line interface (SimpleCLI) and run the incremental versions of algorithms provided.