COMP6560
Computational Intelligence in Business, Economics and Finance Assignment

Credit Scoring with Genetic Algorithms

Deliverables:	Zip file with project and other files (input data, implementation, and report). The implementation can be in Java or another language of your preference.
Deadline:	**23:55 on Tuesday, November 28 2023 (Week 17)**

Your task is to implement a Genetic Algorithm (GA) to create a credit scoring model, which will consist of a list of classification rules. This will be divided into 3 parts:

- **Modelling**: adjust the input values to be used by the GA.
- **Implementation**: decide on the individual representation and the fitness function; and implement a GA that given a dataset, optimizes a credit scoring model.
- **Report**: complete a set of experiments and compile the results.

For this assessment, you should use the `Credit.arff` dataset (available on Moodle). You will notice that the dataset has 6 numeric and 8 categorical predictor attributes. You will explore an approach to deal with numeric attributes, since these attributes can take many different values and complete the implementation of a GA to create a classification model.

Part A: Modelling [20 marks]

There are generally two approaches to handle numeric value when using GA to create classification rules:

1. Numeric values can be transformed to categorical by defining a set of ranges – e.g., a value `v` can be replaced by a category representing its range: "`v <= 5`", "`5 < v <= 8`", and "`v > 8`".
2. The GA can be modified to handle the numeric values directly.

In this assessment, we will explore the approach (1). As part of the modelling task, you will create a new version of the `Credit.arff` dataset where the numeric values are replaced by corresponding categorical values. To complete this, you will need to:

- Determine the range of values of each numeric attribute.
- Decide on the number of categories (value ranges) to divide the numeric attributes' values. Note that the number of categories can influence the performance of the GA: choosing too few/many might prevent the GA to find good rules.
- Modify the original dataset, replacing numeric attributes by their "categorical" version.

The process is illustrated in the diagram below:

A1	***A2***	**...**
0	3	
1	5	
1	7	
0	10	

A1	***A2***	**...**
0	0	
1	0	
1	1	
0	2	

In this diagram, the value of the numeric attribute A2 is divided into three categories: 0 representing the values “v <= 5”, 1 representing the values “5 < v <= 8” and 2 representing values “v > 8”.

Your submission **must include the implementation** (code) of this process and the transformed dataset, which should have 14 categorical predictor attributes; **no marks** will be awarded otherwise. A spreadsheet can be added as an alternative to coding, but the spreadsheet **must include the formulas** used to calculate the values.

PART B: Implementation [60 marks]

Now that you have your input data prepared, your task is to implement a Genetic Algorithm (GA) to optimize the classification rules created by the GA. You should use as a starting point the skeleton implementation found on the Moodle page.

For your GA, you need to complete:

- **individual initialisation [5%]**: the population should be randomly initialised.

- **fitness function [10%]**: evaluation of each individual (rule). This will consist in calculating the sensitivity and specificity for each rule – more about this below.

- **selection method [5%]**: the GA should use tournament selection.

To calculate the fitness of an individual (rule) in your fitness function, you will need to calculate the sensitivity x specificity for each rule. The equations are given below:

$$sensitivity = \frac{TP}{TP + FN} \qquad specificity = \frac{TN}{FP + TN}$$

- **true positives** (TP): number of data points covered by the rule that have the target value predicted by the rule
- **false positives** (FP): number of data points covered by the rule that have a target value different from the one predicted by the rule
- **false negatives** (FN): number of data points that are not covered by the rule but that have the target value predicted by the rule
- **true negatives** (TN): number of data points that are not covered by the rule and that do not have the target value predicted by the rule

Use the methods `covers(boolean[] rule, boolean[] instance)` and `target(boolean[] encoding)` to determine if a rule is covering a particular instance and if the instance has the same value as the one predicted by the rule.

After completing the steps above, you should have a working version of a GA that creates classification rules.

Extending the rule representation [35%]

While we transformed numeric attributes to be categorical, numeric values have a natural order. For example, values on the category "`v <= 5`" are smaller than values on the category "`5 < v <= 7`". At the moment, the GA implicitly uses a "`=`" (equal) relational operator.

Your task is to extend the GA individual representation and add one bit to represent the relational operator to be used: `0` represents the operator `<` and `1` represents the operator `>`. For simplicity, this extra bit should only be added to categorical attributes derived from numeric attributes. This will allow the GA to create rules that includes conditions such as "`A2 < 2`". You will need to modify how rules are evaluated to take into consideration that attributes can have different relational operators – e.g., the `covers(boolean[] rule, boolean[] instance)` method.

PART C: Report [20 marks]

Prepare a 2-pages report detailing your implementation and results. Topics you should cover:

- How much you managed to achieve in terms of the given tasks.
- Difficulties during your implementation: what has gone well, what has gone wrong.
- Report on your experimental results, including summary statistics from multiple runs. If you haven't obtained any results yet, that is fine, but you should still mention that you don't have any results and describe what has been completed.
- Report on different GA parameters that you might have used, and how/if they affected the performance of the algorithm.
- Anything else you consider useful to mention, e.g., any additional methods implementation you decided to implement.
- Would you do anything differently if you had to re-do the assignment.

Note: The page limit is **2 pages**. Anything beyond 2 pages won't be considered.

Submission

Electronic version (zip file) of project and other files (input data, implementation, and report). The zip file should be submitted via Moodle before the deadline, no later than **23:55 on Tuesday, November 28 2023 (Week 17)**. Any other form of submission will not be accepted. Note that the submission link will not be available after the deadline.