Homework 9

DS-210 @ Boston University

Before you start. . .

Collaboration policy: You may verbally collaborate on required homework problems. However, you must write your solutions independently without showing them to other students. If you choose to collaborate on a problem, you are allowed to discuss it with at most 2 other students currently enrolled in the class.

The header of each assignment you submit must include the field “Collaborators:” with the names of the students with whom you have had discussions concerning your solutions. If you didn’t collaborate with anyone, write “Collaborators: none.“ A failure to list collaborators may result in a credit deduction.

You may use external resources such as software documentation, textbooks, lecture notes, and videos to supplement your general understanding of the course topics. You may use references such as books and online resources for well known facts. However, you must always cite the source.

You may not look up answers to a homework assignment in the published literature or on the web. You may not share written work with anyone else.

Submitting: Solutions should be submitted via Gradescope.

Grading: Whenever we ask for a solution, you may receive partial credit if your solution is not sufficiently efficient or close to optimal. For instance, if we ask you to solve a specific problem that has a polynomial– time algorithm that is easy to implement, but the solution you provide is exponentially slower, you are likely to receive partial credit.

Explaining: Always explain your work clearly in your writeup, even if it is correct. A good explanation can help you get points back if there are mistakes in your code. A missing or bad explanation can result in points being deducted.

Questions

To solve problems in this homework, you should use Rust. Your solution to the homework should consist of a zip file containing

- a compilable Rust source file (.rs) solving Question 1,
- a report (the pdf format is recommended) that answers sub-questions denoted by “Report” below.

1. (40 points) Input: Your program should read a set of points in R with labels from data.txt (you will create file data.txt). Each line of data.txt describes one point. More specifically, the i-th line consists of two numbers x_i and z_i, where x_i is an integer s.t. $|x_i| \leq 100{,}000{,}000$ and $z_i \in \{0, 1\}$. x_i is the coordinate of the point and z_i is its label. Make sure your file contains at least 100 points randomly distributed in the allowed space with labels

assigned to each point in a random fashion as well.

Your task: Write a program that reads the data and determines a decision tree with at most two leaves that performs best at predicting z_i based on x_i on this set of points. The program should output the decision tree and its accuracy on the input data set (no need to split on a training and testing set though you are welcome to do so if you want).
Report: Explain why your solution works and what complexity it has as a function of the number of points (denote this quantity by n).
Sample input and output: For the following data.txt:

```
-15 0
 15 1
 -5 1
  5 0
```

the following output is a correct solution:

```
if x >= 10
   output 1
else
  output 0
```

accuracy: 0.75
Hint: You do not need to implement the whole gini logic here. Your input is 1-dimensional so you need to find a split point that minimizes error.

2. (Optional, no credit) Report: How much time did you spend on this homework? The answer will have no impact on the credit you receive, but it may help us adjust the difficulty of future homework assignments.
3. (Optional, no credit)
   Report and/or code: How can you solve Question 1 when more than two leaves are allowed? What is the complexity of your solution as a function of the numbers of leaves and points?
   Hint: Once you have decided on a split there is no backtracking in decision tree logic. All you can do is decide how to split the two sides further.