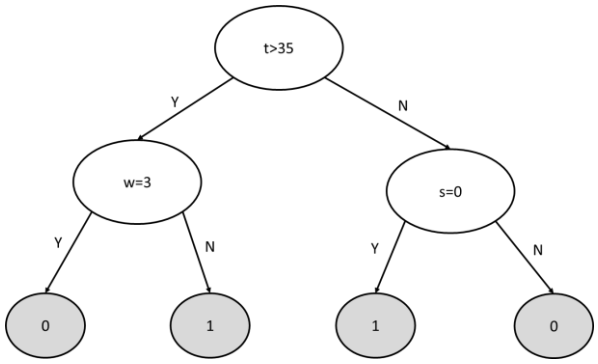
CSE102 – Computer Programming with C
Assignment #2

In this assignment, you will learn how to implement decision trees in C and compare their performances.

A decision tree is a simple model for classification and regression in machine learning. It gives the steps of a set of comparisons to come up with a a decision on a given input. For example, if you want to build a system to automatically turn on the heater in a room based on several inputs, you can use the following decision tree.



Given *temperature(t)*, *pressure(p)*, *humidity(h)*, *sunny_or_not(s)* and *day_of_the_week(w)* as your input, this decision tree answers wheter or not to turn on the AC (0=turn off, 1=turn on).

Complete the code in the given project with the following tasks:

1. main.c: In the main function you will be solving three problems using two decision trees for each. Your program will ask which problem to solve (1, 2 or 3). Based on the answer, user's inputs will be read (as many as required with the right types). The input will be processed by the two alternative decision trees (you will implement these in functions as described below). The results will be compared and the final result will be printed. In the case of classifier (output is a number from a set of possibilities), if both answers are the same, that result will be given as the answer to the problem. If the results differ, both decisions will be reported. Similarly, for regression (the output is an ordinal number), if the two results are similar (within a threshold, defined as the constant CLOSE_ENOUGH in util.h), the average will be printed otherwise both results will be output.
2. util.c (and util.h): There are six functions in this file. Each pair of functions are solutions to the same problem. First four decision trees are given in the figures below. The last two are supposed to be designed by you in the following manner:
    a. The decision trees will solve a classification problem.
    b. It should have at least 10 decision nodes.
    c. The input should be 5 dimensional with two real numbers and 3 categorical (one binary, the others with more than 5 possible values).
    d. Each of these inputs should at least once used in the decision nodes.
    e. Provide two significantly different such trees and implement them in dt3a and dt3b.

General Rules:

- Obey the style guidelines.
- Do not change the provided function prototypes (you will not get any credits).
- The program must be developed on Ubuntu using GCC compiler (version provided in class), compilation problems due to the use of another OS or compiler is your responsibility (you will not get any credits).
- Your program should work as expected. Do not expect partial credit if your code works only in some cases but not in all cases as expected.
- You can ask your questions about the homework by positing on the forum in Teams.

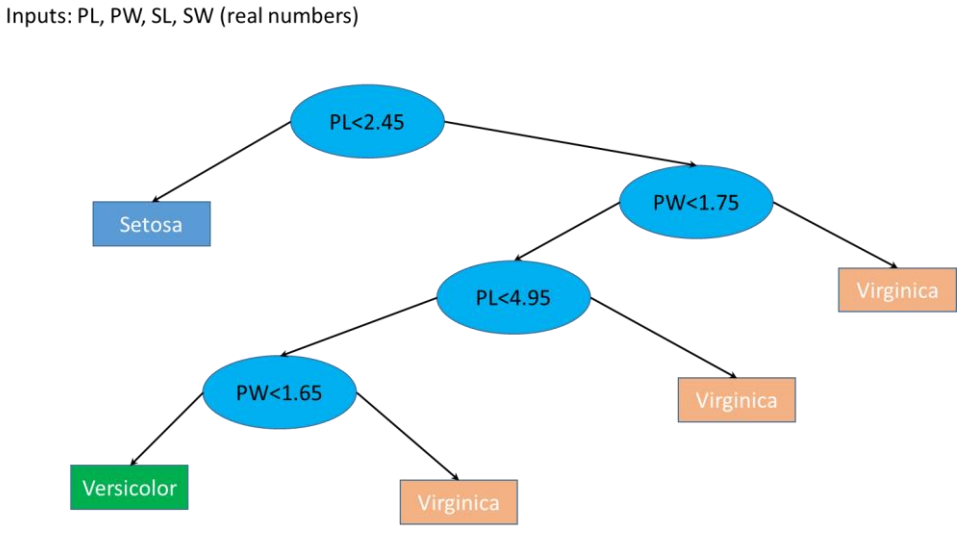
Handing in your work:

- Hand in your work using the appropriate class Teams assignment site.
- No late submissions will be accepted.
- Please pack your solution directory in the following way (assuming a student with number 20180000001 and name X Y Z is submitting):
  - A directory named 20180000001_X_Z is created
  - All the solutions files along with a make file are created as part of the assignment. For example:

```
ygenc@YG-XPS:~/cse102-ubuntu/20180000001_X_Z$ ls -l
total 0
-rwxrwxrwx 1 ygenc ygenc 86 Feb 17 10:14 main.c
-rwxrwxrwx 1 ygenc ygenc 39 Feb 17 10:16 makefile
-rwxrwxrwx 1 ygenc ygenc 19 Feb 17 10:17 util.c
-rwxrwxrwx 1 ygenc ygenc 58 Feb 17 10:15 util.h
ygenc@YG-XPS:~/cse102-ubuntu/20180000001_X_Z$
```

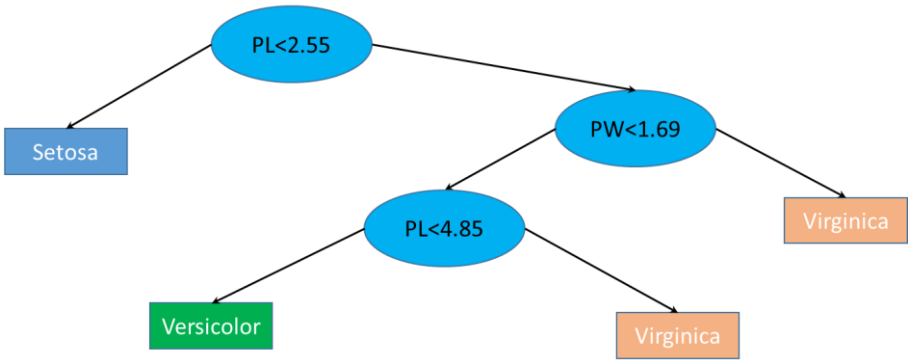
  - Pack this directory into a zip file named 20180000001_X_Z.zip
  - When unpacked as above in Ubuntu (version provided in class) it should allow executing the following commands in a shell:
    - "$make clean" removes everything except makefile, source code (.c and .h) and other resource files (if any) – all compiling results and intermediate files should be removed.
    - "$make compile" should compile the code.
    - "$make run" should run the code along with any parameters needed.

The first decision tree for P1:



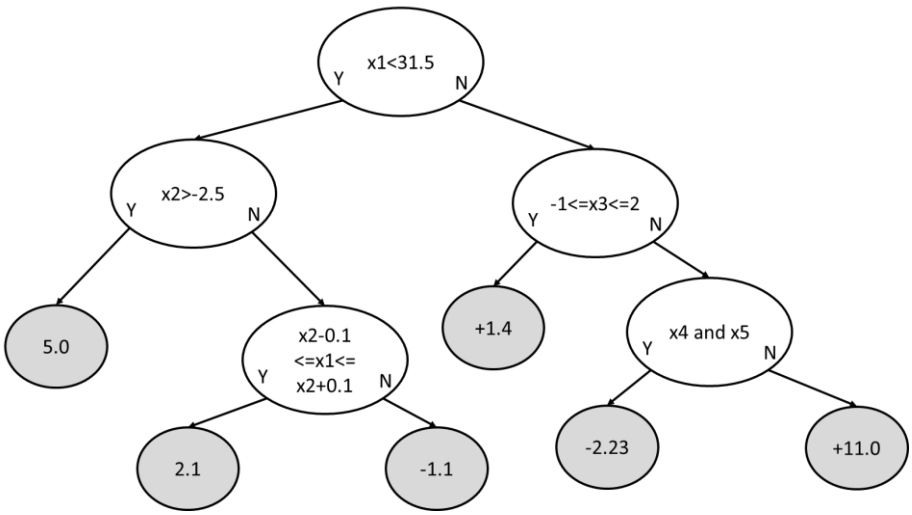
The second decision tree for P1:

Inputs: PL, PW, SL, SW (real numbers)



The first decision trees for P2:

Inputs: x1, x2, x3 (real numbers) and x4, x5 (binary values)



The second decision trees for P2:

Inputs: x1, x2, x3 (real numbers) and x4, x5 (binary values)