Explanation and Observation in the TIC TAC TOE problem.

The given data set is about the moves which would result in a **POSITIVE** win for the player playing the **X** move. The different classes in the decision tree are the cells in the TIC-TAC-TOE game, and each cell can be in three states. 'x', 'o', or blank represented by 'b'

So a given cell can be represented by

CELL-1	C1-x	C1-b	C1-o

So a total of 9 cells with three states each, that is 27 Cellular automata cells.

One of the major mistakes I had been doing in taking the test data was ignoring the fact that in a decision tree, for a given class the states are mutually exclusive in nature. That is cell-1 can be only in any one of the three states at any given time frame. Currently the cellular automata supports only discreet values, that is for example a cell state is represented by setting the corresponding x, b, or o states on or off, and this state has to be mutually exclusive.

I did a little experiment with continuous values. Here is an explanation of the experiment, which I plan on doing as future extension of the current problem.

Number of classes : c1, c2, c3, c4 ,c5, c6, c7, c8, c9
Total number of Cellular automata cells = 9 (instead of the previous 27).
The states are specified with values 1 (x) , 1.5 (b), 2 (o)

Continuing on the same principle as the previous approach, that is setting the threshold for deciding the classes as the mean between two decision classes (In this case 0.5 for decision classes **POSITIVE (**1**)** and **NEGATIVE (**0**)**), and adjusting the weights for each input classes based on the deviation from the threshold, I came across a curious result. (For test2.dat file)

```
-1.949999 -ve
-1.949999 -ve
-1.949999 -ve
-1.949999 -ve
-1.949999 -ve
-1.949999 -ve
-2.024999 -ve
-2.024999 -ve
-2.024999 -ve
-2.024999 -ve
-2.024999 -ve
-2.025000 -ve
-1.949999 -ve
-1.949999 -ve
-1.949999 -ve
```

As you can see the values are in fact classified into two classes, -1.9 and -2.0 and after crosschecking with the test data, these classification are based on the expected results. That is -1.9 denotes cases where the result is a **POSITIVE** win for **X** and -2.0 is a **NEGATIVE** win for **X.** The

reason why the output is marked as -ve is because all these values fall below the threshold for deciding classes that is 0.5 and thus tends out to the **NEGATIVE** win for **X**. This is something I wish to continue as a future work.

The rest of the explanation of the program and approach is same as the one stated in the previous report I send you.