# HOMEWORK 2

>>Tingyou Guo<<
>>9077313519<<

**Instructions:** Although this is a programming homework, you only need to hand in a pdf answer file. There is no need to submit the latex source or any code. You can choose any programming language, as long as you implement the algorithm from scratch (e.g. do not use Weka on questions 1 to 7).
Use this latex file as a template to develop your homework. Submit your homework on time as a single pdf file to Canvas. Please check Piazza for updates about the homework.

## 1 A Simplified Decision Tree

You are to implement a decision-tree learner for classification. To simplify your work, this will not be a general purpose decision tree. Instead, your program can assume that

- each item has two continuous features $\mathbf{x} \in \mathbb{R}^2$
- the class label is binary and encoded as $y \in \{0, 1\}$
- data files are in plaintext with one labeled item per line, separated by whitespace:

$$x_{11} \quad x_{12} \quad y_1$$
$$\ldots$$
$$x_{n1} \quad x_{n2} \quad y_n$$

Your program should implement a decision tree learner according to the following guidelines:

- Candidate splits $(j, c)$ for numeric features should use a threshold $c$ in feature dimension $j$ in the form of $x_{\cdot j} \geq c$.
- $c$ should be on values of that dimension present in the training data; i.e. the threshold is on training points, not in between training points.
- The left branch of such a split is the "then" branch, and the right branch is "else".
- Splits should be chosen using mutual information (i.e. information gain). If there is a tie you may break it arbitrarily.
- The stopping criteria (for making a node into a leaf) are that
  - the node is empty, or
  - all splits have zero mutual information
- To simplify, whenever there is no majority class in a leaf, let it predict $y = 1$.

## 2 Questions

1. (Our algorithm stops at pure labels) [10 pts] If a node is not empty but contains training items with the same label, why is it guaranteed to become a leaf? Explain.

   It is guaranteed to be a leaf because all splits have zero mutual information.

   $$MI(Y, \theta) = H(Y) - H(Y|\theta) \tag{1}$$

As it contains the training items with the same label,

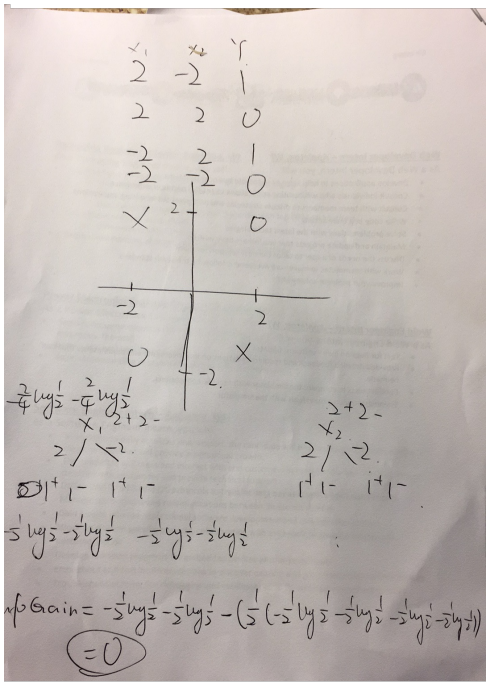
$$H(Y) = -1log_2 1 = 0; \tag{2}$$

Assume the split is $(j, \theta)$,

$$\begin{aligned} H(Y|\theta_{right}) &= -1log1 = 0, && (3)\\ H(Y|\theta_{left}) &= -1log1 = 0, && (4)\\ H(Y|\theta) &= P(\theta_{left}) * H(Y|\theta_{left}) + P(\theta_{right}) * H(Y|\theta_{right}) && (5)\\ &= P(\theta_{left}) * 0 + P(\theta_{right}) * 0 = 0, && (6)\\ MI(Y, \theta) &= H(Y) - H(Y|\theta) = 0 - 0 = 0 && (7)\\ & && (8) \end{aligned}$$

As $(j, \theta)$ is arbitrary, so all splits have zero mutual information, thus it is guaranteed to be a leaf.

2. (Our algorithm is greedy) [10 pts] Handcraft a small training set where both classes are present but the algorithm refuses to split; instead it makes the root a leaf and stop; Importantly, if we were to manually force a split, the algorithm will happily continue splitting the data set further and produce a deeper tree with zero training error. You should (1) plot your training set, (2) explain why. Hint: you don't need more than a handful of items.



Because all the information gain is 0

3. (Mutual information exercise) [10 pts] Use the training set Druns.txt. For the root node, list all candidate cuts and their mutual information. Hint: to get $\log_2(x)$ when your programming language may be using a different base, use `log(x)/log(2)`.

   If we use x1=0 as cuts, the information gain is 0.04417739186726144.
   When choosing x2:

Info gain at split -2 is 0.0
Info gain at split -1 is 0.04417739186726133
Info gain at split 0 is 0.03827452220629246
Info gain at split 1 is 0.004886164091842726
Info gain at split 2 is 0.0010821659130775263
Info gain at split 3 is 0.016313165825732057
Info gain at split 4 is 0.04945207278939401
Info gain at split 5 is 0.10519553207004628
Info gain at split 6 is 0.19958702318968735
Info gain at split 7 is 0.03827452220629246
Info gain at split 8 is 0.18905266854301617
So we use x2 as cuttings where 5 as threshold the information gain is 0.19958702318968735 which is the most. so we should split x2.

4. (The king of interpretability) [10 pts] Decision tree is not the most accurate classifier in general. However, it persists. This is largely due to its rumored interpretability: a data scientist can easily explain a tree to a non-data scientist. Build a tree from D3leaves.txt. Then manually convert your tree to a set of logic rules. Show the tree[1] and the rules.

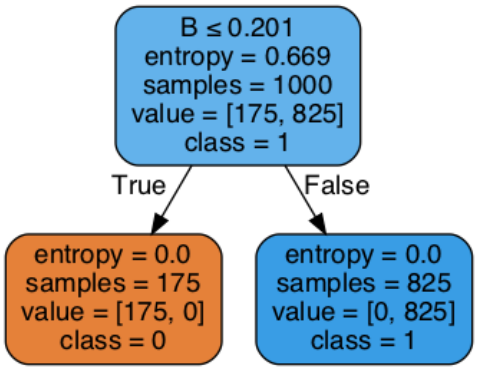
   If I use x1=10 the infogain is 0.4453509366224364
   Info gain at split 1 is 0.12342284173507412
   Info gain at split 10 is 0.4453509366224364
   If I use x2= 2the infogain is 0.4453509366224364
   Info gain at split 1 is 0.12342284173507412
   Info gain at split 2 is 0.4453509366224364
   Info gain at split 3 is 0.29437343618974277 MY decision tree

```
In [200]: print_tree(my_tree)
Is diameter >= 2.0?
--> True:
  Predict {1.0: 3}
--> False:
  Is color >= 10.0?
  --> True:
    Predict {1.0: 1}
  --> False:
    Predict {0.0: 1}
```

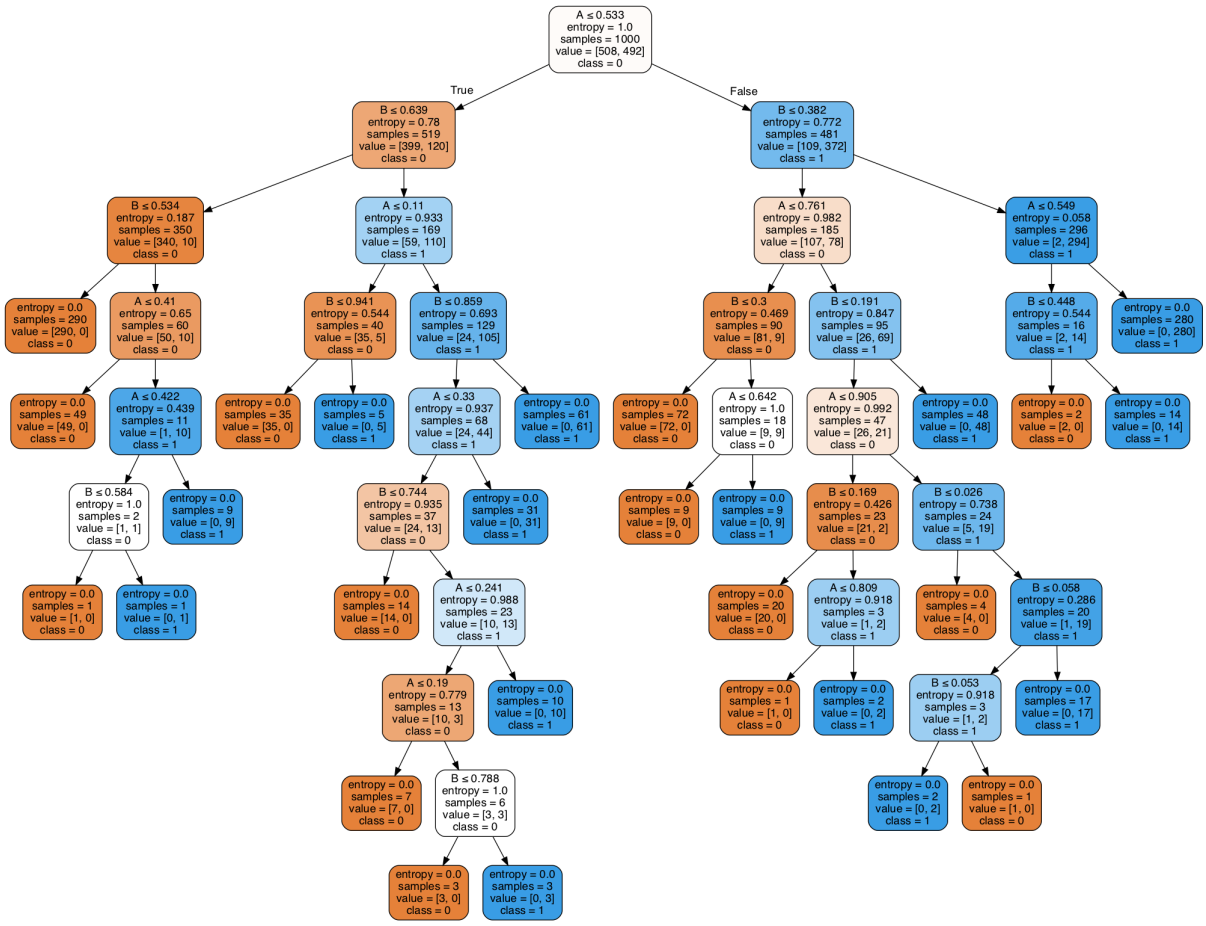
5. (Or is it?) [20 pts] For this question only, make sure you DO NOT VISUALIZE the data sets or plot your tree's decision boundary in the 2D $\mathbf{x}$ space. If your code does that, turn it off before proceeding. This is because you want to see your own reaction when trying to interpret a tree. You will get points no matter what your interpretation is. And we will ask you to visualize them in the next question anyway.

   - Build a decision tree on D1.txt. Show it to us in any format (e.g. could be a standard binary tree with nodes and arrows, and denote the rule at each leaf node; or Weka style plaintext tree; or as simple as plaintext output where each line represents a node with appropriate line number pointers to child nodes; whatever is convenient for you). Again, do not visualize the data set or the tree in the $\mathbf{x}$ input space. In real tasks you will not be able to visualize the whole high dimensional input space anyway, so we don't want you to "cheat" here.
   - Look at your tree in the above format (remember, you should not visualize the 2D dataset or your tree's decision boundary) and try to interpret the decision boundary in human understandable English. D1

[1] When we say show the tree, we mean either the standard computer science tree view, or some crude plaintext representation of the tree – as long as you explain the format. When we say visualize the tree, we mean a plot in the 2D $\mathbf{x}$ space that shows how the tree will classify any points.




if the second column x2 is less than 0.201, then we can classify it to 0.
D2



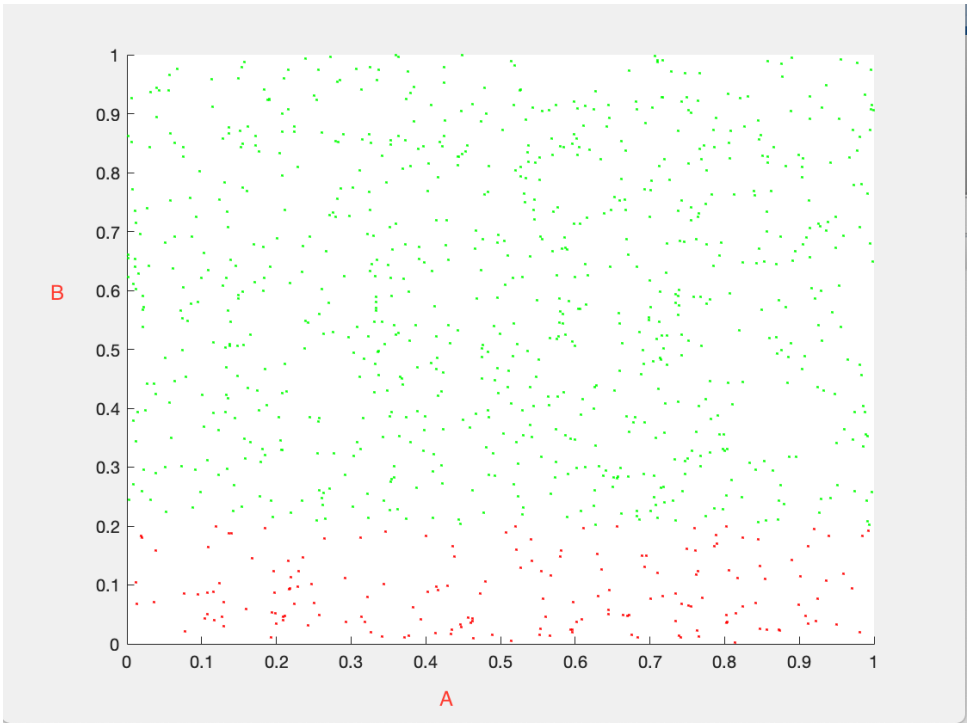

- Build a decision tree on D2.txt. Show it to us.
- Try to interpret your D2 decision tree.

First we decide if A is larger than 0.533, if yes, then we decide if B is larger than 0.382, if yes then we continue decide if A is larger than 0.549.There are lots of things to be decided .

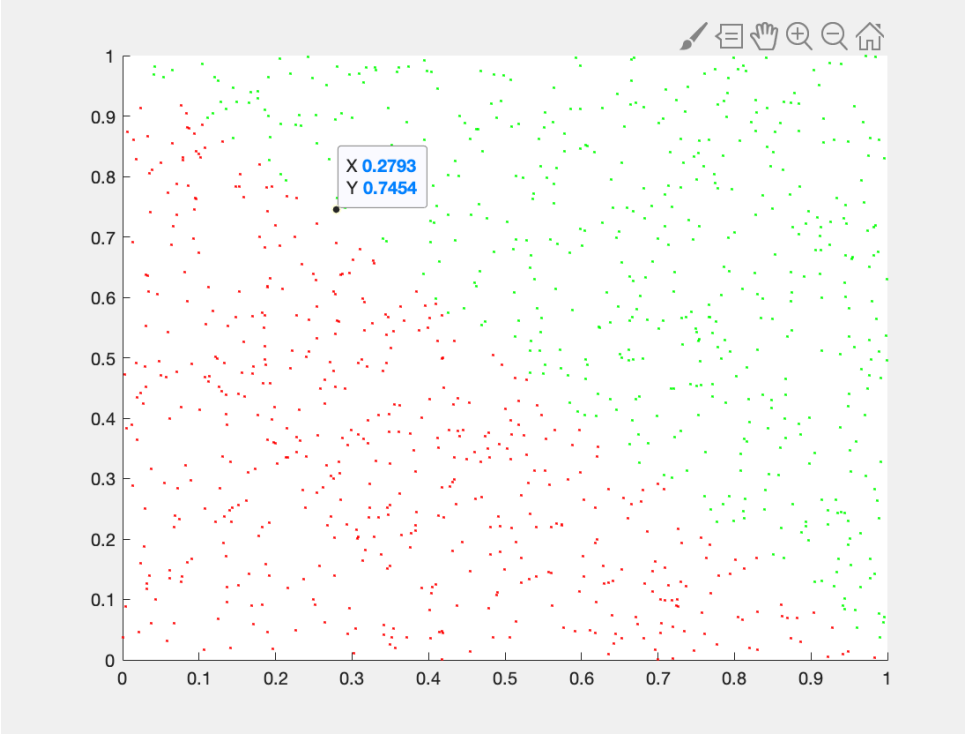
(I was trying very hard to make the left hand as larger, but it turns out to be less, I feel really sorry !! but in my code it is always the left as larger !)

6. (Hypothesis space) [10 pts] For D1.txt and D2.txt, do the following separately:

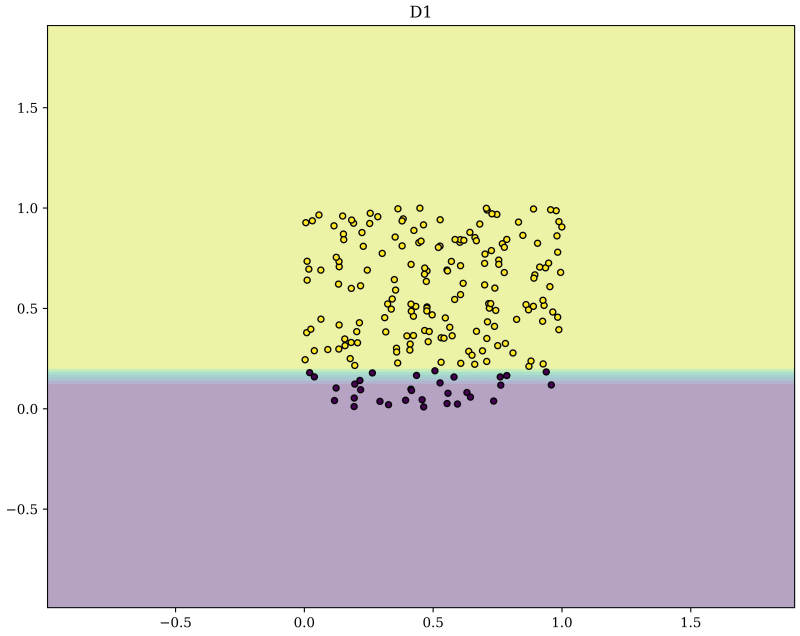
   - Produce a scatter plot of the data set.
     D1's plot

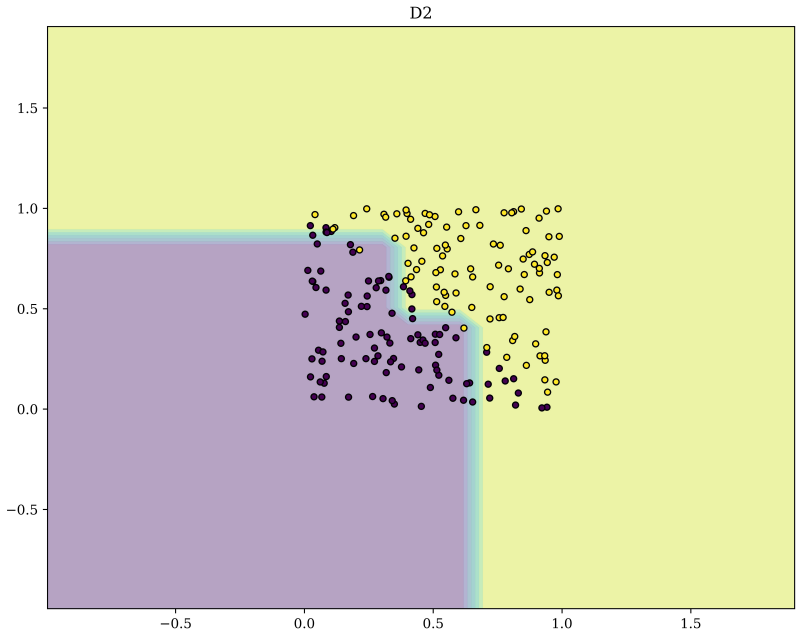



D2's plot



- Visualize your decision tree's decision boundary (or decision region, or some other ways to clearly visualize how your decision tree will make decisions in the feature space).



| $n$ | $err_n$ |
|---|---|
| 32 | 0.195927 |
| 128 | 0.155389 |
| 512 | 0.168002 |
| 2048 | 0.164487 |
| 8192 | 0.172013 |

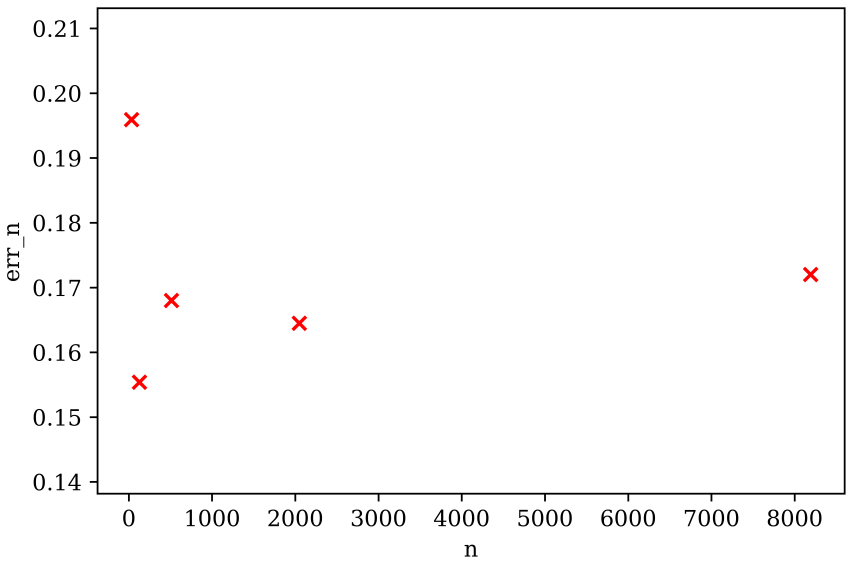
Then discuss why the size of your decision trees on D1 and D2 differ. Relate this to the hypothesis space of our decision tree algorithm.
Because we are using decision tree and we have to set threshold, it is a rude hypothesis. So the points around threshold will be difficult to define so it.

7. (Learning curve) [20 pts] We provide a data set Dbig.txt with 10000 labeled items. Caution: Dbig.txt is sorted.

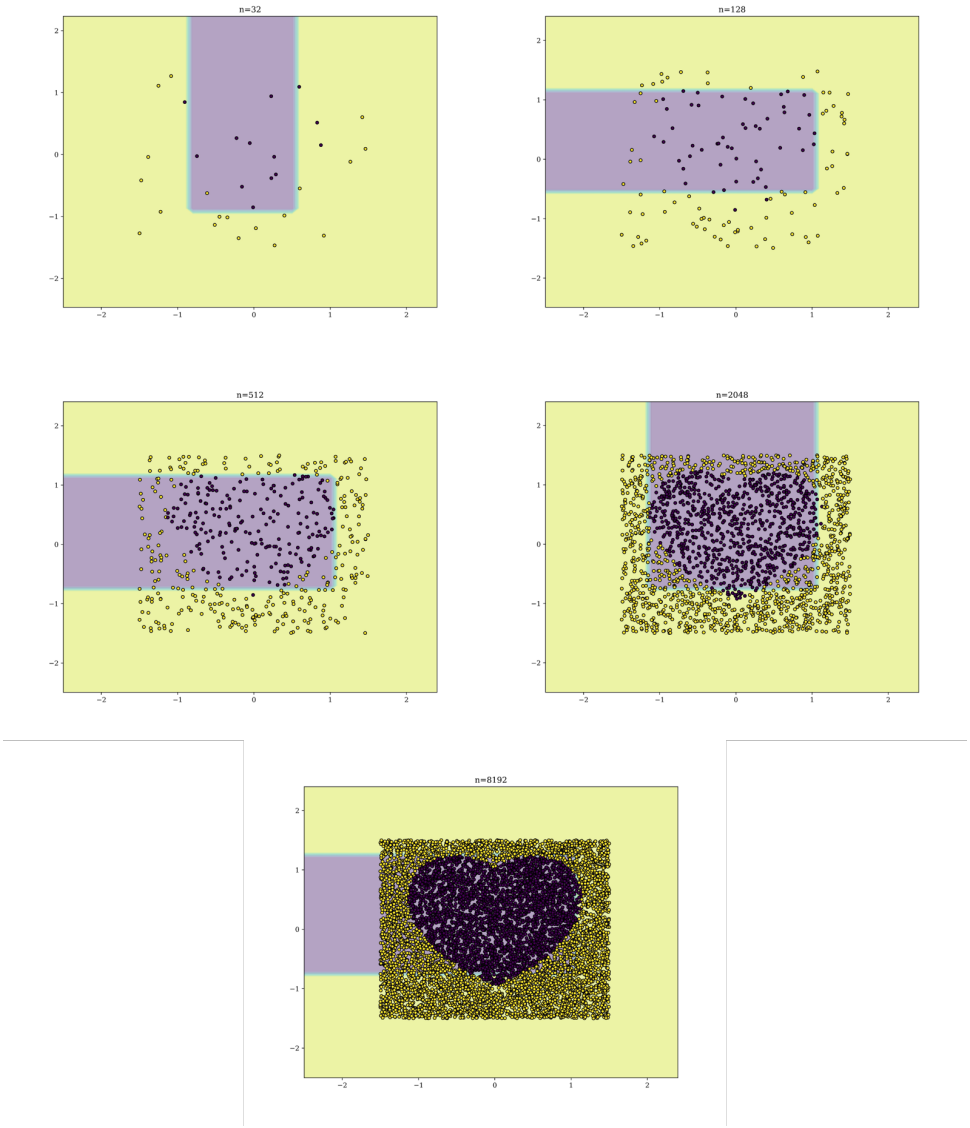
   - You will randomly split Dbig.txt into a candidate training set of 8192 items and a test set (the rest). Do this by generating a random permutation, and split at 8192.
   - Generate a sequence of five nested training sets $D_{32} \subset D_{128} \subset D_{512} \subset D_{2048} \subset D_{8192}$ from the candidate training set. The subscript $n$ in $D_n$ denotes training set size. The easiest way is to take the first $n$ items from the (same) permutation above. This sequence simulates the real world situation where you obtain more and more training data.
   - For each $D_n$ above, train a decision tree. Measure its test set error $err_n$. Show three things in your answer: (1) List $n$, number of nodes in that tree, $err_n$. (2) Plot $n$ vs. $err_n$. This is known as a learning curve (a single plot). (3) Visualize your decision trees' decision boundary (five plots).

   The list of $n$ and $err_n$ is shown above

   The plot between $n$ and $err_n$ is below




   The boundary of decision trees with different sets of training is shown below




## 3 Weka [10 pts]

Learn to use Weka https://www.cs.waikato.ac.nz/~ml/weka/index.html. Convert appropriate data files into ARFF format. Use trees/J48 as the classifier and default settings. Produce five Weka trees for $D_{32}, D_{128} \ldots D_{8192}$. Show two things in your answer: (1) List $n$, number of nodes in that tree, $err_n$. (2) Plot $n$ vs. $err_n$.

| $n$ | Nodes | $err_n$ |
|---|---|---|
| 32 | 165 | 0.3741 |
| 128 | 165 | 0.2278 |
| 512 | 165 | 0.1977 |
| 2048 | 165 | 0.1308 |
| 8192 | 167 | 0.093 |