HOMEWORK 2

>>Zhiqi Gao<<
>>zgao93 / 9081156037<<

Instructions: Use this latex file as a template to develop your homework. Submit your homework on time as a single pdf file to Canvas. Please wrap your code and upload to a public GitHub repo, then attach the link below the instructions so that we can access it. You can choose any programming language (i.e. python, R, or MATLAB), as long as you implement the algorithm from scratch (e.g. do not use sklearn on questions 1 to 7 in section 2). Please check Piazza for updates about the homework.

Here is a link to my CS760 GitHub repository:
https://github.com/ZhiqiGao-Leo/CS_760_Machine_Learning/blob/main/HW2/DecisionTree.py

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:

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

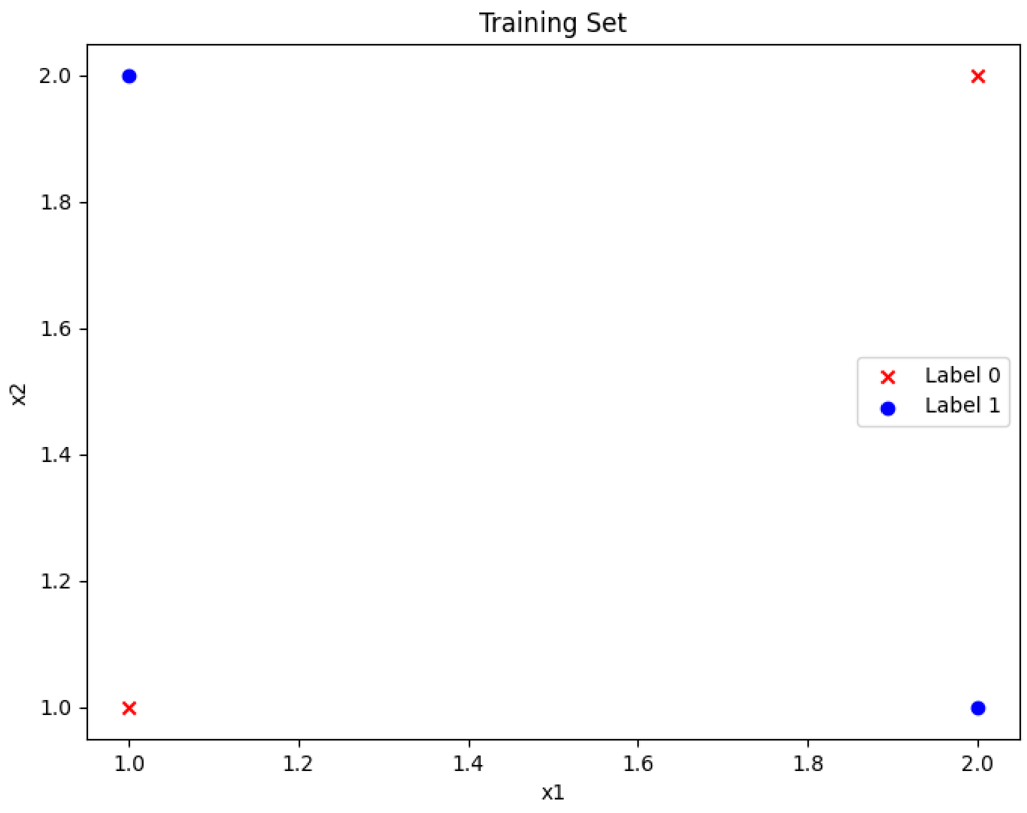
Your program should implement a decision tree learner according to the following guidelines:

- Candidate splits (j, c) for numeric features should use a threshold c in feature dimension j in the form of $x_j \geq c$.
- 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. You may enumerate all features, and for each feature, use all possible values for that dimension.
- You may skip those candidate splits with zero split information (i.e. the entropy of the split), and continue the enumeration.
- The left branch of such a split is the "then" branch, and the right branch is "else".
- Splits should be chosen using information gain ratio. 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 gain ratio (if the entropy of the split is non-zero), or
  - the entropy of any candidates split is zero
- 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. You may assume that the feature values of these items are not all the same.
   If a nonempty node contains items with the same label. We can see that the entropy of any candidate split is Zero by the characteristic of this function $H(X) = -\sum p(x)\log_2(p(x))$. which will meet one of our stopping criteria and guaranteed to become a leaf.
   Note: in the implementation, I manually check if the current node only contains one label and split it to save running time

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.



   In the given training set, the algorithm will refuse to split. The reason is all possible splits will have a zero gain ratio and reach the stopping criteria. I.E., every split will have a set with 50% points with label 1 and 50% points with label 0.

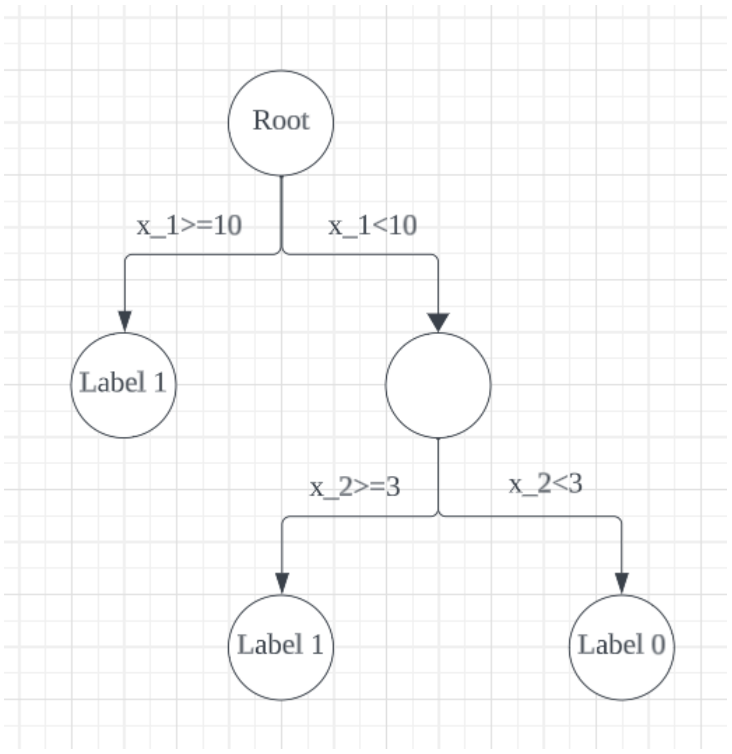
3. (Information gain ratio exercise) [10 pts] Use the training set Druns.txt. For the root node, list all candidate cuts and their information gain ratio. If the entropy of the candidate split is zero, please list its mutual information (i.e. information gain). Hint: to get $\log_2(x)$ when your programming language may be using a

different base, use `log(x)/log(2)`. Also, please follow the split rule in the first section.

Candidate Cut	Information Gain	Gain Ratio
$x1 \geq 0.0$	0.0	-0.0
$x1 \geq 0.1$	0.04417739185414593	0.10051807679636868
$x2 \geq -2.0$	0.0	-0.0
$x2 \geq -1.0$	0.044177391854145945	0.1005180767963687
$x2 \geq 0.0$	0.03827452220629268	0.055953759654866105
$x2 \geq 1.0$	0.004886164091842837	0.005780042207125321
$x2 \geq 2.0$	0.0010821659130776373	0.0011443495176260297
$x2 \geq 3.0$	0.016313165825732168	0.016411136846865940
$x2 \geq 4.0$	0.04945207278939401	0.0497490641962193
$x2 \geq 5.0$	0.10519553207004634	0.11124029589733961
$x2 \geq 6.0$	0.19958702318968752	0.23609960622419476
$x2 \geq 7.0$	0.03827452220629268	0.055953759654866105
$x2 \geq 8.0$	0.18905266852990077	0.43015691638354603

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.

Tree: Yellow Region: Label 1, Blue Region: Lable 0



Rules:

$$\text{If } x_1 \geq 10.0\text{, then Label 1}$$
$$\text{If } x_1 < 10.0 \text{ and } x_2 \geq 3.0\text{, then Label 1}$$
$$\text{If } x_1 < 10.0 \text{ and } x_2 < 3.0\text{, then Label 0}$$

[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.

5. (Or is it?) [10 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 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.

$$\text{If } x_2 \geq 0.201829, \text{ then Label 1}$$
$$\text{If } x_2 < 0.201829, \text{ then Label 0}$$

   - 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.
     Classify all points based on statement $x_2 \geq 0.201829$, label 1 if hold and 0 otherwise

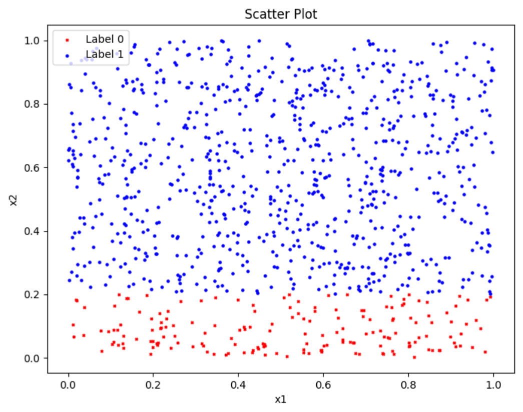
   - Build a decision tree on D2.txt. Show it to us.
     If $x_1 \geq 0.533076$ and $x_2 \geq 0.383738$ and $x_1 \geq 0.550364$, then Label 1
     If $x_1 \geq 0.533076$ and $x_2 \geq 0.383738$ and $x_1 < 0.550364$ and $x_2 \geq 0.474971$, then Label 1
     If $x_1 \geq 0.533076$ and $x_2 \geq 0.383738$ and $x_1 < 0.550364$ and $x_2 < 0.474971$, then Label 0
     If $x_1 \geq 0.533076$ and $x_2 < 0.383738$ and $x_1 \geq 0.761423$ and $x_2 \geq 0.191206$, then Label 1
     If $x_1 \geq 0.533076$ and $x_2 < 0.383738$ and $x_1 \geq 0.761423$ and $x_2 < 0.191206$ and $x_1 \geq 0.90482$ and $x_2 \geq 0.037708$ and $x_1 \geq 0.930371$, then Label 1
     If $x_1 \geq 0.533076$ and $x_2 < 0.383738$ and $x_1 \geq 0.761423$ and $x_2 < 0.191206$ and $x_1 \geq 0.90482$ and $x_2 \geq 0.037708$ and $x_1 < 0.930371$ and $x_1 \geq 0.927522$, then Label 0
     If $x_1 \geq 0.533076$ and $x_2 < 0.383738$ and $x_1 \geq 0.761423$ and $x_2 < 0.191206$ and $x_1 \geq 0.90482$ and $x_2 \geq 0.037708$ and $x_1 < 0.930371$ and $x_1 < 0.927522$, then Label 1
     If $x_1 \geq 0.533076$ and $x_2 < 0.383738$ and $x_1 \geq 0.761423$ and $x_2 < 0.191206$ and $x_1 \geq 0.90482$ and $x_2 < 0.037708$, then Label 0
     If $x_1 \geq 0.533076$ and $x_2 < 0.383738$ and $x_1 < 0.761423$ and $x_2 \geq 0.301105$ and $x_1 \geq 0.66337$, then Label 1
     If $x_1 \geq 0.533076$ and $x_2 < 0.383738$ and $x_1 < 0.761423$ and $x_2 \geq 0.301105$ and $x_1 < 0.66337$, then Label 0
     If $x_1 \geq 0.533076$ and $x_2 < 0.383738$ and $x_1 < 0.761423$ and $x_2 < 0.301105$, then Label 0
     If $x_1 < 0.533076$ and $x_2 \geq 0.639018$ and $x_1 \geq 0.111076$ and $x_2 \geq 0.861$, then Label 1
     If $x_1 < 0.533076$ and $x_2 \geq 0.639018$ and $x_1 \geq 0.111076$ and $x_2 < 0.861$ and $x_1 \geq 0.33046$, then Label 1
     If $x_1 < 0.533076$ and $x_2 \geq 0.639018$ and $x_1 \geq 0.111076$ and $x_2 < 0.861$ and $x_1 < 0.33046$ and $x_2 \geq 0.745406$ and $x_1 \geq 0.254049$, then Label 1
     If $x_1 < 0.533076$ and $x_2 \geq 0.639018$ and $x_1 \geq 0.111076$ and $x_2 < 0.861$ and $x_1 < 0.33046$ and $x_2 \geq 0.745406$ and $x_1 < 0.254049$ and $x_1 \geq 0.191915$ and $x_2 \geq 0.792752$, then Label 1
     If $x_1 < 0.533076$ and $x_2 \geq 0.639018$ and $x_1 \geq 0.111076$ and $x_2 < 0.861$ and $x_1 < 0.33046$ and $x_2 \geq 0.745406$ and $x_1 < 0.254049$ and $x_1 \geq 0.191915$ and $x_2 < 0.792752$, then Label 0
     If $x_1 < 0.533076$ and $x_2 \geq 0.639018$ and $x_1 \geq 0.111076$ and $x_2 < 0.861$ and $x_1 < 0.33046$ and $x_2 \geq 0.745406$ and $x_1 < 0.254049$ and $x_1 < 0.191915$, then Label 0
     If $x_1 < 0.533076$ and $x_2 \geq 0.639018$ and $x_1 \geq 0.111076$ and $x_2 < 0.861$ and $x_1 < 0.33046$ and $x_2 < 0.745406$, then Label 0
     If $x_1 < 0.533076$ and $x_2 \geq 0.639018$ and $x_1 < 0.111076$ and $x_2 \geq 0.964767$, then Label 1
     If $x_1 < 0.533076$ and $x_2 \geq 0.639018$ and $x_1 < 0.111076$ and $x_2 < 0.964767$, then Label 0
     If $x_1 < 0.533076$ and $x_2 < 0.639018$ and $x_2 \geq 0.534979$ and $x_1 \geq 0.409972$ and $x_1 \geq 0.426073$, then Label 1
     If $x_1 < 0.533076$ and $x_2 < 0.639018$ and $x_2 \geq 0.534979$ and $x_1 \geq 0.409972$ and $x_1 < 0.426073$ and $x_1 \geq 0.417579$, then Label 0
     If $x_1 < 0.533076$ and $x_2 < 0.639018$ and $x_2 \geq 0.534979$ and $x_1 \geq 0.409972$ and $x_1 < 0.426073$

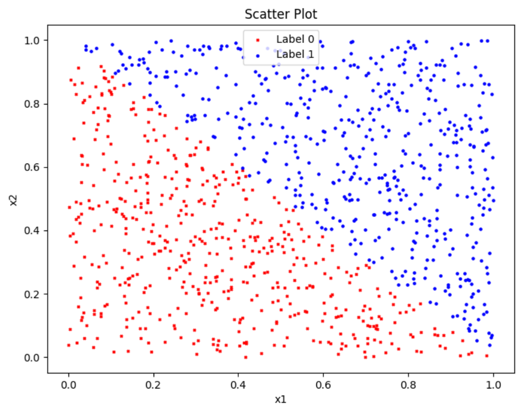
and $x_1 < 0.417579$, then Label 1
If $x_1 < 0.533076$ and $x_2 < 0.639018$ and $x_2 \geq 0.534979$ and $x_1 < 0.409972$, then Label 0
If $x_1 < 0.533076$ and $x_2 < 0.639018$ and $x_2 < 0.534979$, then Label 0

- Try to interpret your D2 decision tree. Is it easy or possible to do so without visualization?
  It is almost impossible to interpret this decision tree without visualization, because there are too many rules and is too complicated, the only thing we can do is shorten each argument, I.E. merge constraints of

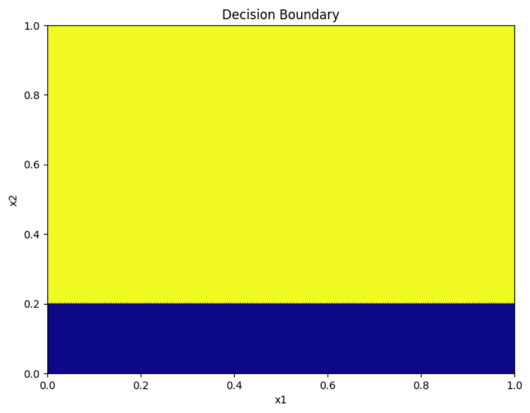
6. (Hypothesis space) [10 pts] For D1.txt and D2.txt, do the following separately:

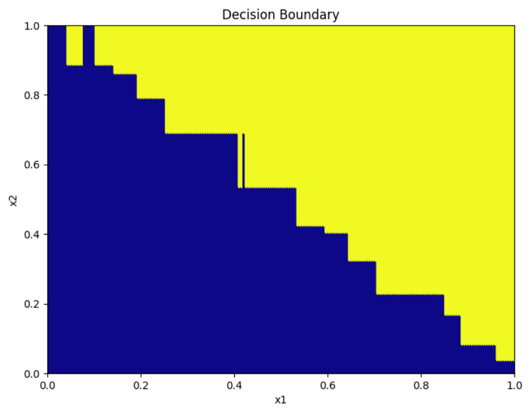
   - Produce a scatter plot of the data set.





   - 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).
     Yellow Region: Label 1, Blue Region: Lable 0

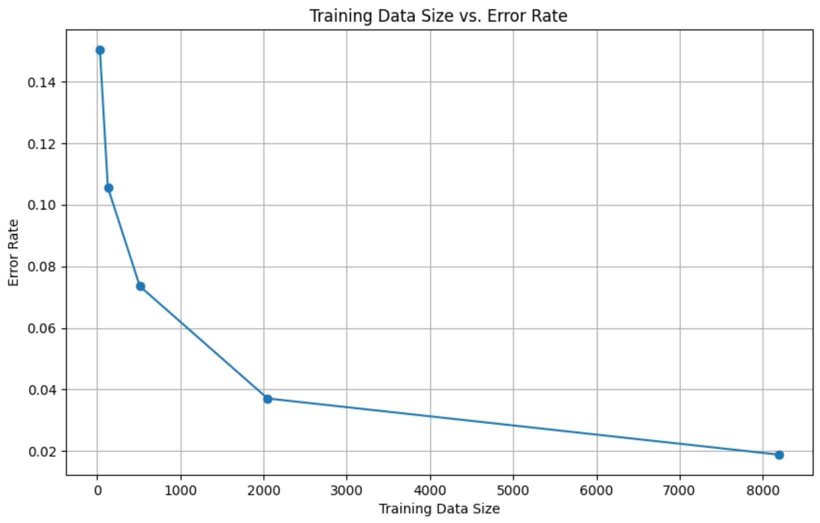




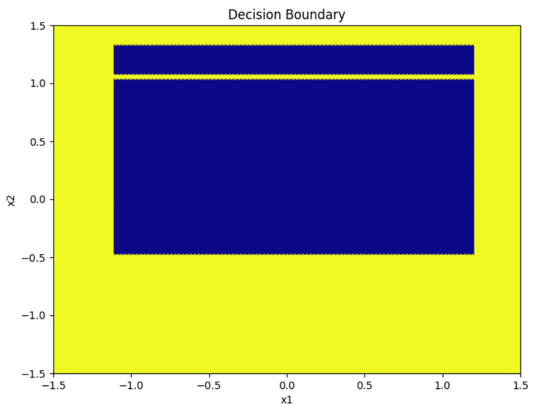
   Then discuss why the size of your decision trees on D1 and D2 differ. Relate this to the hypothesis space of our decision tree algorithm.
   since our hypothesis space has 2 dimensions. Then if the boundary is separated by something like a function of x_1 and x_2, the size of the decision tree will be significantly larger than those that only depends on one of the dimension. The process is similar to using vertical and horizontal lines to imitate a diagonal.

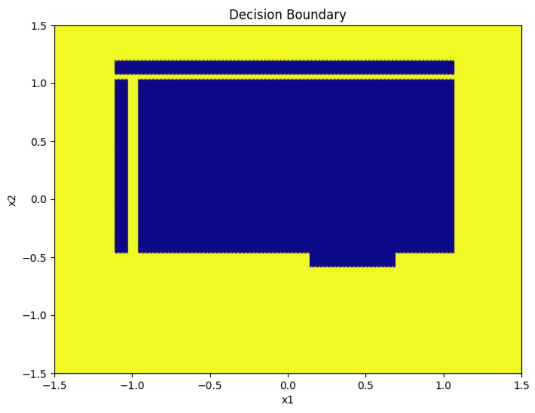
7. (Learning curve) [20 pts] We provide a data set Dbig.txt with 10000 labeled items. Caution: Dbig.txt is sorted.

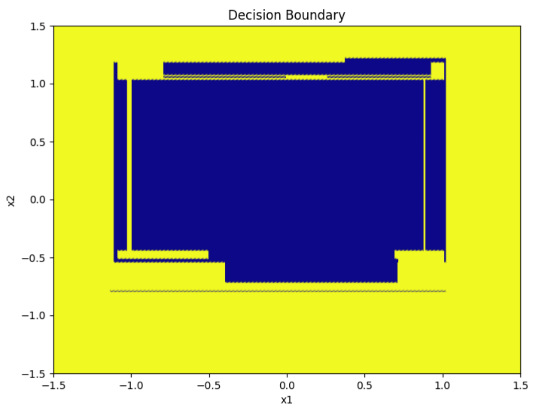
- You will randomly split Dbig.txt into a candidate training set of 8192 items and a test set (the rest). Do this by generating a random permutation, and split at 8192.
- Generate a sequence of five nested training sets $D_{32} \subset D_{128} \subset D_{512} \subset D_{2048} \subset D_{8192}$ from the 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).
  training data size: 32, number of nodes: 13, error rate: 0.1504424778761062
  training data size: 128, number of nodes: 23, error rate: 0.10564159292035398
  training data size: 512, number of nodes: 73, error rate: 0.07356194690265487
  training data size: 2048, number of nodes: 133, error rate: 0.03705752212389381
  training data size: 8192, number of nodes: 261, error rate: 0.018805309734513276

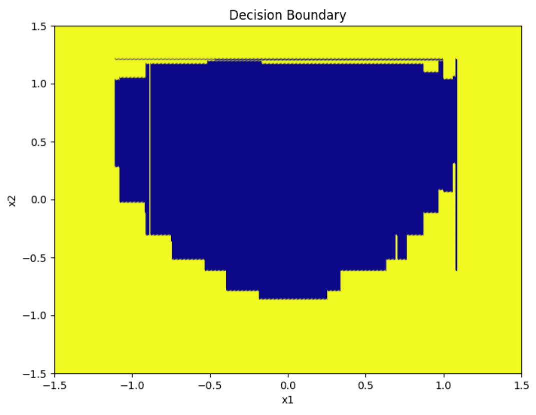


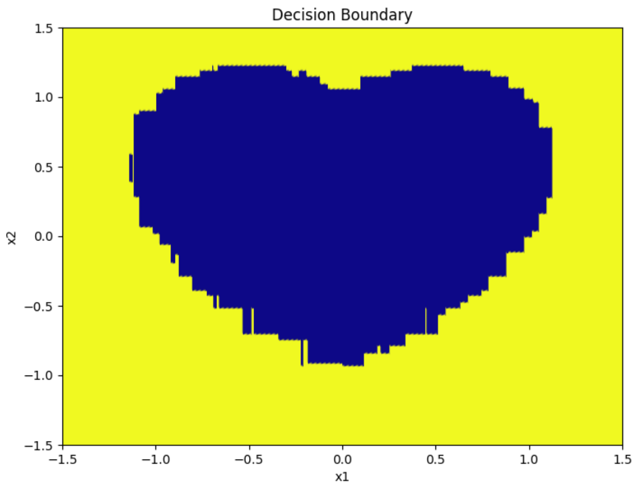
Yellow Region: Label 1, Blue Region: Lable 0





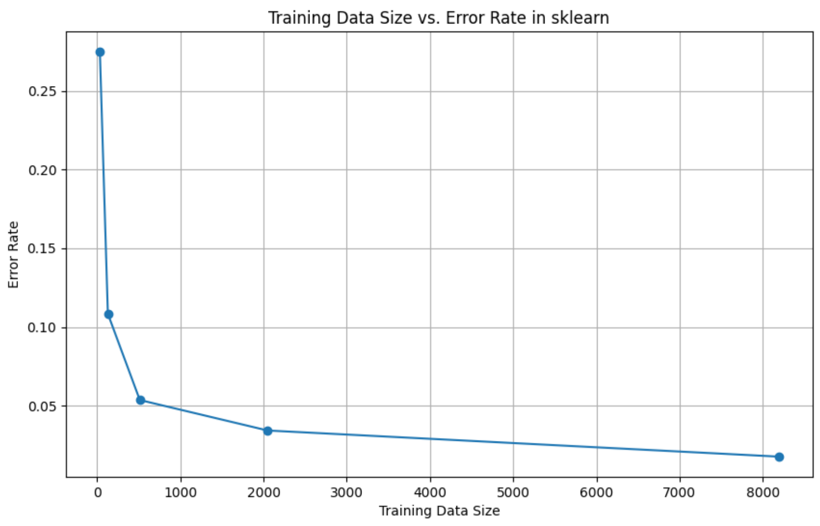






3 sklearn [10 pts]

Learn to use sklearn (https://scikit-learn.org/stable/). Use sklearn.tree.DecisionTreeClassifier to produce trees for datasets $D_{32}, D_{128}, D_{512}, D_{2048}, D_{8192}$. Show two things in your answer: (1) List n, number of nodes in that tree, err_n. (2) Plot n vs. err_n.
training data size: 32, number of nodes: 13, error rate: 0.27488938053097345
training data size: 128, number of nodes: 25, error rate: 0.1084070796460177
training data size: 512, number of nodes: 55, error rate: 0.05365044247787609
training data size: 2048, number of nodes: 117, error rate: 0.034292035398230114
training data size: 8192, number of nodes: 277, error rate: 0.017699115044247815



4 Lagrange Interpolation [10 pts]

Fix some interval $[a, b]$ and sample $n = 100$ points x from this interval uniformly. Use these to build a training set consisting of n pairs (x, y) by setting function $y = sin(x)$.

Build a model f by using Lagrange interpolation, check more details in https://en.wikipedia.org/wiki/Lagrange_polynomial and https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.lagrange.html.

Generate a test set using the same distribution as your test set. Compute and report the resulting model's train and test error. What do you observe? Repeat the experiment with zero-mean Gaussian noise ϵ added to x. Vary the standard deviation for ϵ and report your findings.
We pick [a, b] = [0, 2π].
Findings: when we used 100 points to do Lagrange interpolation, we got a 100-1=99 degree polynomials. Hence, we will incur a really large error in the training set due to errors in float calculation. The error in the test set is also large since the polynomial is oscillating dramatically [a, b]
However, we can observe that the test error becomes significantly smaller as we increase the standard deviation in the Gaussian noise. From my perspective, when we increase standard deviation, we spread the training dataset along with the X-axis and create a function that is more "flat" in the original interval [a, b] and this results in a significantly lower error in our test set, which lies in [a, b].

```
==============================
Lagrange Interpolation with no Gaussian Noise
Train log MSE Error: 340.25424032166745
Test log MSE Error: 341.29680441482924
==============================
Standard Deviation: 0.1
Train log MSE Error (Noisy): 337.0096859526621
Test log MSE Error (Noisy): 336.7015634640425
==============================
Standard Deviation: 1.0
Train log MSE Error (Noisy): 396.0104673797386
Test log MSE Error (Noisy): 367.59024240955125
==============================
Standard Deviation: 10
Train log MSE Error (Noisy): 268.0474921559637
```

```
Test log MSE Error (Noisy): 71.82999709191162
==============================
Standard Deviation: 100
Train log MSE Error (Noisy): 253.24328355004036
Test log MSE Error (Noisy): 8.07402015734845
==============================
Standard Deviation: 1000
Train log MSE Error (Noisy): 306.4836192402427
Test log MSE Error (Noisy): 1.3273045628949207
==============================
```