HOMEWORK 2

Mingcong Cao
9084259218

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.

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:

$$\begin{matrix} x_{11} & x_{12} & y_1 \\ & \dots & \\ x_{n1} & x_{n2} & y_n \end{matrix}$$

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.
   Because in this case, the information gain will be zero. WLOG, assume all the label is one, we have $p(label = 1) = 1$ and $p(label = 0) = 0$. So $H_D(Y) = 0$. Because $H_D(Y|S) >= 0$ so $InfoGain = 0$, which meets the stopping creterion.

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.
   The dataset is described in table 3. The algorithm will refuse to split it because in every split as well as the original dataset $p(y = 1) = 0.5$. Thus, we have $infoGain = 0$. However, if we force a split of x_1 with threshold 1. The algorithm will further split the tree with zero training error.

   Table 1: Dataset

index	x_1	x_2	y
1	1	0	0
2	1	1	1
3	0	1	0
4	0	0	1

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 split is: x_1 threshold = 0.00
    split entropy: 0.000  info gain: 0.000  gain ratio: 0.000
candidate split is: x_1 threshold = 0.10
    split entropy: 0.439  info gain: 0.044  gain ratio: 0.101
candidate split is: x_2 threshold = -2.00
    split entropy: 0.000  info gain: 0.000  gain ratio: 0.000
candidate split is: x_2 threshold = -1.00
    split entropy: 0.439  info gain: 0.044  gain ratio: 0.101
candidate split is: x_2 threshold = 0.00
    split entropy: 0.684  info gain: 0.038  gain ratio: 0.056
candidate split is: x_2 threshold = 1.00
    split entropy: 0.845  info gain: 0.005  gain ratio: 0.006
candidate split is: x_2 threshold = 2.00
    split entropy: 0.946  info gain: 0.001  gain ratio: 0.001
candidate split is: x_2 threshold = 3.00
    split entropy: 0.994  info gain: 0.016  gain ratio: 0.016
candidate split is: x_2 threshold = 4.00
    split entropy: 0.994  info gain: 0.049  gain ratio: 0.050
candidate split is: x_2 threshold = 5.00
    split entropy: 0.946  info gain: 0.105  gain ratio: 0.111
candidate split is: x_2 threshold = 6.00
    split entropy: 0.845  info gain: 0.200  gain ratio: 0.236
candidate split is: x_2 threshold = 7.00
    split entropy: 0.684  info gain: 0.038  gain ratio: 0.056
candidate split is: x_2 threshold = 8.00
    split entropy: 0.439  info gain: 0.189  gain ratio: 0.430
```

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.
   The fitted tree is shown in the code below, the left tree is then and right is else:

```
X_1 < 10 ?
 left:X_2 < 3 ?
  left:0
  right:1
 right:1
```

   The logic expression is: $(x_1 >= 10) \vee (x_2 >= 3)$

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.

     ```
     X_2 < 0.201829 ?
      left:0
      right:1
     ```

   - 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.

     The decision boundary should be a straight line defined by $x_2 = 0.201829$ in the $x_1 - x_2$ plane.

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

     ```
     X_1 < 0.533076 ?
       left:X_2 < 0.88635 ?
          left:X_2 < 0.691474 ?
             left:X_2 < 0.534979 ?
                left:0
                right:X_1 < 0.426073 ?
                  left:X_1 < 0.409972 ?
                     left:X_1 < 0.393227 ?
                        left:0
                        right:X_1 < 0.39583 ?
                          left:1
                          right:0
                     right:X_1 < 0.417579 ?
                       left:1
                       right:0
                  right:1
             right:X_1 < 0.254049 ?
               left:X_1 < 0.191915 ?
                  left:X_2 < 0.864128 ?
                     left: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.

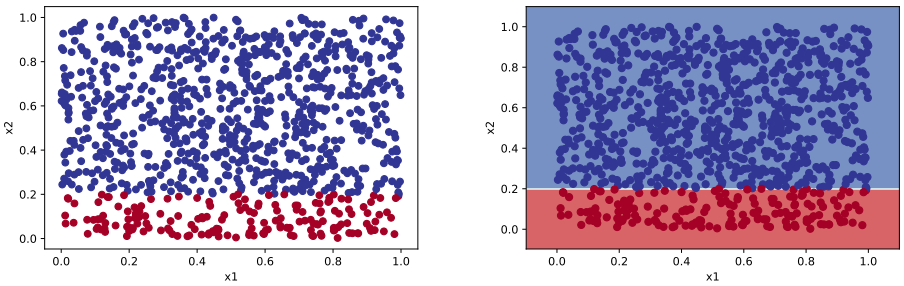
```
                  right:X_1 < 0.144781 ?
                    left:0
                    right:1
               right:X_2 < 0.792752 ?
                 left:0
                 right:1
          right:1
    right:X_1 < 0.041245 ?
      left:0
      right:X_1 < 0.104043 ?
        left:X_1 < 0.07642 ?
           left:1
           right:0
        right:1
right:X_2 < 0.228007 ?
  left:X_1 < 0.887224 ?
     left:X_1 < 0.850316 ?
        left:0
        right:X_2 < 0.169053 ?
          left:0
          right:1
     right:X_2 < 0.037708 ?
       left:0
       right:X_2 < 0.082895 ?
         left:X_1 < 0.960783 ?
            left:0
            right:1
          right:1
  right:X_2 < 0.424906 ?
    left:X_1 < 0.708127 ?
       left:X_2 < 0.32625 ?
          left:0
          right:X_1 < 0.595471 ?
            left:0
            right:X_1 < 0.646007 ?
              left:X_2 < 0.403494 ?
                 left:0
                 right:1
              right:1
       right:1
    right:1
```

- Try to interpret your D2 decision tree. Is it easy or possible to do so without visualization?

  It is impossible to interpret the D2 decision tree in human understandable languages. As the depth of the tree increases, it becomes harder to interpret the result.

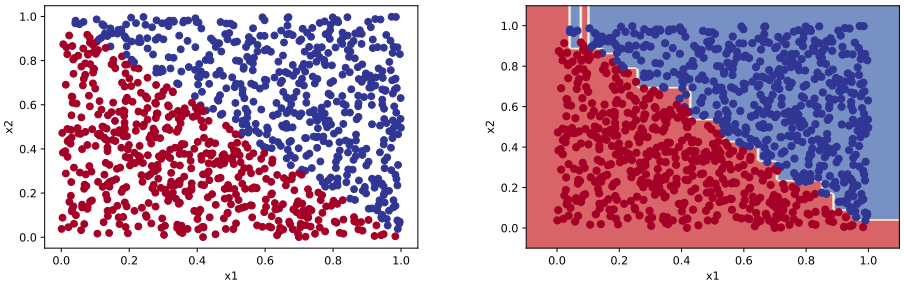
6. (Hypothesis space) [10 pts] For D1.txt and D2.txt, do the following separately:

   - 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).



(a) scatter plot of D1 (b) Figure 2

Figure 1: Scatter Plot and Decision Boundary of D1



(a) scatter plot of D1 (b) Figure 2

Figure 2: Scatter Plot and Decision Boundary of D2

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.

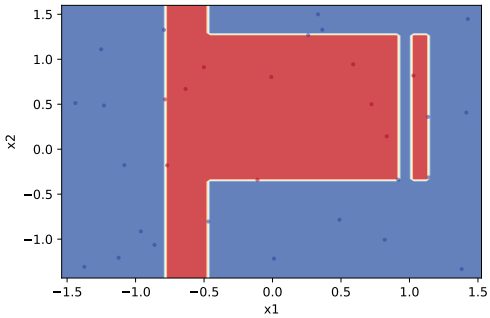
The scatter plot and decision boundary are plotted in figure 1 and figure 2. For this problem, in each split of a decision tree, only a horizontal or vertical split can be created. For D1, the decision boundary is horizontal so the decision tree is simple. In contrast, D2 has an oblique boundary which requires a deeper tree to fit.

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

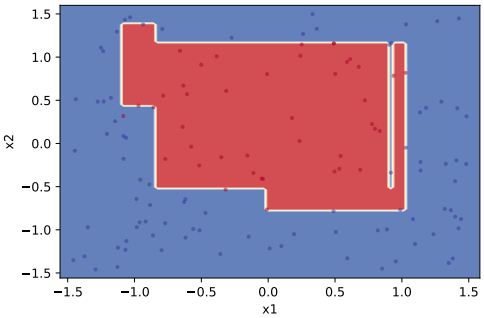
   - You will randomly split Dbig.txt into a candidate training set of 8192 items and a test set (the rest). Do this by generating a random permutation, and split at 8192.
   - Generate a sequence of five nested training sets $D_{32} \subset D_{128} \subset D_{512} \subset D_{2048} \subset D_{8192}$ from 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).

Table 2: List of Node Number and err_n

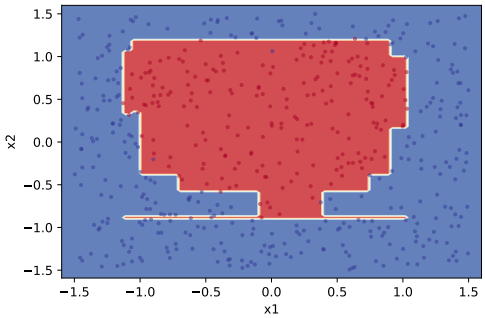
n	$NodeNumber$	err_n
32	15	0.178
128	25	0.082
512	53	0.045
2048	121	0.03
8192	265	0.018



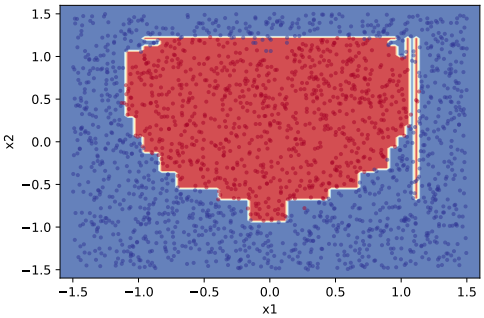
(a) n=32



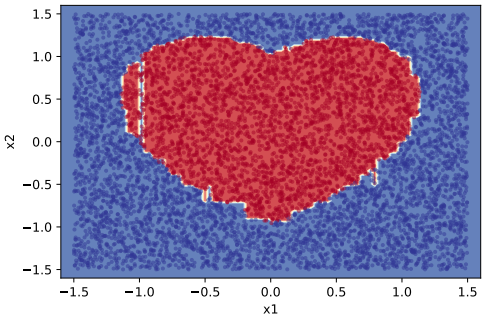
(b) n=128



(c) n=512



(d) n=2048



(e) n=8192

Figure 3: Decision Boundary for Decision Trees with Different n

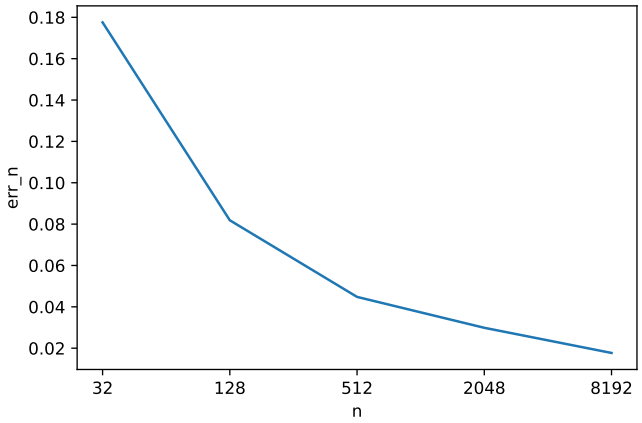


Figure 4: err_n with Respect to Training Data Size n

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.
The criterion is set to gini.

Table 3: List of Node Number and err_n using sklearn

n	$NodeNumber$	err_n
32	13	0.142
128	25	0.063
512	49	0.034
2048	121	0.026
8192	245	0.012

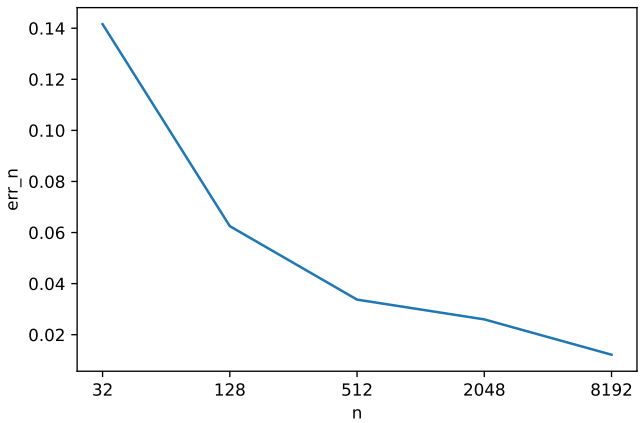


Figure 5: err_n with Respect to Training Data Size n for sklearn (criterion=gini)

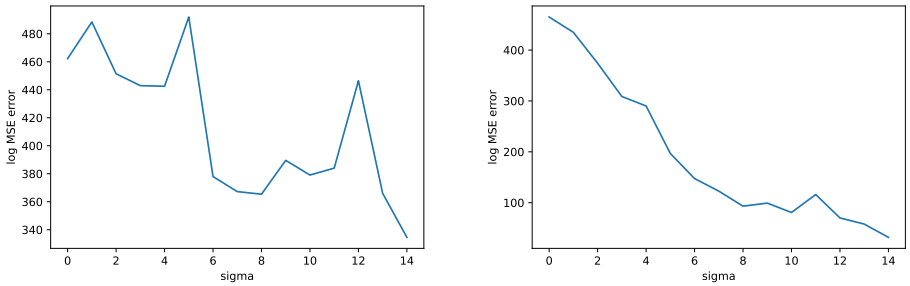
4 Lagrange Interpolation [10 pts]

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.

- $[a, b] = [0, 2\pi]$ and std vary from 0 to 14 with stepsize=1
- For question 1, The train error and test error become very big. The $\log 2$ of the MSE error for the train set is 469.912 and for the test set is 472.69.
- When the standard deviation increases, We can observe an overall decreasing trend in log MSE error for both the training and testing set. Especially for the test set, the error decreased significantly. When the std value increases, the training point will be more separated, reducing the spikes or fluctuations of the fitted polynomial. This is the reason for the decreasing errors.



(a) Training error with Respect to σ (b) Test error with Respect to σ

Figure 6: Training and Testing error with respect to σ