HOMEWORK 2

Karan Vikyath Veeranna Rupashree
908 458 3328
Github

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:

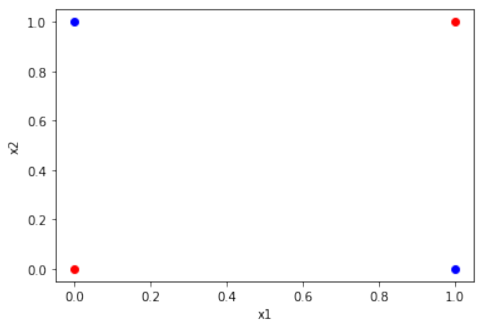
- Candidate splits (j, c) for numeric features should use a threshold c in feature dimension j in the form of $x_{\cdot j} \geq c$.
- c should be on values of that dimension present in the training data; i.e. the threshold is on training points, not in between training points. You may enumerate all features, 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 non-empty node has training items with same label, then information gain on any candidate splits given by $(H_D(Y) - H_D(Y/S))$ is zero, as logarithm of full probability will be zero. Hence, it is guaranteed to become 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.
   Lets consider the following dataset: XOR table

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0



   Now, initial entropy for this dataset is zero Let's consider the split where $x1 >= 1$. The Entropy for the two splits due to this condition also turns out to be:

   $E_{S1,x1>=1} = 1.0 \quad \& \quad E_{S2,x1<1} = 1.0$

   Also, considering other split conditions like $x2 >= 1$ leads to similar results in individual split entropies.Hence it comes to a stopping point making root the leaf.

   As per the last point in decision tree setup, whenever there is no majority class in a leaf, let it predict y = 1.

   We can override this manually by forcing a split where

   $\mathbf{x1 >= 1 \quad \& \quad x2 >= 1 : y = 0}$
   $\mathbf{x1 >= 1 \quad \& \quad x2 < 1 : y = 1}$
   $\mathbf{x1 < 1 \quad \& \quad x2 >= 1 : y = 1}$
   $\mathbf{x1 < 1 \quad \& \quad x2 < 1 : y = 0}$

   Thus, the stopping criteria of having zero information gain can be made to split further manually.

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.

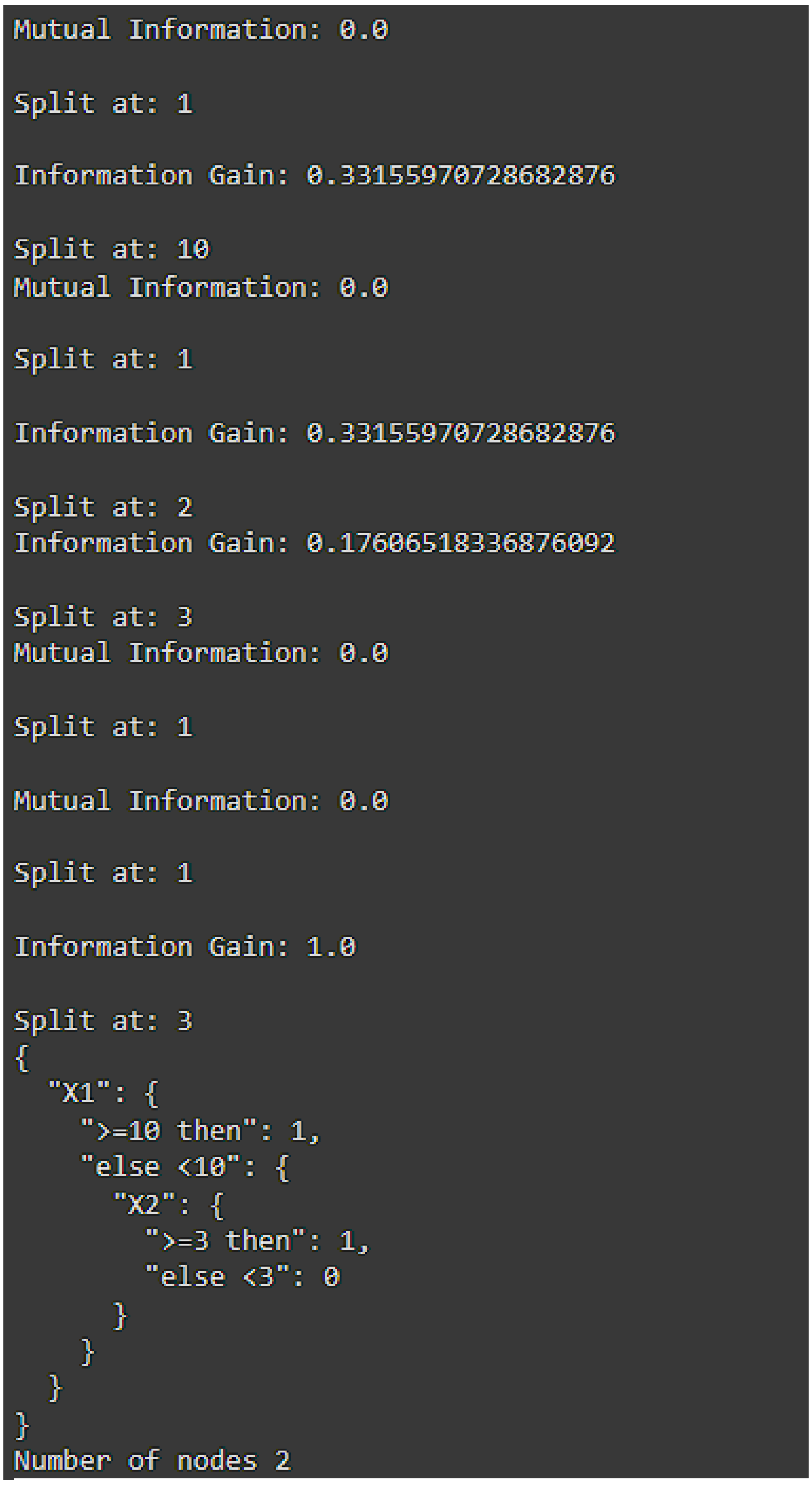
```
Mutual Information: 0.0

Split at: 0.0

Information Gain: 0.10051807676021828

Split at: 0.1
Mutual Information: 0.0

Split at: -2

Information Gain: 0.10051807676021828

Split at: -1
Information Gain: 0.05595375963126352 6

Split at: 0
Information Gain: 0.00578004220515232

Split at: 1
Information Gain: 0.0011443495172767494

Split at: 2
Information Gain: 0.016411136842102134

Split at: 3
Information Gain: 0.049749064181778546

Split at: 4
Information Gain: 0.11124029586339801

Split at: 5
Information Gain: 0.23609960614360798

Split at: 6
Information Gain: 0.055953759631263526

Split at: 7
Information Gain: 0.4301569161309807

Split at: 8
Mutual Information: 0.0

Split at: 0.0

Information Gain: 0.07280247297910734

Split at: 0.1
Mutual Information: 0.0

Split at: -2

Information Gain: 0.07280247297910734

Split at: -1
Information Gain: 0.1206166388929235

Split at: 0
```

```
Information Gain: 0.03661492441548469

Split at: 1
Information Gain: 0.007624890655589 7965

Split at: 2
Information Gain: 0.0

Split at: 3
Information Gain: 0.0076248906555897965

Split at: 4
Information Gain: 0.03661492441548469

Split at: 5
Information Gain: 0.1206166388929235

Split at: 6
Information Gain: 0.07280247297910734

Split at: 7
Mutual Information: 0.0

Split at: 0.0

Mutual Information: 0.0

Split at: 0

Information Gain: 0.04755786045881497

Split at: 1
Information Gain: 0.06908558886719268

Split at: 2
Information Gain: 0.0967687484016179

Split at: 3
Information Gain: 0.13792538097003

Split at: 4
Information Gain: 0.2087137558162035

Split at: 5
Information Gain: 0.36185425731195636

Split at: 6
Information Gain: 0.04755786045881497

Split at: 7
Mutual Information: 0.0

Split at: 0.0

Mutual Information: 0.0

Split at: 6
```

```
Information Gain: 1.0

Split at: 7
Mutual Information: 0.0

Split at: 0.0

Information Gain: 1.0

Split at: 0.1
Mutual Information: 0.0

Split at: -2

Information Gain: 1.0

Split at: -1
{
  "X2": {
    ">=8 then": 1,
    "else <8": {
      "X2": {
        ">=0 then": {
          "X2": {
            ">=6 then": {
              "X2": {
                ">=7 then": 0,
                "else <7": 1
              }
            },
            "else <6": 0
          }
        },
        "else <0": {
          "X2": {
            ">=0.1 then": 0,
            "else <0.1": 1
          }
        }
      }
    }
  }
}
```

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.

[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.


```
Mutual Information: 0.0

Split at: 1

Information Gain: 0.33155970728682876

Split at: 10
Mutual Information: 0.0

Split at: 1

Information Gain: 0.33155970728682876

Split at: 2
Information Gain: 0.17606518336876092

Split at: 3
Mutual Information: 0.0

Split at: 1

Mutual Information: 0.0

Split at: 1

Information Gain: 1.0

Split at: 3
{
  "X1": {
    ">=10 then": 1,
    "else <10": {
      "X2": {
        ">=3 then": 1,
        "else <3": 0
      }
    }
  }
}
Number of nodes 2
```

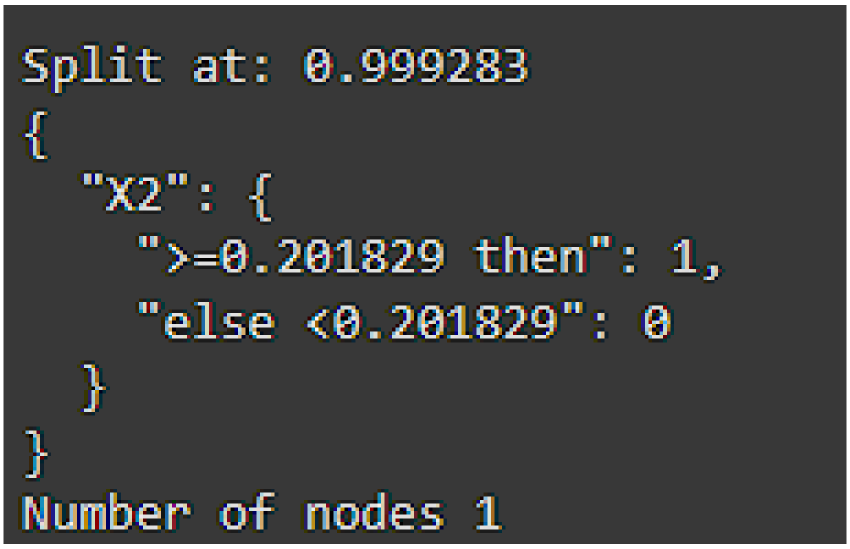
The logic rules:

```
If x1 >= 10:
    y = 1
else:
    If x2 >= 3:
        y = 1
```

```
else:
    y = 0
```

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 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.



```
Split at: 0.999283
{
  "X2": {
    ">=0.201829 then": 1,
    "else <0.201829": 0
  }
}
Number of nodes 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.
     It is observed from scatter plot of D1 in section 6 that the tree split is mainly dependent on X2, where for X2 $\geq$ 0.201829 then y is assigned 1 else it is 0.
   - Build a decision tree on D2.txt. Show it to us.

```
 Split at: 0.639018
{
  "X1": {
    ">=0.533076 then": {
      "X2": {
        ">=0.228007 then": {
          "X2": {
            ">=0.424906 then": 1,
            "else <0.424906": {
              "X1": {
                ">=0.708127 then": 1,
                "else <0.708127": {
                  "X2": {
                    ">=0.32625 then": {
                      "X1": {
                              ">=0.595471 then": {
                                "X1": {
                                  ">=0.646007 then": 1,
                                  "else <0.646007": {
                                    "X2": {
                                      ">=0.403494 then": 1,
                                      "else <0.403494": 0
                                    }
                                  }
                                }
                              },
                              "else <0.595471": 0
                            }
                          },
                          "else <0.32625": 0
                        }
                      }
                    }
                  }
                }
              },
              "else <0.228007": {
                "X1": {
                  ">=0.887224 then": {
                    "X2": {
                      ">=0.037708 then": {
                        "X2": {
                          ">=0.082895 then": 1,
                          "else <0.082895": {
                            "X1": {
                              ">=0.960783 then": 1,
                              "else <0.960783": 0
                            }
                          }
                        }
                      },
                      "else <0.037708": 0
                    }
                  },
                  "else <0.887224": {
                    "X1": {
                      ">=0.850316 then": {
                        "X2": {
                          ">=0.169053 then": 1,
                          "else <0.169053": 0
                        }
                      },
                      "else <0.850316": 0
                    }
                  }
                }
              }
            }
          },
          "else <0.533076": {
            "X2": {
              ">=0.88635 then": {
                "X1": {
```

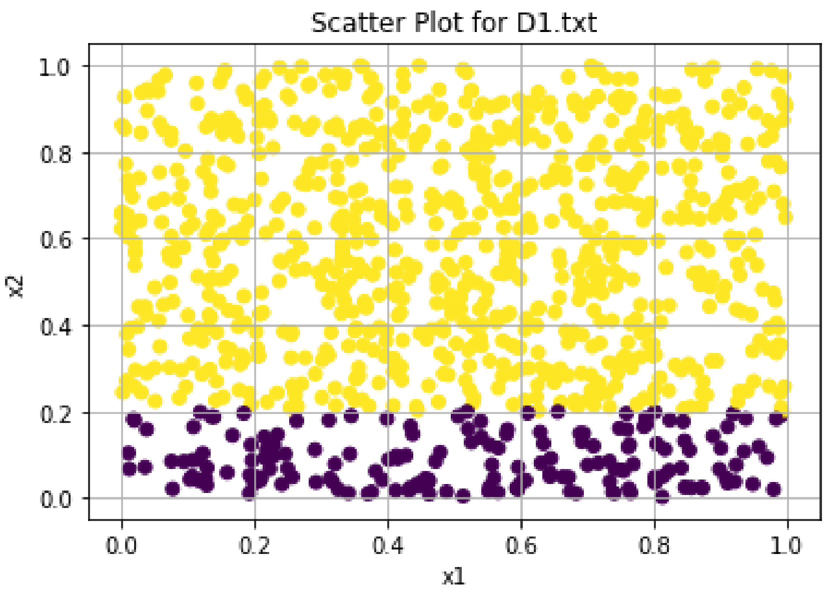
```
                ">=0.041245 then": {
                  "X1": {
                    ">=0.104043 then": 1,
                    "else <0.104043": {
                      "X1": {
                        ">=0.07642 then": 0,
                        "else <0.07642": 1
                      }
                    }
                  }
                },
                "else <0.041245": 0
              }
            },
            "else <0.88635": {
              "X2": {
                ">=0.691474 then": {
                  "X1": {
                    ">=0.254049 then": 1,
                    "else <0.254049": {
                      "X1": {
                        ">=0.191915 then": {
                          "X2": {
                            ">=0.792752 then": 1,
                            "else <0.792752": 0
                          }
                        },
                        "else <0.191915": {
                          "X2": {
                            ">=0.864128 then": {
                              "X1": {
                                ">=0.144781 then": 1,
                                "else <0.144781": 0
                              }
                            },
                            "else <0.864128": 0
                          }
                        }
                      }
                    }
                  }
                },
                "else <0.691474": {
                  "X2": {
                    ">=0.534979 then": {
                      "X1": {
                        ">=0.426073 then": 1,
                        "else <0.426073": {
                          "X1": {
                            ">=0.409972 then": {
                              "X1": {
                                ">=0.417579 then": 0,
                                "else <0.417579": 1
                              }
                            },
                            "else <0.409972": {
                              "X1": {
                                ">=0.393227 then": {
```

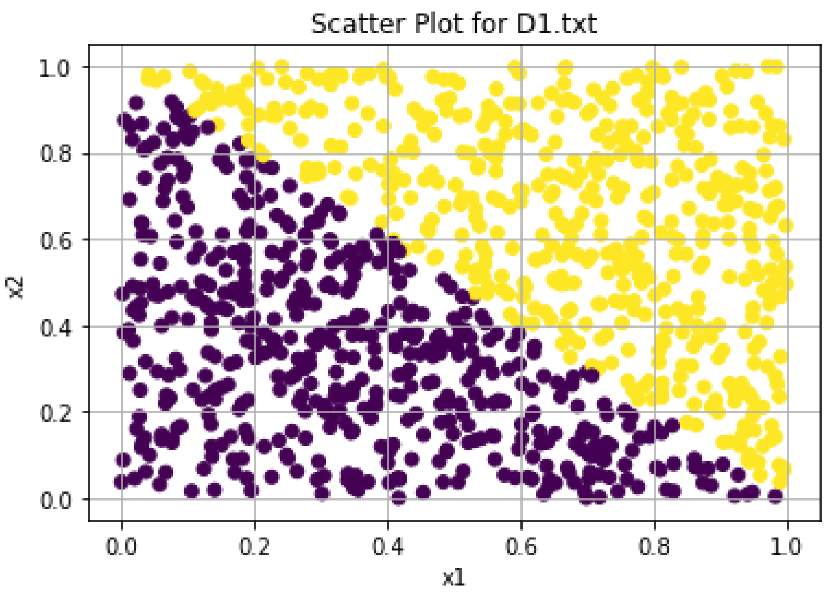
```
                                                "X1": {
                                                  ">=0.39583 then": 0,
                                                  "else <0.39583": 1
                                                }
                                              },
                                              "else <0.393227": 0
                                            }
                                          }
                                        }
                                      }
                                    }
                                  },
                                  "else <0.534979": 0
                                }
                              }
                            }
                          }
                        }
                      }
                    }
                  }
Number of nodes 30
```

- Try to interpret your D2 decision tree. Is it easy or possible to do so without visualization?
  The tree for D2 is clearly very complex and hence is very difficult to intrepret without proper visualization.

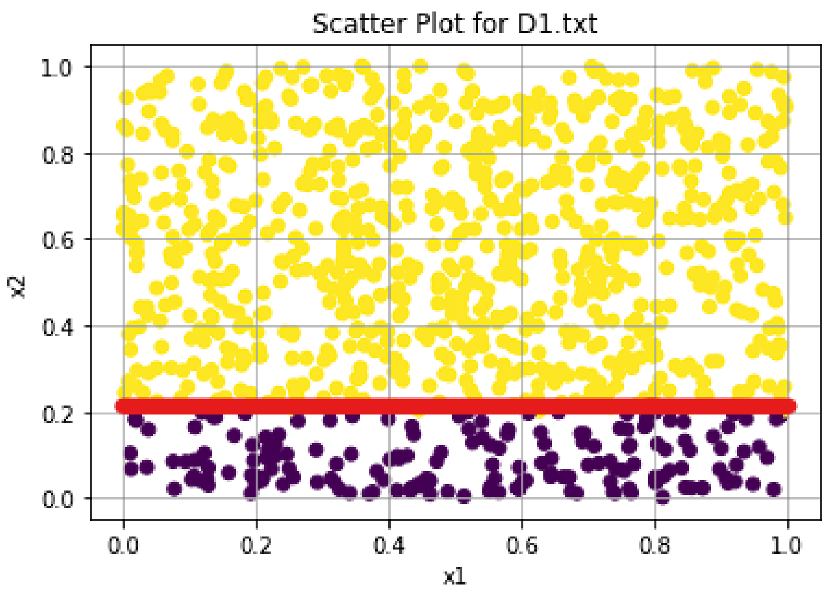
6. (Hypothesis space) [10 pts] For D1.txt and D2.txt, do the following separately:

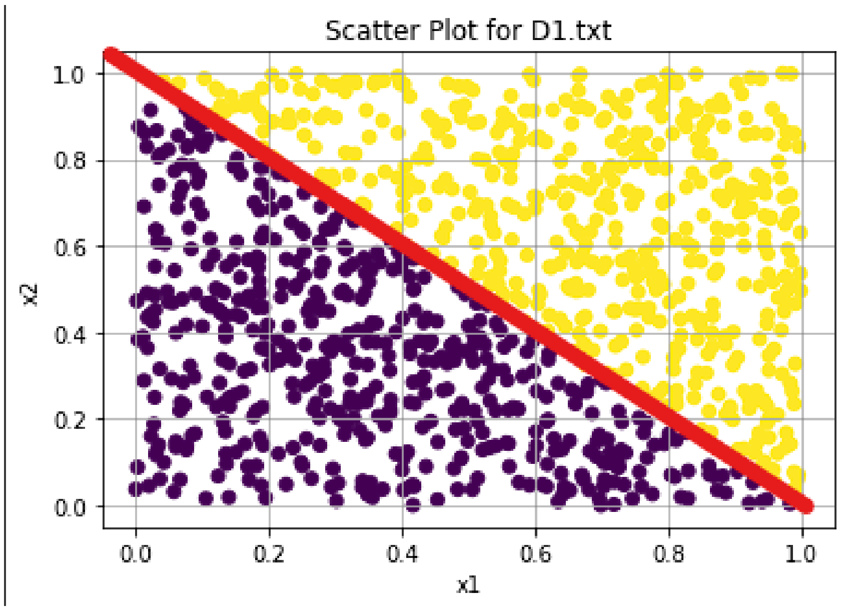
   - Produce a scatter plot of the data set.





- 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).





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.

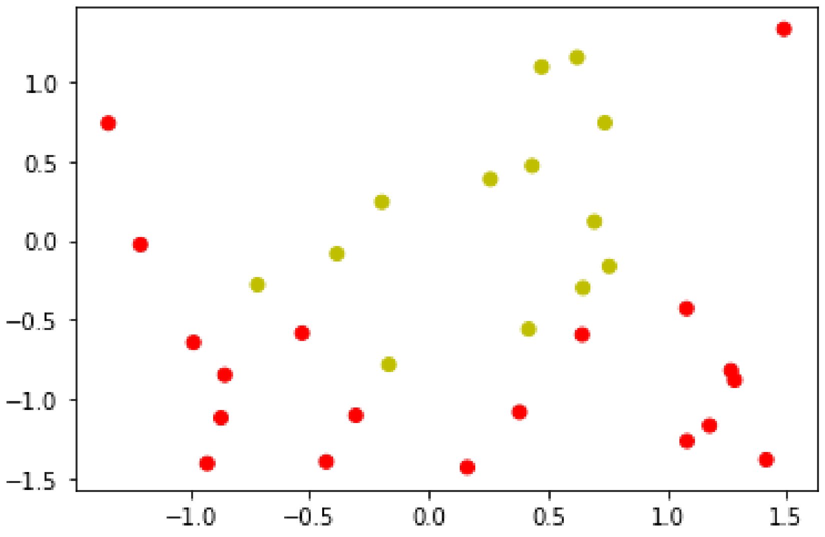
The decision boundary for D1 is a straight line parallel to the x1 axis.The decision hypothesis of D1 is not perfect as the nearby constant x2 could be discarded as irrelevant feature parameter. D2 has a negative slope boundary approximately $y = -x$ line due to the large number of decision splits concentrated in the top-left and bottom-right quadrant of the space.

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

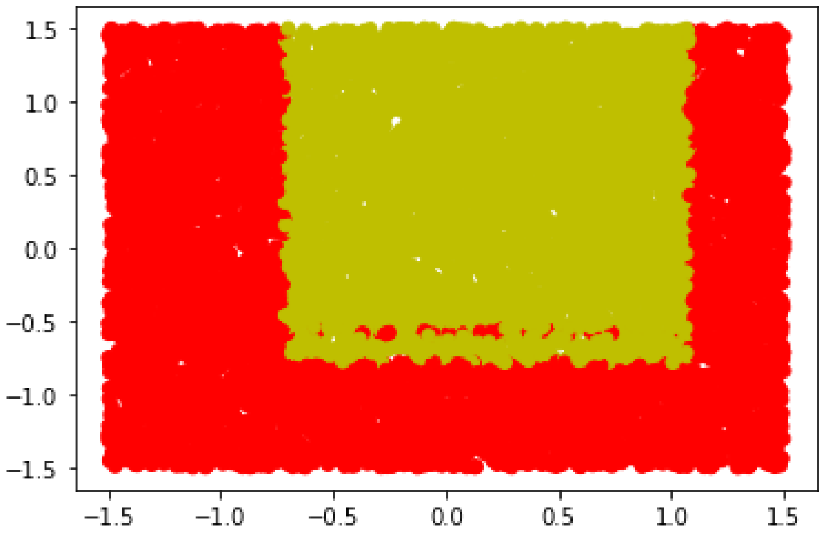
   - You will randomly split Dbig.txt into a candidate training set of 8192 items and a test set (the rest). Do this by generating a random permutation, and split at 8192.
   - Generate a sequence of five nested training sets $D_{32} \subset D_{128} \subset D_{512} \subset D_{2048} \subset D_{8192}$ from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

- <u>D32</u>

Training set plot

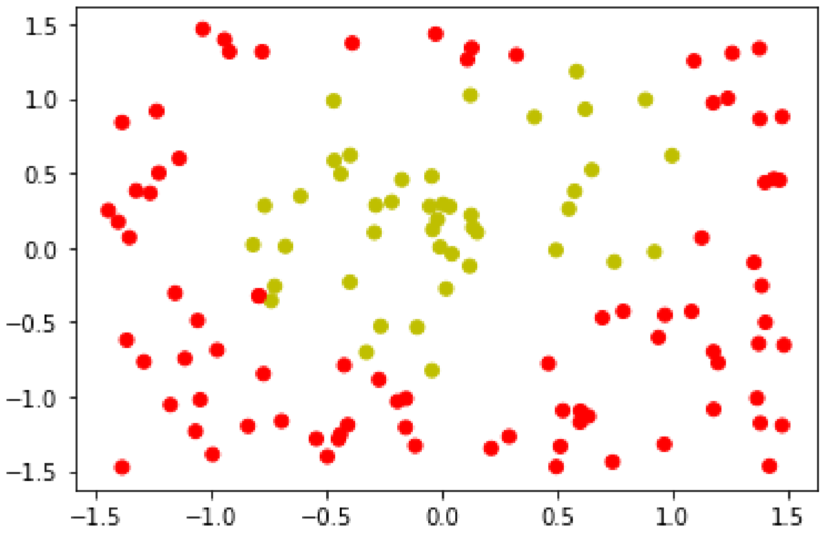


Decision Boundary

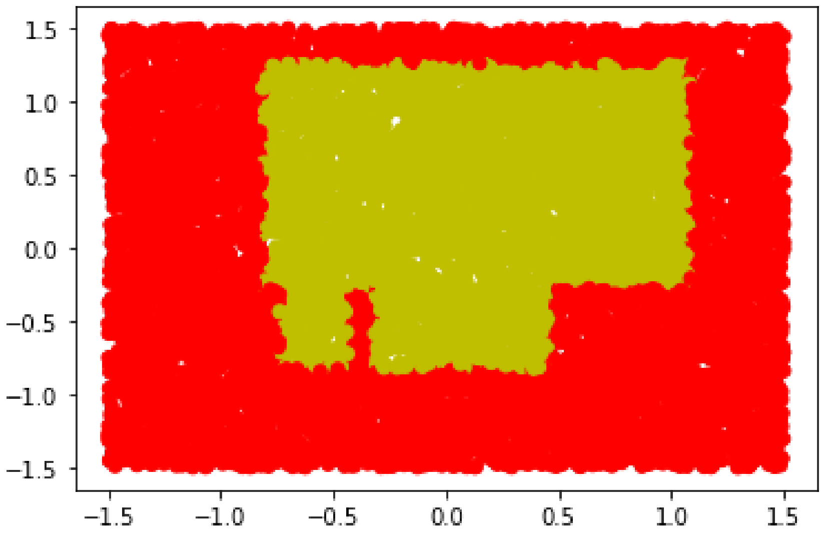


- D128

Training test plot

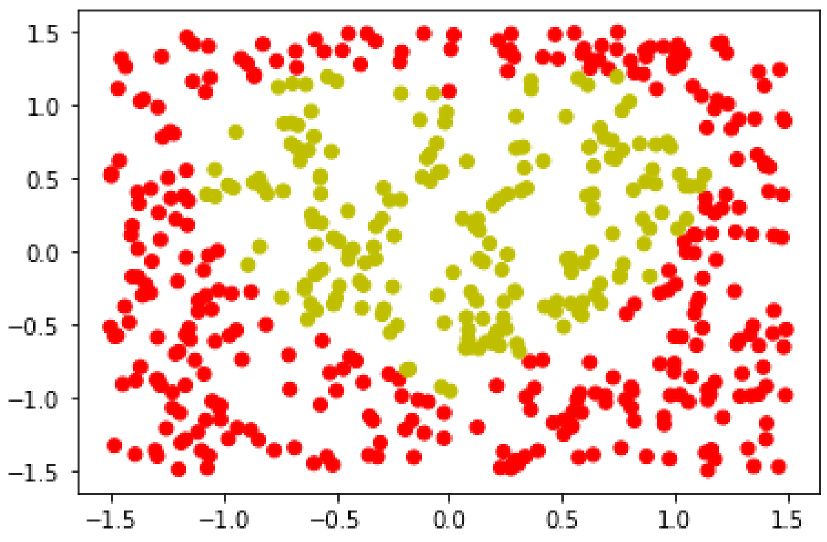


Decision Boundary

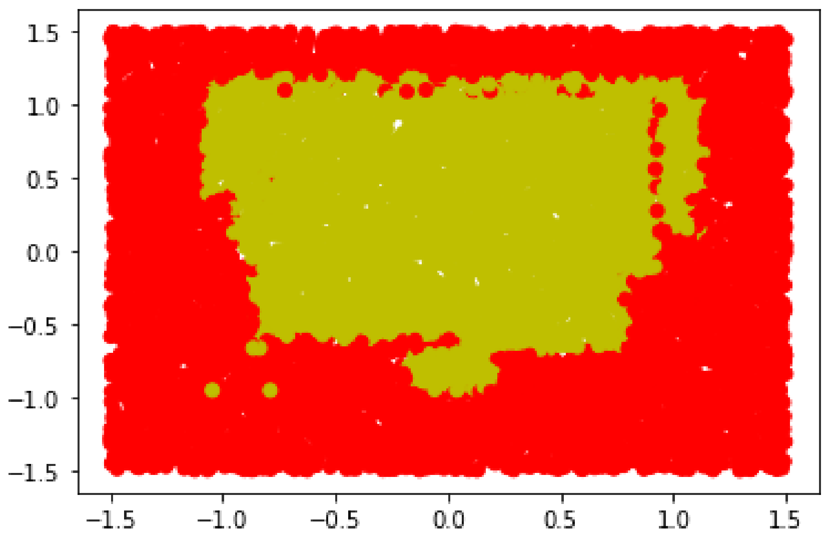


- D512

Training test plot

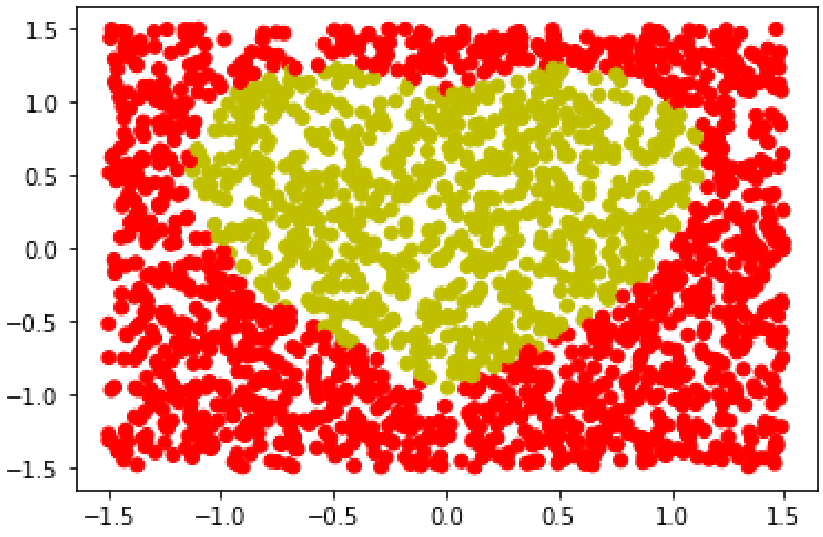


Decision Boundary

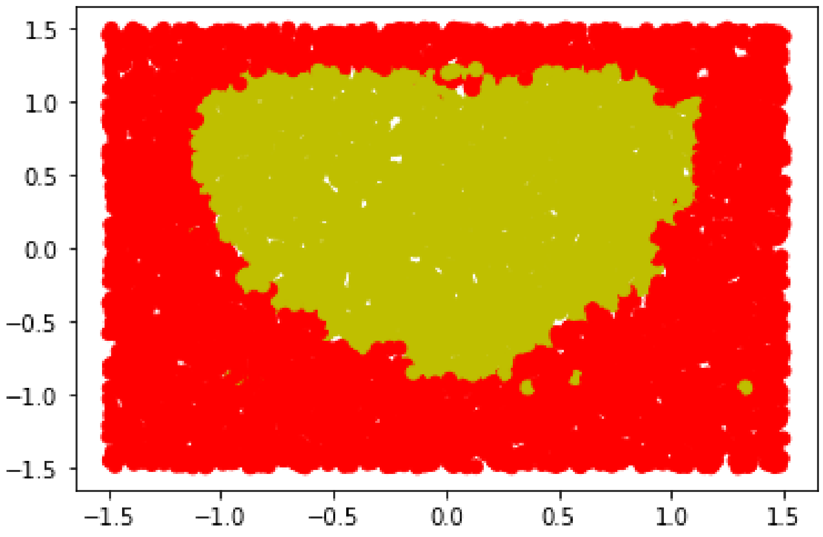


- D2048

Training test plot

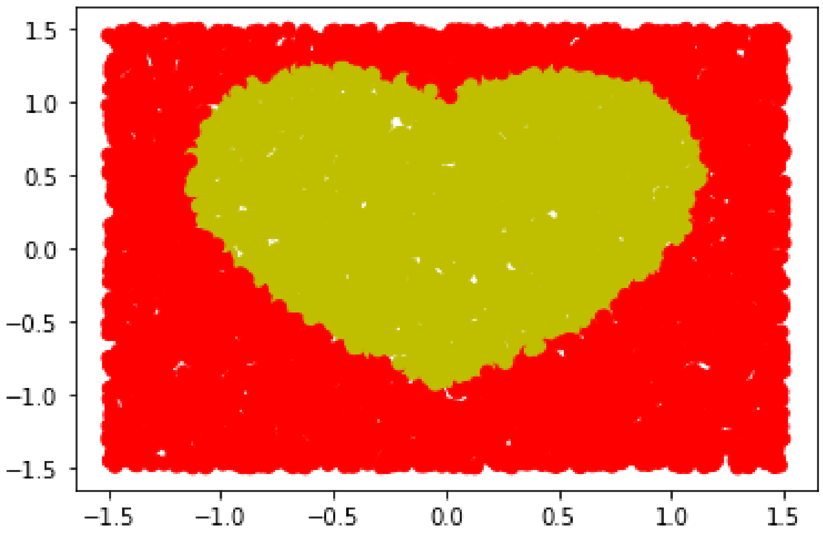


Decision boundary

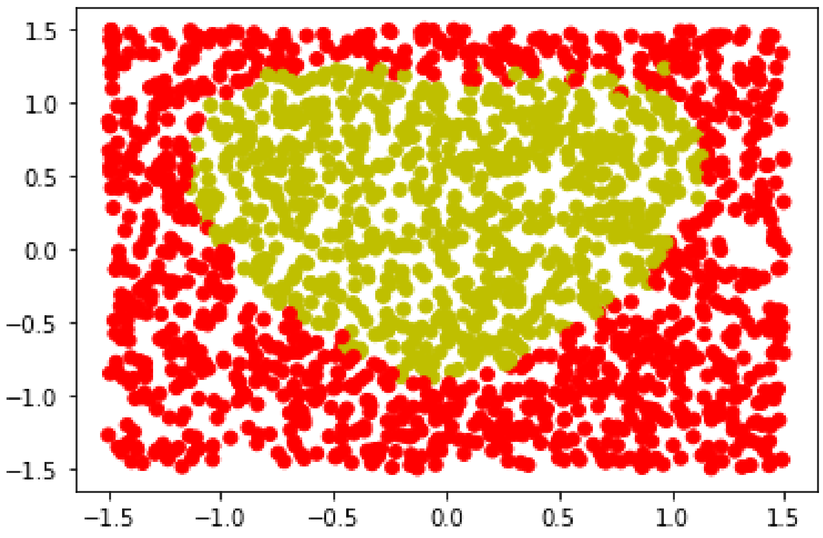


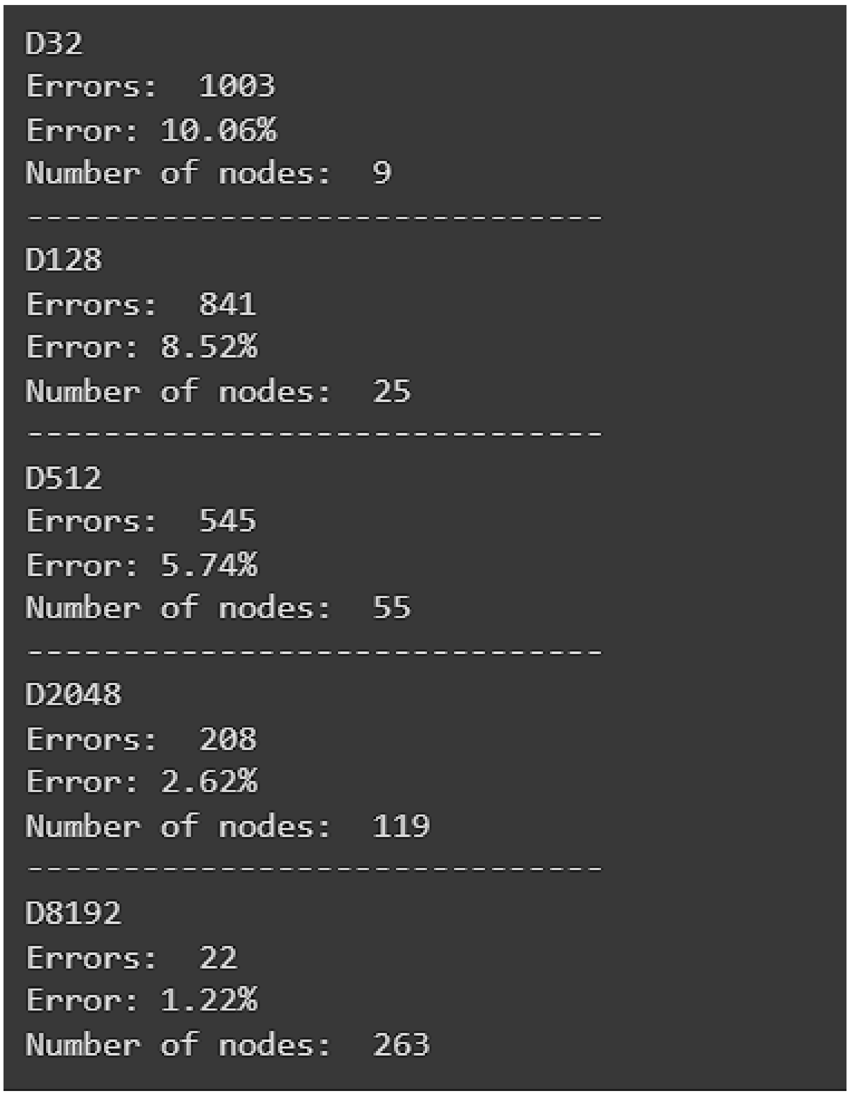
- D8192

Training set plot

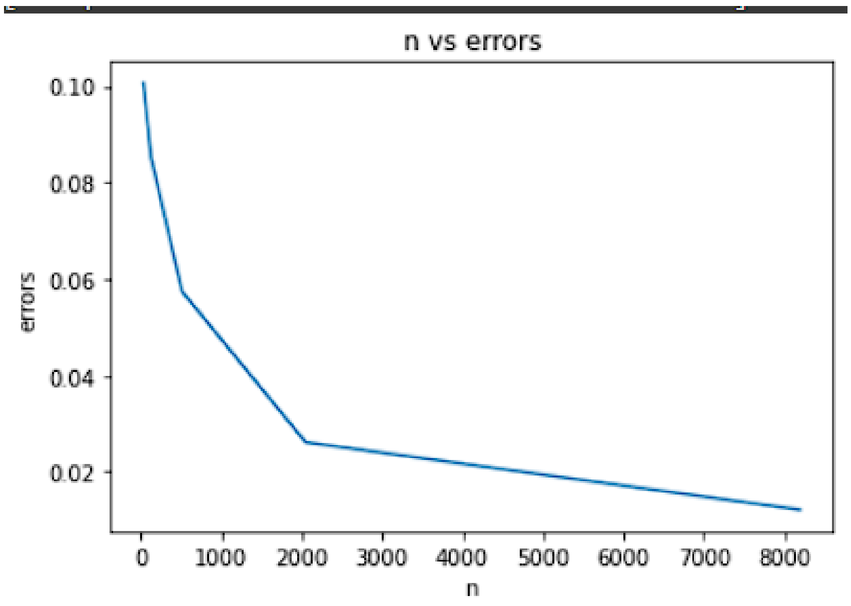


Decision Boundary



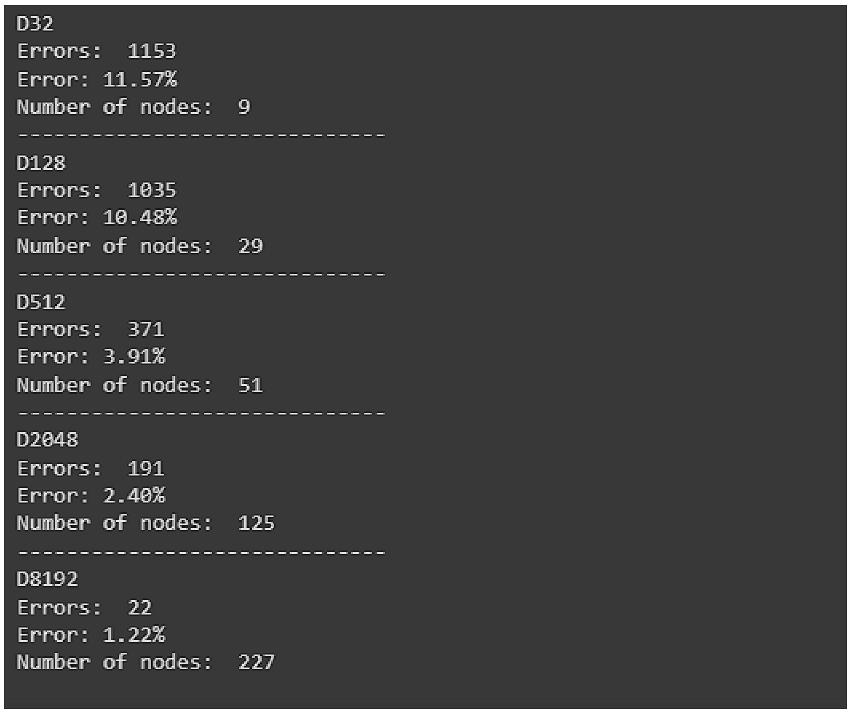


```
D32
Errors:  1003
Error: 10.06%
Number of nodes:  9
------------------------------
D128
Errors:  841
Error: 8.52%
Number of nodes:  25
------------------------------
D512
Errors:  545
Error: 5.74%
Number of nodes:  55
------------------------------
D2048
Errors:  208
Error: 2.62%
Number of nodes:  119
------------------------------
D8192
Errors:  22
Error: 1.22%
Number of nodes:  263
```

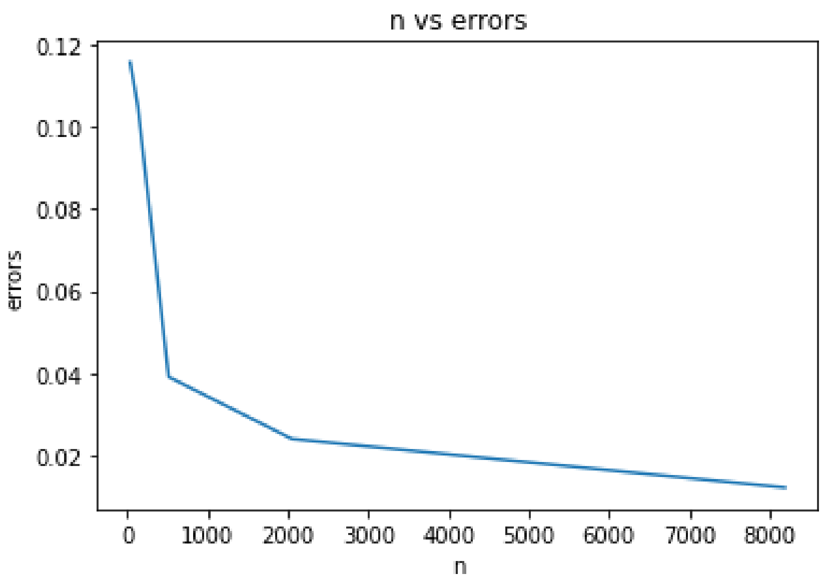


3 sklearn [10 pts]

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.



```
D32
Errors:  1153
Error: 11.57%
Number of nodes:  9
------------------------------
D128
Errors:  1035
Error: 10.48%
Number of nodes:  29
------------------------------
D512
Errors:  371
Error: 3.91%
Number of nodes:  51
------------------------------
D2048
Errors:  191
Error: 2.40%
Number of nodes:  125
------------------------------
D8192
Errors:  22
Error: 1.22%
Number of nodes:  227
```

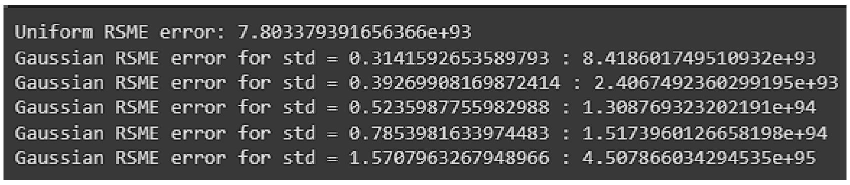


4 Lagrange Interpolation [10 pts]

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.


```
Uniform RSME error: 7.803379391656366e+93
Gaussian RSME error for std = 0.3141592653589793 : 8.418601749510932e+93
Gaussian RSME error for std = 0.39269908169872414 : 2.4067492360299195e+93
Gaussian RSME error for std = 0.5235987755982988 : 1.308769323202191e+94
Gaussian RSME error for std = 0.7853981633974483 : 1.5173960126658198e+94
Gaussian RSME error for std = 1.5707963267948966 : 4.507866034294535e+95
```

The Test error first increases with sigma.