HOMEWORK 2

Smit Priyesh Shah
9085876440

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.

Github Username: smit-1999
Project name: DecisionTreeLearner
Github Repo link

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 $y = 1$.

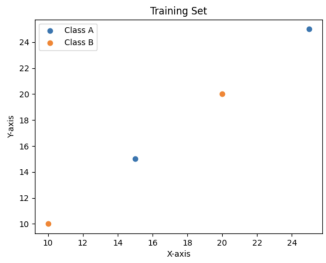
2 Questions

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.

   When a node is not empty and it contains items which all have the same label, it is guaranteed to become a leaf. A leaf in a decision tree is that node where the info gain is 0. When at a point, we encounter a node where all items have same label, the entropy before the split and after will be same. Hence, there is no information gain obtained. Also, if after a depth d we observe that all points have the same label, it doesn't make sense for building a tree of greater depth without any useful gain. As the output obtained on going deeper is going to be the same: the class label. Hence, our algorithm stops at pure labels and we terminate and call it 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.

   If we take a dataset which alternates in class labels we can find a case where the algorithm refuses to split.



   In the above case if we run the algorithm we learnt in class, we will first sort based on each feature value. here the points are chosen such that x=y and class labels are alternating. Hence, we can sort either on X or Y and it will be symmetric. Here, if we sort on X and then check differing class labels, we can take 2nd and 3rd point as possible split. In both cases, the initial entropy is 1 and the final entropy is also 1. Hence, there is no information gain and hence it is not a valid split. This

   However, if we manually force a split at any point on the X-axis, every point will be in a leaf node corresponding to its class

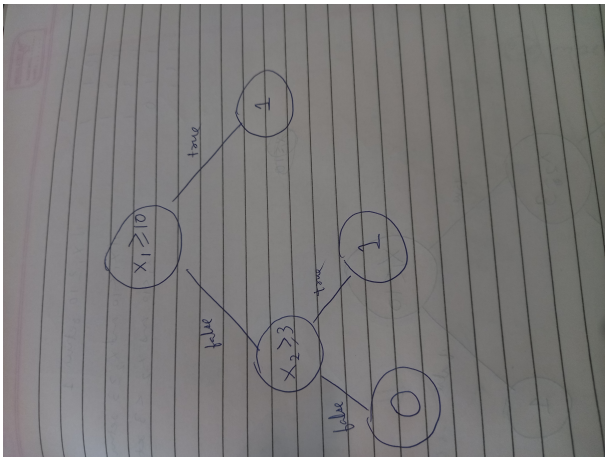
3. (Information gain ratio exercise) [10 pts] Use the training set Druns.txt. For the root node, list all candidate cuts and their information gain ratio. If the entropy of the candidate split is zero, please list its mutual information (i.e. information gain). Hint: to get $\log_2(x)$ when your programming language may be using a different base, use `log(x)/log(2)`. Also, please follow the split rule in the first section.

   $X2 >= 8.0$ Gain ratio: 0.4301569
   $X2 >= 7.0$ Gain ratio: 0.0559537
   $X2 >= 6.0$ Gain ratio: 0.236099
   $X2 >= 5.0$ Gain ratio: 0.111240
   $X2 >= 4.0$ Gain ratio: 0.049749
   $X2 >= 3.0$ Gain ratio: 0.016411
   $X2 >= 2.0$ Gain ratio: 0.001144
   $X2 >= 1.0$ Gain ratio: 0.055780
   $X2 >= 0.0$ Gain ratio: 0.055953
   $X2 >= -1.0$ Gain ratio: 0.100518
   $X2 >= -2.0$ Info gain: 0.0

$X1 >= 0.1$ Gain ratio: 0.100518
$X1 >= 0.0$ Info gain: 0.0

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.

Here, in D3leaves.txt there are 5 data points given to us. We need to build a classifier/tree. We can see that
if $X1 >= 10$ the class label is 1
if $X1 < 10$:
a)$X2 >= 3$ the class label is again 1
b)if $X1 < 10 and X2 < 3$ the class label is 0
These are the 3 rules based on which we can classify a given point.

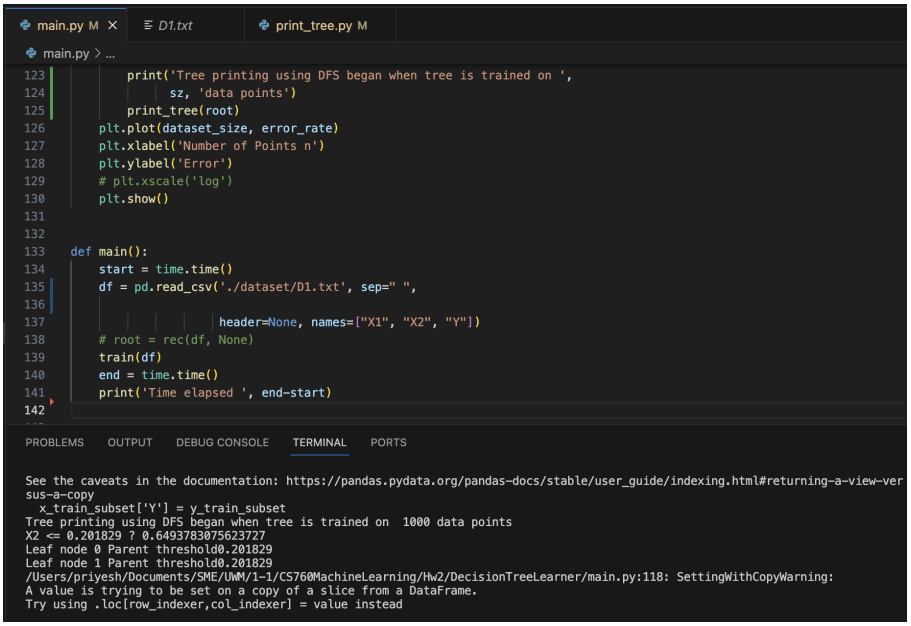


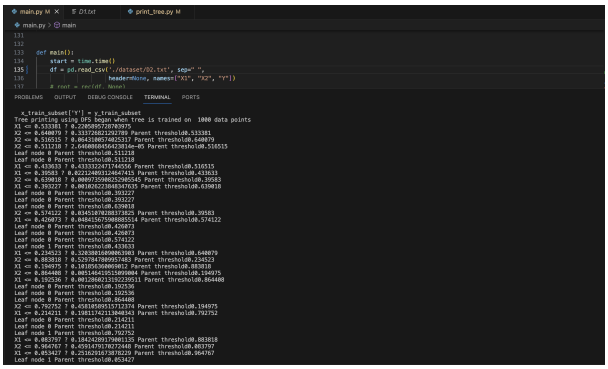
Q2.5.3

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.

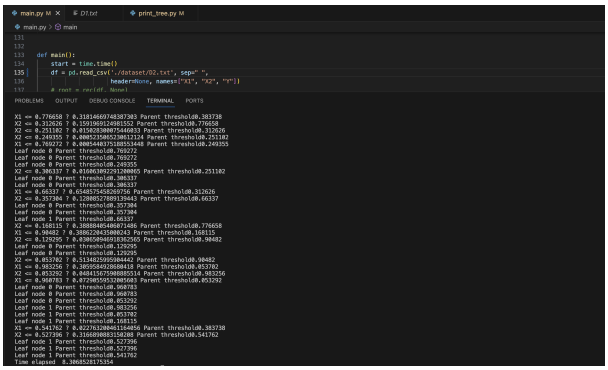
   - Build a decision tree on D1.txt. Show it to us in any format (e.g. could be a standard binary tree with nodes and arrows, and denote the rule at each leaf node; or as simple as plaintext output where each line represents a node with appropriate line number pointers to child nodes; whatever is convenient 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. There is only one root node in the tree based on which either all values go to left subtree or right subtree. Hence, in the image belowe we can see that X2¡=0.201829 is the answer and rest are leaf nodes. For leaf nodes, I am printing the parent threshold. For every other node, the column name (X1,X2) along with ¡= *threshold* and parent is printed.
   - 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.
     Since there is only one point in the decision tree, it means that the data should be linearly separable , either a line parallel to x-axis or a line parallel t y-axis should exist which clearly separates the class labels in the dataset D1.txt.
   - Build a decision tree on D2.txt. Show it to us.

[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.





Q2.5.3

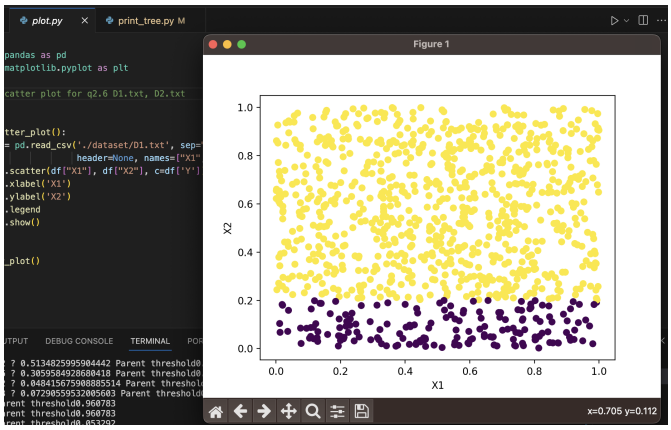


Q2.5.3 contd

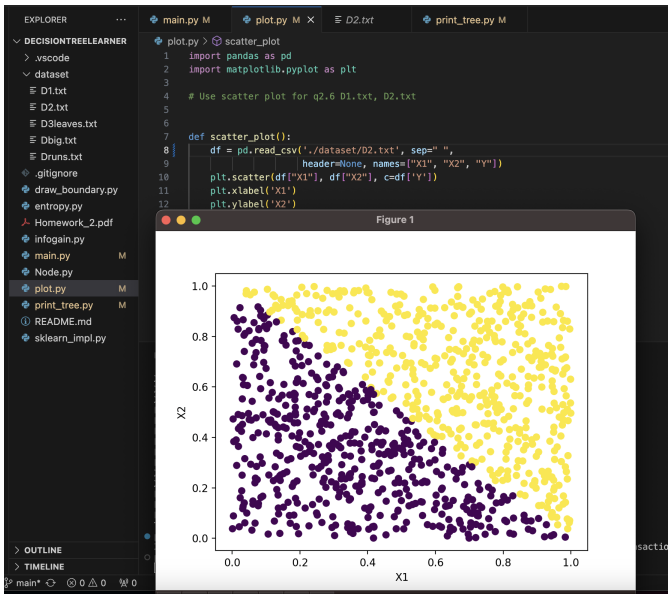
- Try to interpret your D2 decision tree. Is it easy or possible to do so without visualization? It is not possible to interpret the decision tree. This is simply because there are too many points and clearly imagining the cuts of parallel lines across y and x axis is very difficult. Hence, without visualizing the boundary it is very difficult to interpret how our current decision tree lies.

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

- Produce a scatter plot of the data set. *Scatter plot produced using plot.py file*



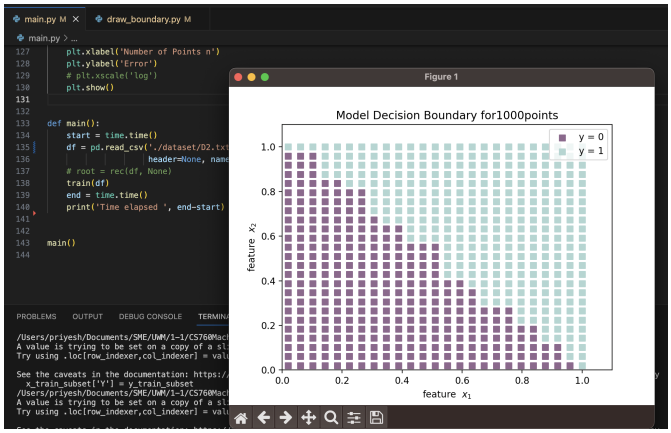
Q2.6.1: D1.txt scatter plot



Q2.6.1: D2.txt scatter 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).

Q2.6.2: D1.txt visualization plot



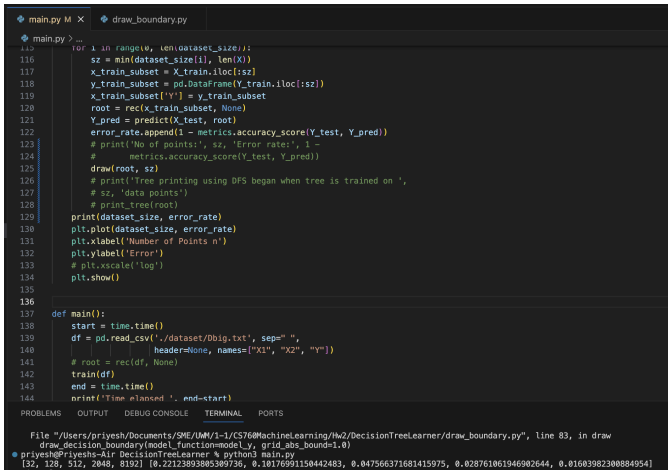
Q2.6.2: D2.txt visualization plot

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.

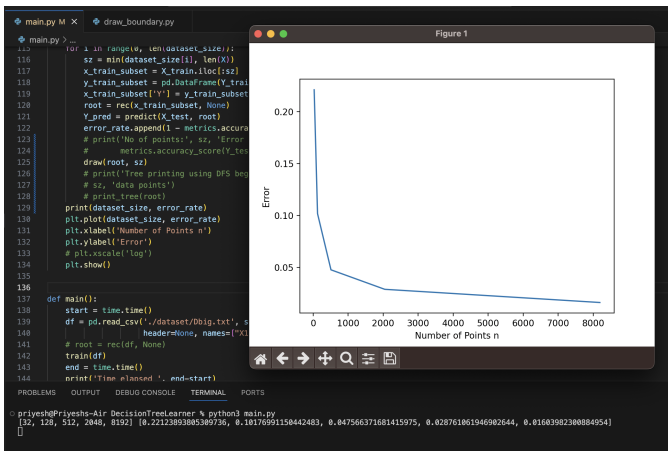
After visualizing the decision boundary of trees, it is evident that the sizes of the trees must differ. In D1.txt the decision boundary is a straight line parallel to x axis and hence there is only one point as expected. However, in D2.txt there is a line with a slope and hence there are multiple nodes in the decision tree. It is only with visualization that we are able to compare and get a better idea. Since, decision tree boundaries are lines parallel to the axes, to obtain a decision boundary as shown in D2.txt we will need multiple horizontal and vertical lines, hence multiple nodes in the decision tree (each for one horizontal/vertical line on axes). This correlates to the hypothesis space of our decision tree where trees with single feature and axis parallel split are available. All possible combinations of trees make up the hypothesis space in a decision tree.

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

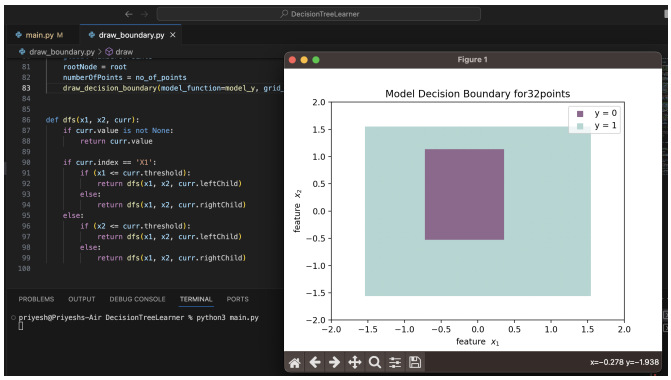
   - You will randomly split Dbig.txt into a candidate training set of 8192 items and a test set (the rest). Do this by generating a random permutation, and split at 8192.
   - Generate a sequence of five nested training sets $D_{32} \subset D_{128} \subset D_{512} \subset D_{2048} \subset D_{8192}$ from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



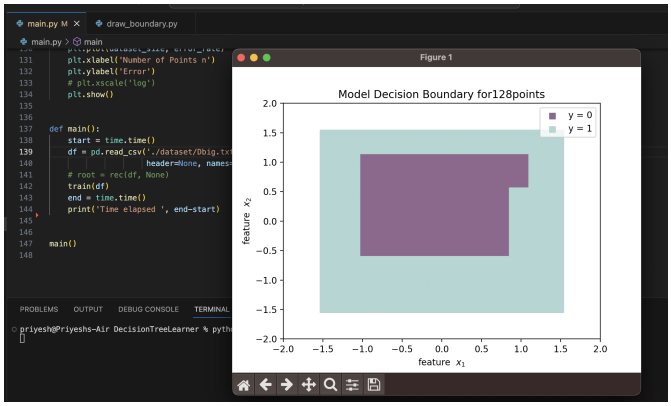
Q2.7.3: Dbig.txt Error value



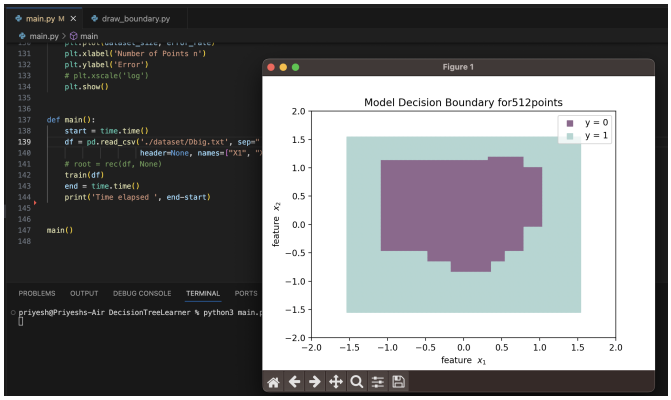
Q2.7.3: Dbig.txt Error plot



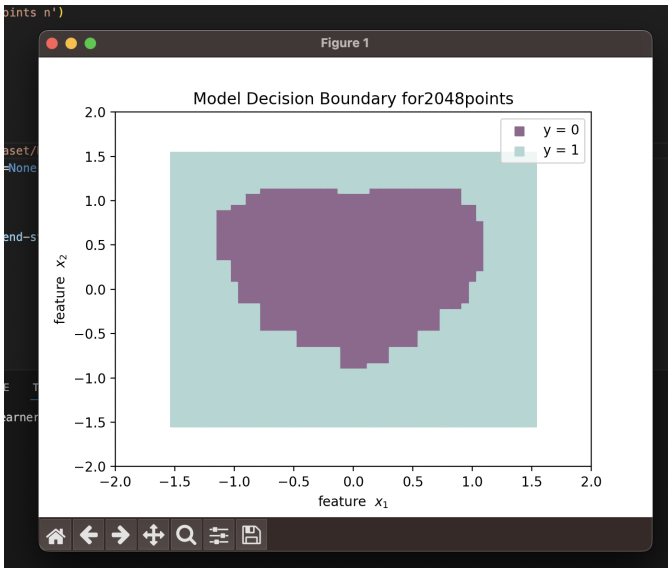
Q2.7.3: Dbig.txt 32 points



Q2.7.3: Dbig.txt 128 points



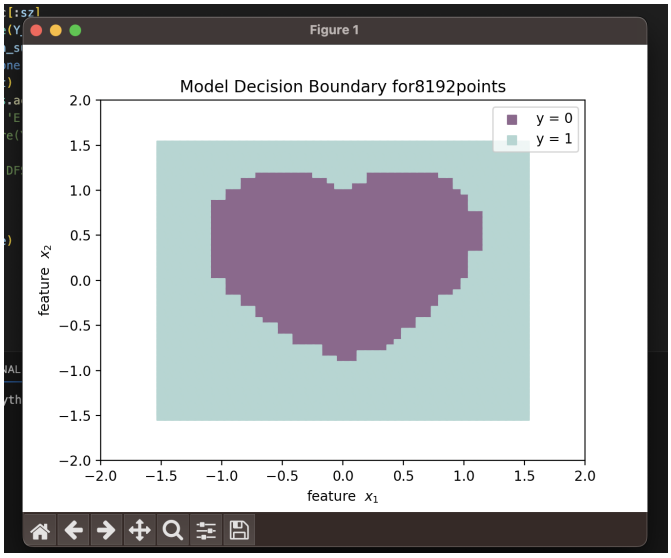
Q2.7.3: Dbig.txt 512 points



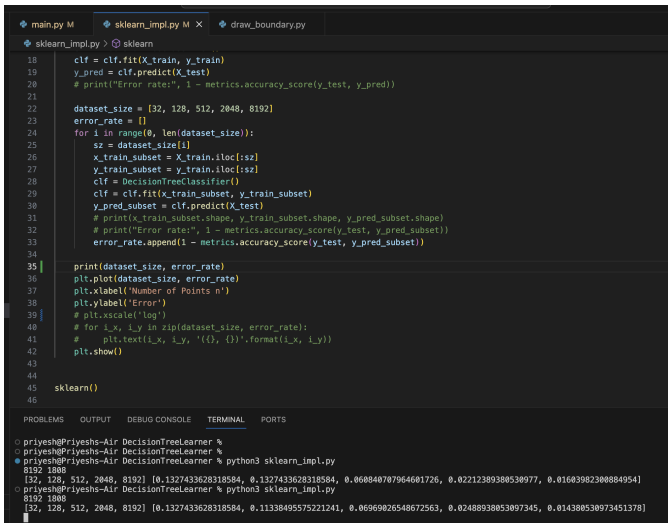
Q2.7.3: Dbig.txt 2048 points

3 sklearn [10 pts]

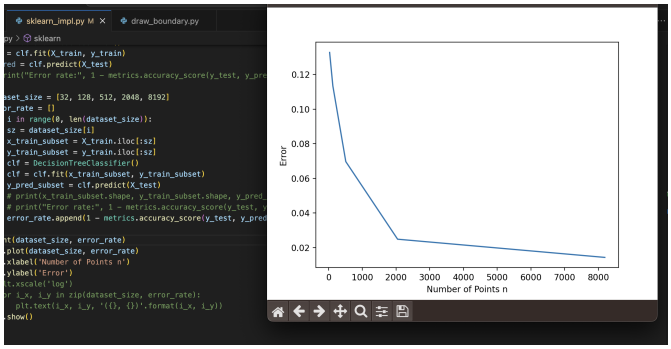
Learn to use sklearn (https://scikit-learn.org/stable/). Use sklearn.tree.DecisionTreeClassifier to produce trees for datasets $D_{32}, D_{128}, D_{512}, D_{2048}, D_{8192}$. Show two things in your answer: (1) List n, number of nodes in that tree, err_n. (2) Plot n vs. err_n.



Q2.7.3: Dbig.txt 8192 points



Q3.1: SKLearn Error value for differnt points



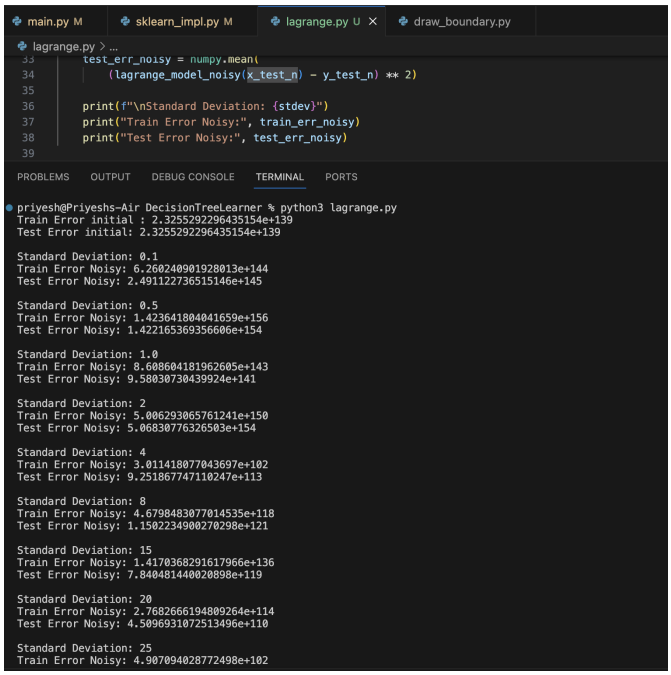
Q3.1: SKLearn Error plot

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.



Lagrange error trend

Initially without the noise introduction, it is observed that the test error is insanely high in orders of e^{139}. This is because Lagrange works well only upto 20 points as mentioned on the documentation. Then we introduce noise in the system. We observe that the test error first increases and then decreases for higher standard deviation values. Similarly, the training error increases and then decreases as well when noise is introduced. The error calculation seems to oscillate between value in terms of higher degree interpolation. This can not be viewed as a trend when we have huge number of points like 100.