HOMEWORK 2

>>Hemanth Sridhar Nakshatri<<
>>nakshatri@wisc.edu<<

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:

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

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 $x_j \geqslant c$.
- 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.

   Consider two hypotheses:

   - **Hypothesis H1**: Stop the splits when all training items are of same class.
   - **Hypothesis H2**: Determine some thresholds, $T_n = t_1, t_2, ...t_n$ such that further split is forced with $m_1, m_2, ...m_n$ elements in each split group.

   Entropy is given by, $H(S) = -\sum_{i=1}^{c} p_i * log_2(p_i)$

   For ***H1***, entropy is given by, $H_1(S) = -0 * log_2(0) - 1log_2(1) = 0$

   Similary, for ***H2***, Entropy of each split is 0.
   Thus, $H_2(S) = 0$.

   Information gain is $H_1(S) - H_2(S) = 0$.

   Thus, there is no additional gains from splitting further if all samples in a node belong to the same class. Since both give the same result, the simplest solution is usually the best solution i.e, No further splits for that node.

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.

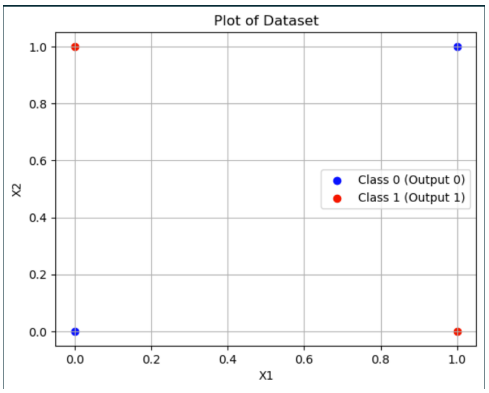

Figure 1: Plot for 2.2

x_1	x_2	Outcome
0	0	0
0	1	1
1	0	1
1	1	0

Table 1: Dataset

Consider the data points shown in Fig. 1. The dataset could be depicted as shown in Table. **??**.

The probability of 1's and 0's outcome for this dataset is 0.5 and 0.5 respectively.

Thus, the entropy of the dataset is given by,
$H(S) = -p_0 log_2(p_0) - p_1 log_2(p_1)$
$H(S) = 0$

The Decision tree algorithm will not be able to split the dataset with regular thresholding without error and will thus make the root node, a leaf node.

We could **force the decision tree to make splits based on XOR gate logic** and it will result in a deeper tree with zero training error.

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.
IG = Information Gain,
GR = Info Gain Ratio For feature x1,

   - Threshold = 0.1 $\Longrightarrow$ IG = 0.4417

   For feature X2,

   - Threshold = -1.0 $\Longrightarrow$ IG = 0.04417
   - Threshold = 0.0 $\Longrightarrow$ GR = 0.05595
   - Threshold = 1.0 $\Longrightarrow$ GR = 0.00578
   - Threshold = 2.0 $\Longrightarrow$ GR = 0.00108
   - Threshold = 3.0 $\Longrightarrow$ GR = 0.01631
   - Threshold = 4.0 $\Longrightarrow$ GR = 0.04945
   - Threshold = 5.0 $\Longrightarrow$ GR = 0.1052
   - Threshold = 6.0 $\Longrightarrow$ GR = 0.1996
   - Threshold = 7.0 $\Longrightarrow$ GR = 0.0383
   - Threshold = 8.0 $\Longrightarrow$ IG = 0.18905

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 tree built using the **Graphviz library (Digraphs will be uploaded to the github repo)** looks as shown in Fig. 2. Leaf nodes represents the class it belongs to. Manually, the logic for the tree can be written as:

   if $X1 \geqslant 10.00$:
   Make Label = 1

   else:
   if $X2 \geqslant 3$:
   Make Label = 1
   else:
   Make Label = 0

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.

[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.

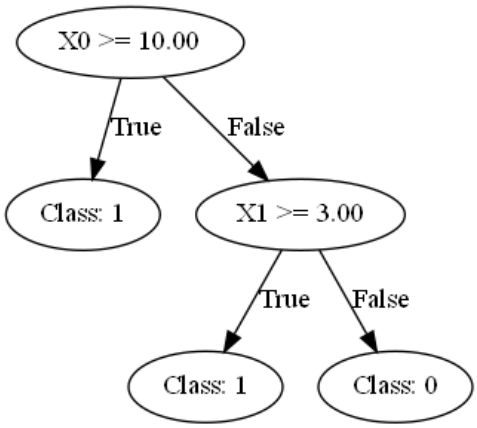

Figure 2: Plot for 2.4

- 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.

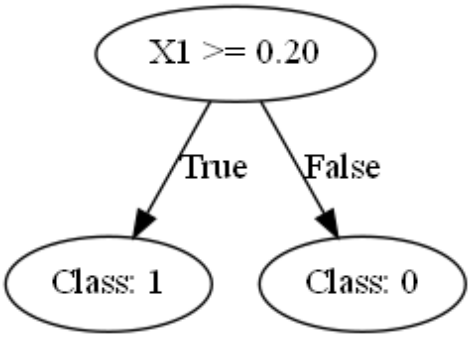

Figure 3: Plot for 2.5.1

Fig. 3 represents the decision tree plot for "D1.txt".

- 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.
  For "D1.txt", From Fig. 3, it can be easily concluded that any instance with feature X2 that has it's value greater than 0.20 will be assigned Class 1. i.e,
  $if\ X2 \geqslant 0.20,\ then\ Class = 1$
  $else : Class = 0.$
  Thus the decision boundary will be a horizontal line at X2 = 0.20.
- Build a decision tree on D2.txt. Show it to us.
  Fig. 4 denotes the tree for the dataset "D2.txt".
- Try to interpret your D2 decision tree. Is it easy or possible to do so without visualization?

While it is not impossible to derive the logical equations to achieve the split, it is impractical to derive the logical solution or explanation for the classification of "D2.txt". It is clearly evident in Fig. 4 as well.

6. (Hypothesis space) [10 pts] For D1.txt and D2.txt, do the following separately:
   - Produce a scatter plot of the data set.

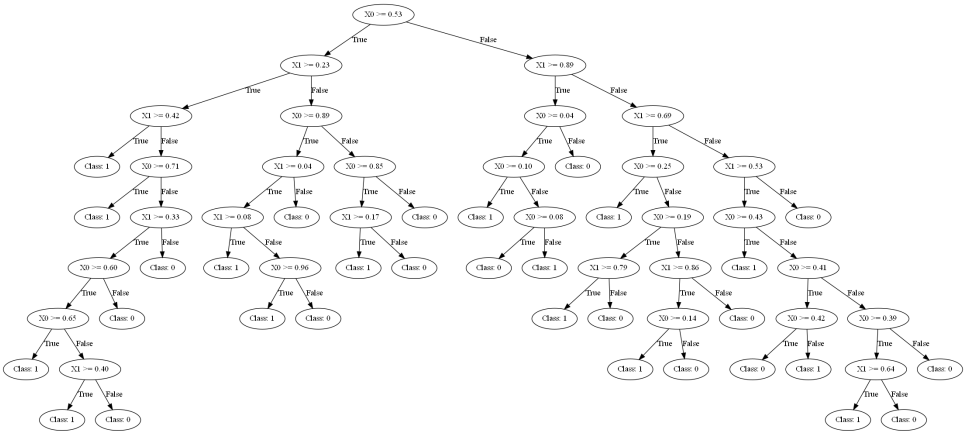

Figure 4: Plot for 2.5.2

- 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).

Figure. 5 denotes the scatter plot of the datapoints and the corresponding boundary for "D1.txt".

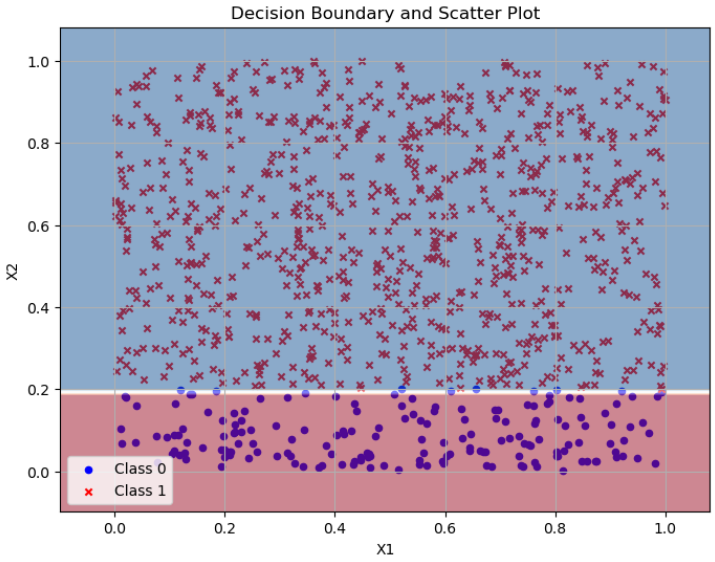

Figure 5: Plot for 2.5.3

Figure. 6 denotes the scatter plot of the datapoints and the corresponding boundary for "D2.txt".

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.
As depicted in Q2.5 and Fig. 3 and Fig. 5, for "D1.txt", the decision boundary is pretty simple to interpret due to it being a horizontal line. It can be seen that the decision boundary performs very good as well. We can see that most of the data points of Class 1 is above the decision boundary (i.e, Class 1 : $X2 >= 0.20$).

For the "D2.txt" dataset, it can be seen that Class 0 points tend to occupy the lower diagonal half of the space and Class 1 occupies upper diagonal half of the space. It can be noted that the decision boundary is seemingly moving along a straight line with negative slope. The jagged boundary of the decision tree might be due to it's ability to create a boundary that is horizontal or vertical (Due to thresholding decision). The boundary might be finer if there were a significantly larger samples in the space.

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

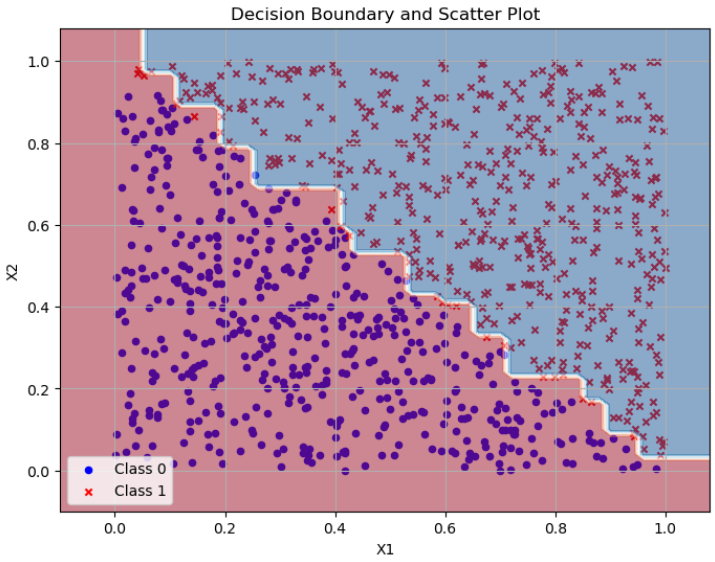

Figure 6: Plot for 2.5.4

- 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 Size	Nodes
32	9
128	12
512	32
2048	77
8192	123

Table 2: Training Size vs. Nodes

- The number of nodes for each tree with n nodes is as shown in Table. 2,
- Plotting n vs err_n, i.e, the the learning curve.
- Plotting Decision boundaries for all the subsets.

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.

Using sklearn, the trees for D_{32}, D_{128}, D_{512}, D_{2048}, D_{8192} are produced. The following results were obtained from it.

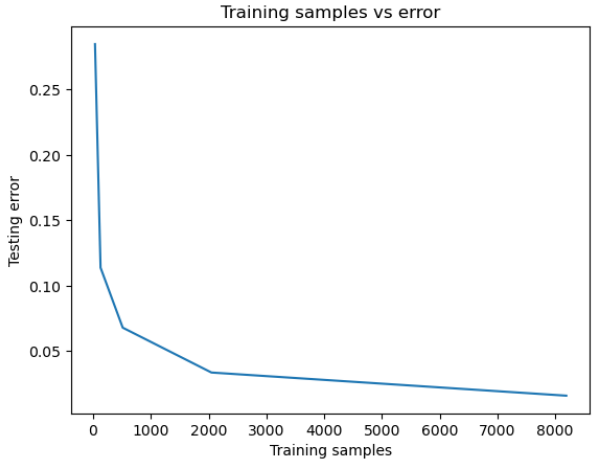

Figure 7: Training error plot with size

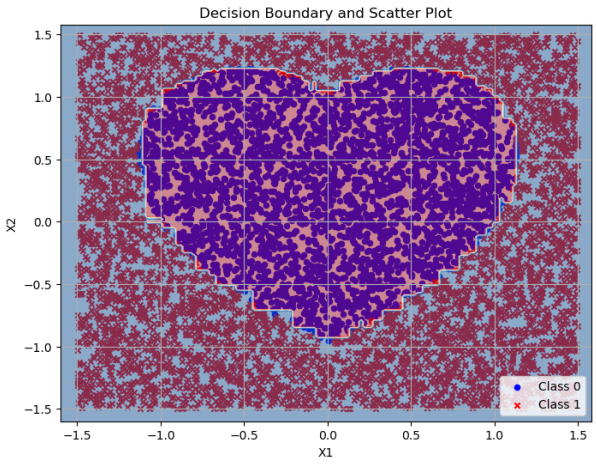

Figure 8: Decision boundary Plot for D_{8192}

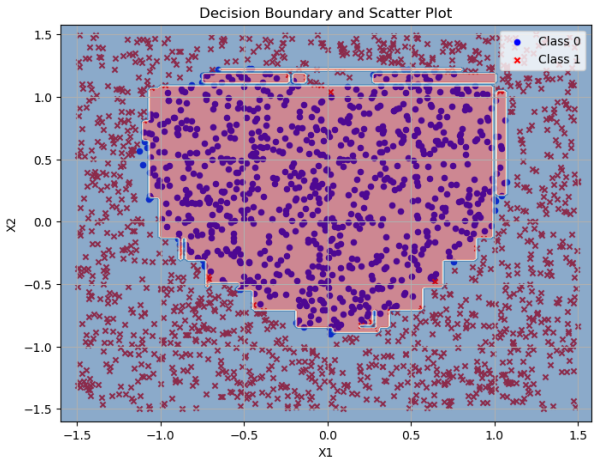

Figure 9: Decision boundary Plot for D_{2048}

- The training size vs the number of nodes n is tabulated and is hown in Table. 3.
- The plots for each subset representing $n\ vs\ err_n$ is plotted in the following figure.

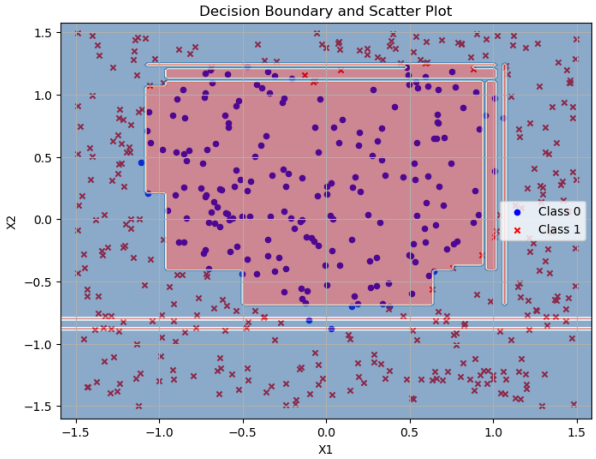

Figure 10: Decision boundary Plot for D_{512}

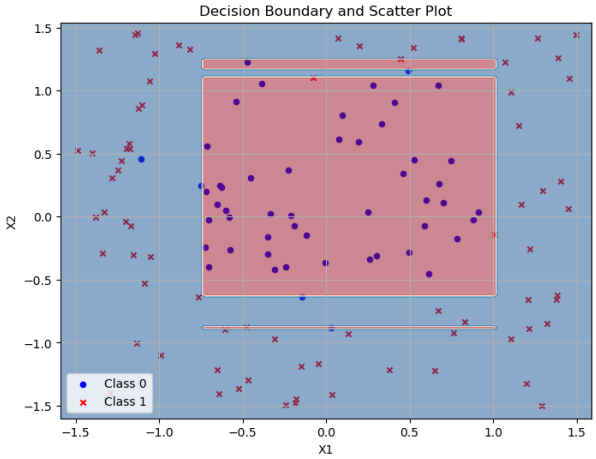

Figure 11: Decision boundary Plot for D_{128}

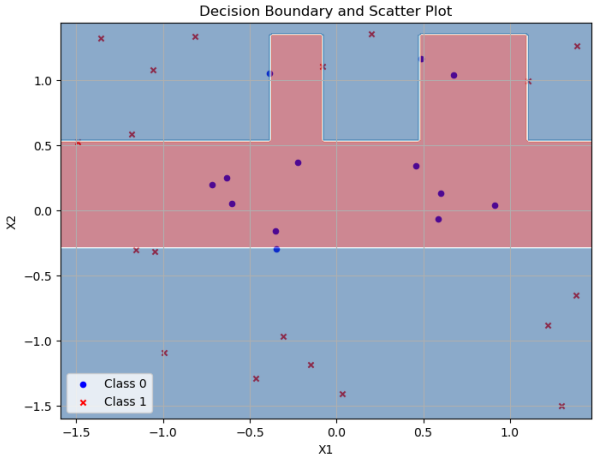

Figure 12: Decision boundary Plot for D_{32}

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 $y = sin(x)$.

Training Size	Nodes
32	7
128	16
512	30
2048	59
8192	112

Table 3: Training Size vs. Nodes (Sklearn)

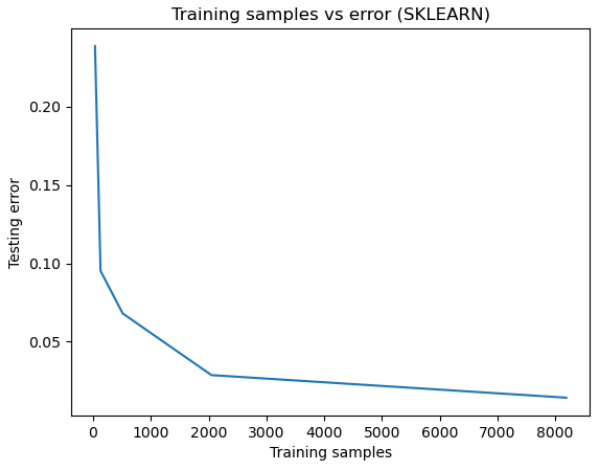

Figure 13: Training error plot with size (SKLearn)

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.

- Fitting the Langragian model to the generated data, we notice that we get the training and testing error with same/similar values.
  - Training error = 2.579×10^{138}
  - Testing error = 2.579×10^{138}
- Howver, when noise is added to the the data x, the error changes,
  - $\epsilon = 0.1$, Error = 3.408×10^{144}
  - $\epsilon = 0.2$, Error = 2.177×10^{140}
  - $\epsilon = 0.3$, Error = 5.003×10^{142}
  - $\epsilon = 0.4$, Error = 6.122×10^{144}
- Figure for noiseless data and noisy data ($\epsilon = 0.1$) has been shown in Fig. 14 and Fig. 15

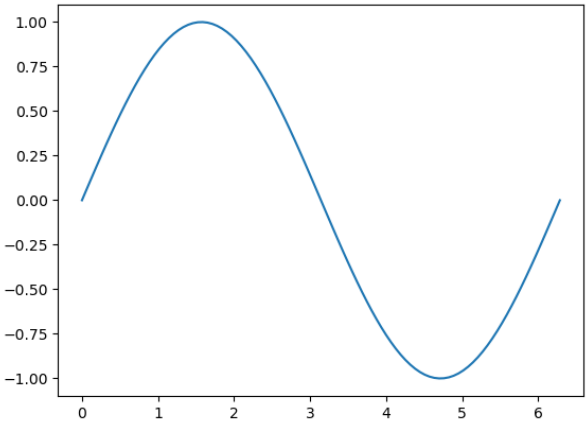

Figure 14: Noiseless data

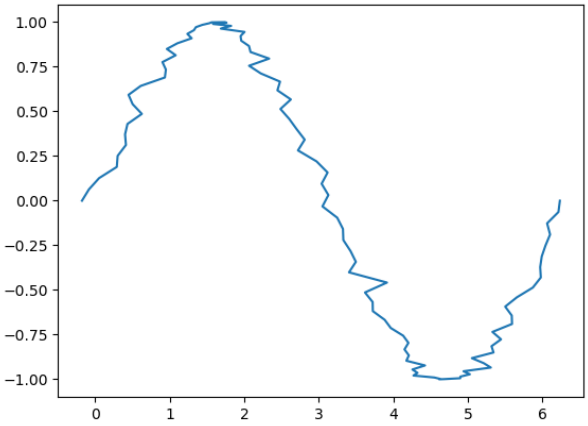

Figure 15: Noisy data ($\epsilon = 0.1$)