HOMEWORK 2

Devansh Goenka
Student ID: 908 335 1354

Answers

Answer 2.1

Lets assume that a non-empty node contains training items of the same label. In this case, we have:

$$I(Y;X) = H(Y) - H(Y|X)$$

$$H(Y) = -\sum_{y \in Y} P(y) log_2(P(y))$$

$$H(Y|X) = \sum_{x \in X} P(X = x) H(Y|X = x)$$

Finally, we have :

$$H(Y|X = x) = -\sum_{y \in Y} P(Y = y|X = x) log_2(P(Y = y|X = x))$$

For the binary class situation:

$$H(Y|X = x) = -P(Y = 1|X = x) log_2(P(Y = 1|X = x)) - P(Y = 0|X = x) log_2(P(Y = 0|X = x))$$

However, since the node contains all the same labels, lets assume $P(Y = 1) = 0$ and $P(Y = 0) = 1$.
Now,

$$H(Y|X = x) = -0 * log_2(0) - 1 * log_2(1) = 0$$

Hence,

$$H(Y|X) = \sum_{x \in X} P(X = x) H(Y|X = x) = \sum_{x \epsilon X} P(X = x) * 0 = 0$$

Irrespective of the feature, the conditional entropy is 0.
Similarly, $H(Y) = -\sum_{y \in Y} P(y) log_2(P(y)) = 0$
Thus, the Informaiton Gain is 0 in this case, and hence the node is expected to become a leaf. [Answer]

Answer 2.2

If we consider the following training set, where we have 2 features X_1, X_2 and the label column is Y:

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

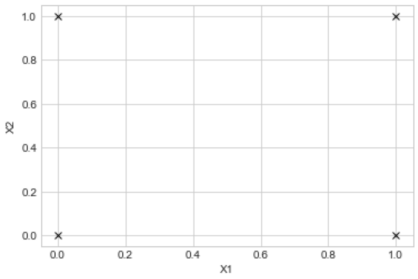
We can see that when training on this, our algorithm will simply stop at the root node and assign the label $Y = 1$. The proof for this is as follows:

$$H(Y) = -\frac{1}{2}log_2(\frac{1}{2}) - \frac{1}{2}log_2(\frac{1}{2}) = 1$$

Moreover, for any candidate split (let's say $X_1 >= 1$), we have the conditional entropy as :

$$H(Y|X) = \frac{1}{2} * 1 + \frac{1}{2}. * 1 = 1$$

In fact, for all such candidate splits, we get the conditional entropy as 1.
Thus, the Information Gain for this training set is always 0, and based on the stopping criteria mentioned, our algorithm would just make this a leaf node and assign the label $Y = 1$ for this.
The plot for this training data is as:



Training data that resembles XOR function

If we force a split, we can get a tree as follows:

$X_1 >= 1$ `and` $X_2 >= 1 \rightarrow predict\ \ 0$
$X_1 < 1$ `and` $X_2 >= 1 \rightarrow predict\ \ 1$
$X_1 >= 1$ `and` $X_2 < 1 \rightarrow predict\ \ 1$
$X_1 < 1$ `and` $X_2 < 1 \rightarrow predict\ \ 0$

Thus, after manually deciding the root node, the algorithm creates this tree and achieves zero training error. This is potentially overfitting to the training set as heuristically, there is no information gain after performing this manual splitting.

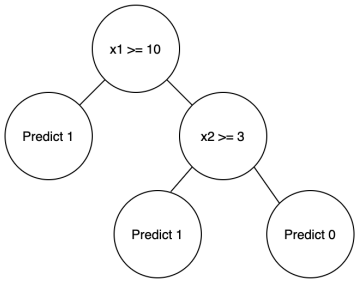
Answer 2.3

The candidate splits and respective information gain ratio and information gain are as follows:

Candidate Split	Gain Ratio	Information Gain
$x2 >= 8$	0.430	0.189
$x2 >= 7$	0.056	0.038
$x2 >= 6$	0.236	0.199
$x2 >= 5$	0.111	0.105
$x2 >= 4$	0.049	0.049
$x2 >= 3$	0.016	0.016
$x2 >= 2$	0.001	0.001
$x2 >= 1$	0.005	0.004
$x2 >= 0$	0.056	0.038
$x2 >= -1$	0.100	0.044
$x2 >= -2$	∞	0
$x1 >= 0.1$	0.100	0.044
$x1 >= 0$	∞	0

Answer 2.4

Here is the tree formed from the file D3Leaves.txt



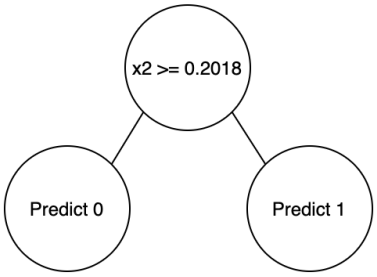
Tree for D3leaves.txt

The logic rules are simple for this:

$if \;\; x_1 >= 10 \rightarrow predict \;\; 1$
$else \;\; if \;\; x_2 >= 3 \rightarrow predict \;\; 1$
$else \rightarrow predict 0$

Answer 2.5

From the file D1.txt, we have the following decision tree classifier:

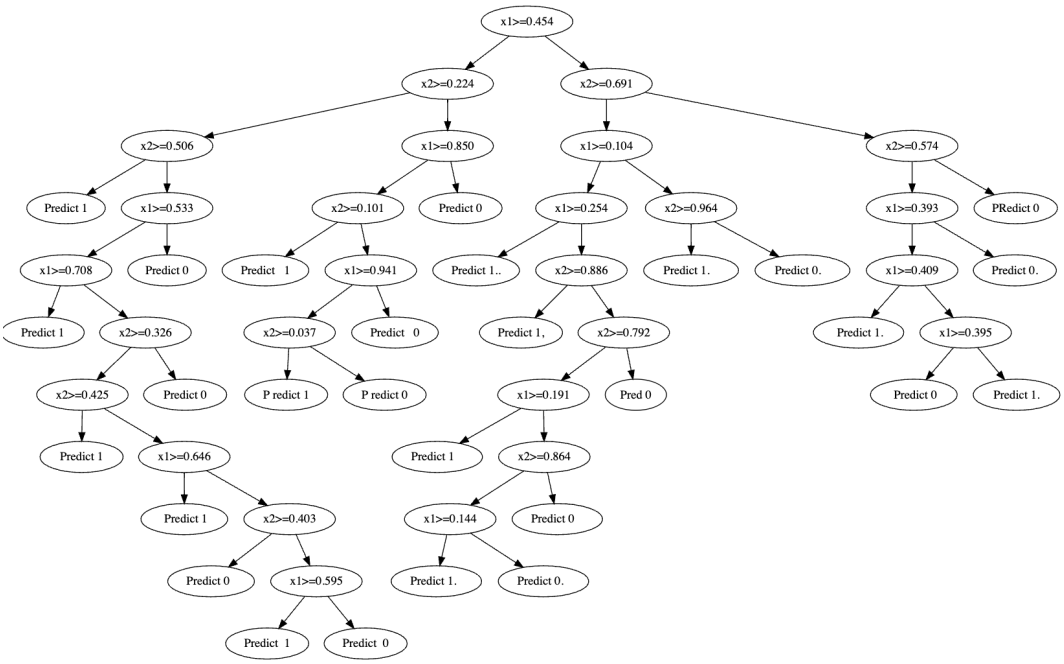


Tree for D1.txt

Without visualizing it, the logic for this can be interpreted in this way:

$if \;\; x_2 >= 0.201 \rightarrow predict \;\; 1$
$else \rightarrow predict \;\; 0$

Based on this result, we can interpret the decision boundary as a line parallel to the axis, dividing the space of x_1 and x_2 with $x_2 >= 0.201$.
From the file D2.txt, we have the following tree:

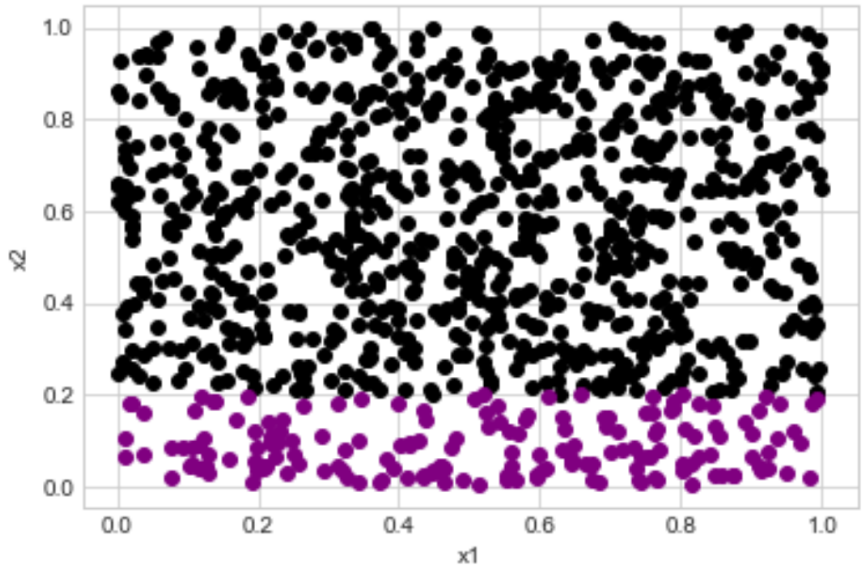


Tree for D2.txt

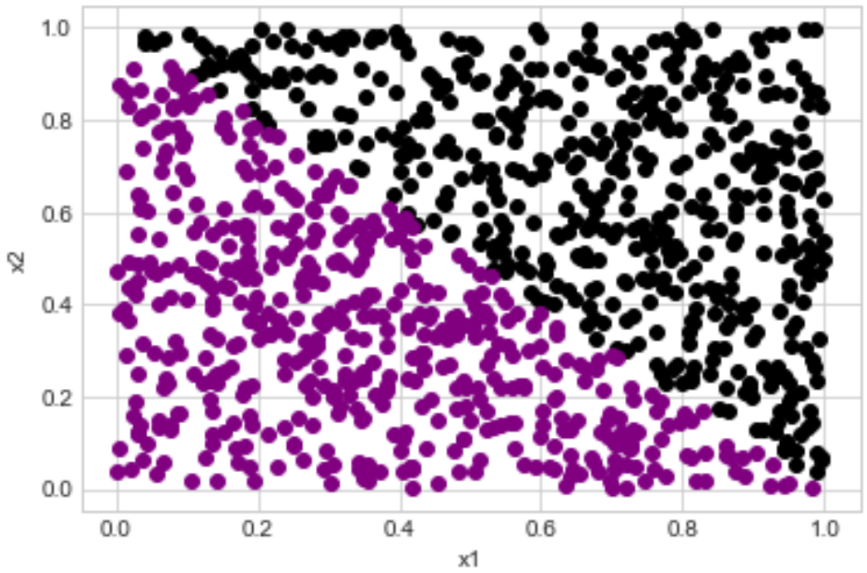
The tree for this looks much more complex than the previous file. It is a little difficult to interpret the decision boundary without visualizing the plot. However, one can say that the training set for this is much more nuanced, with the label depending on multiple thresholds of both features.

Answer 2.6

Here are the scatter plots for the two datasets. The purple points indicate when the label is 0, and the black points indicate when the label is 1.



Scatter plot for D1.txt

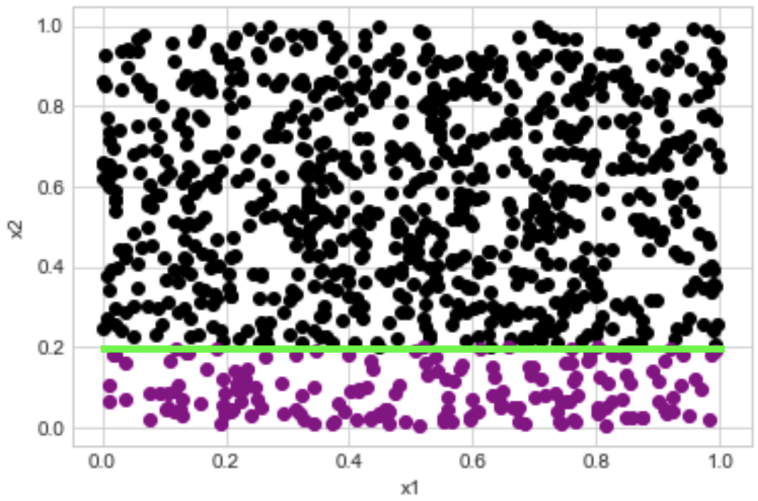


Scatter plot for D2.txt

We can visualize the decision boundary for each of them. For D1.txt, it is a simple line parallel to the

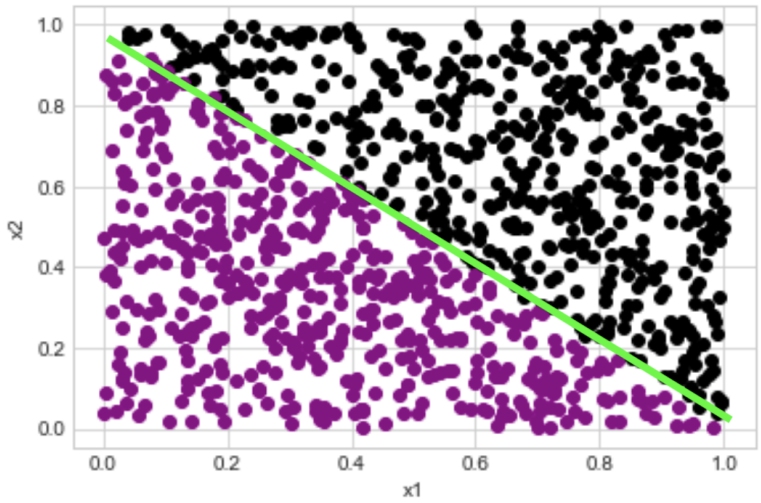
X-axis and signifies the condition:

$$if \;\; x_2 >= 0.201 \rightarrow predict \;\; 1$$
$$else \rightarrow predict \;\; 0$$



Decision Boundary for D1

For D2.txt, the decision boundary is having a negative slope, similar to the line $y = -x$.
One thing to note is that individual segments of the decision boundary are parallel to the x and y axis. However, here the plot is a relaxed version of that decision boundary to better help in visualization.



Decision Boundary for D2

The decision boundaries for D1 and D2 tell us why the size for both of the trees differ. In the case of D1, it is a simple line separating two regions that is enough, and hence we just have 1 node in the tree. For D2, since the boundary is tying to mimic $y = -x$, the threshold and the feature keeps changing to

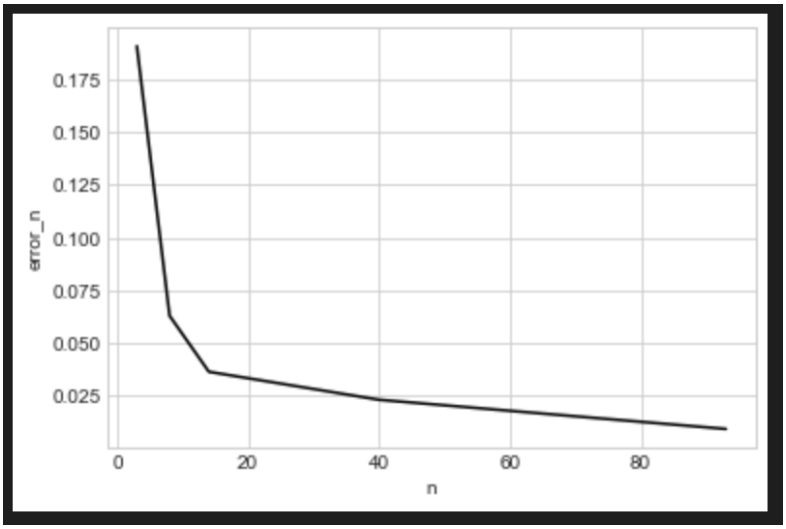
accommodate this motion of going down and right from the top-left corner. Hence, the tree size is much larger.

Answer 2.7

For each of the individual datasets, the Decision Tree is trained on the algorithm specified. Here are the number of nodes in the tree for each set.

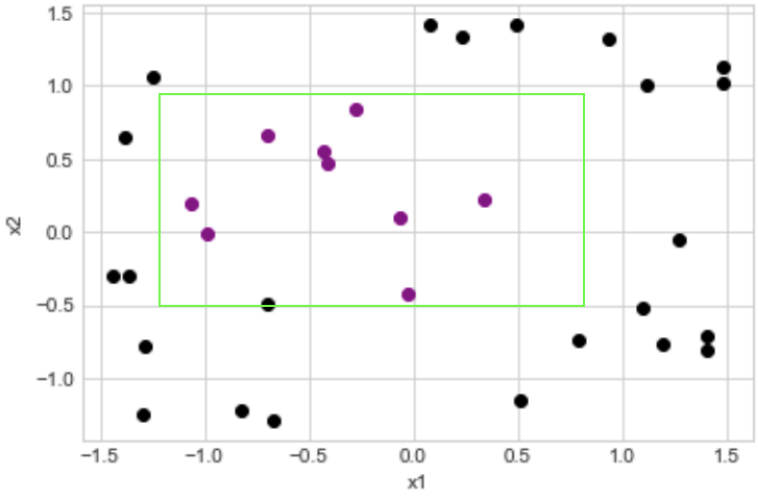
Training Set	Number of Nodes
D_{32}	3
D_{128}	8
D_{512}	14
D_{2048}	40
D_{8192}	93

The number of nodes vs error plot for this as follows:

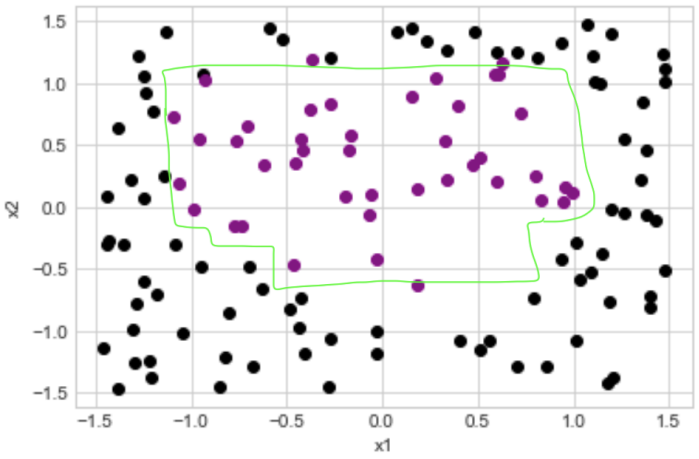


Number of nodes vs Error

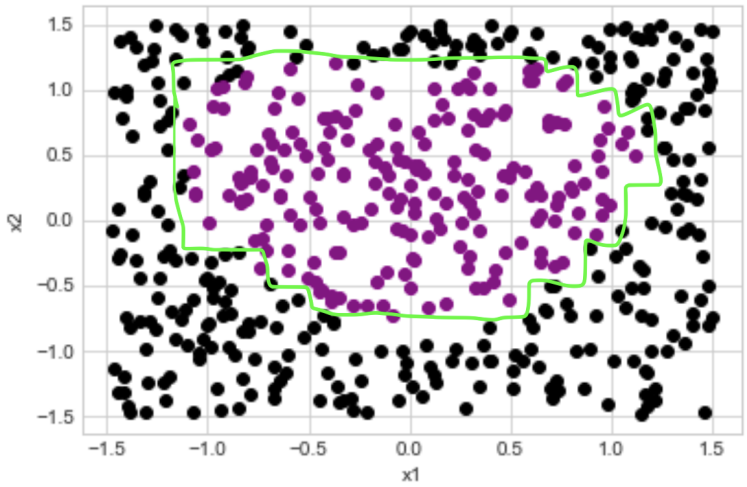
We notice that the error decreases sharply as the nodes increase, and then flattens out. This is consistent with theoretical observations about decision trees as well.
Moreover, here are the decision boundaries for each dataset:



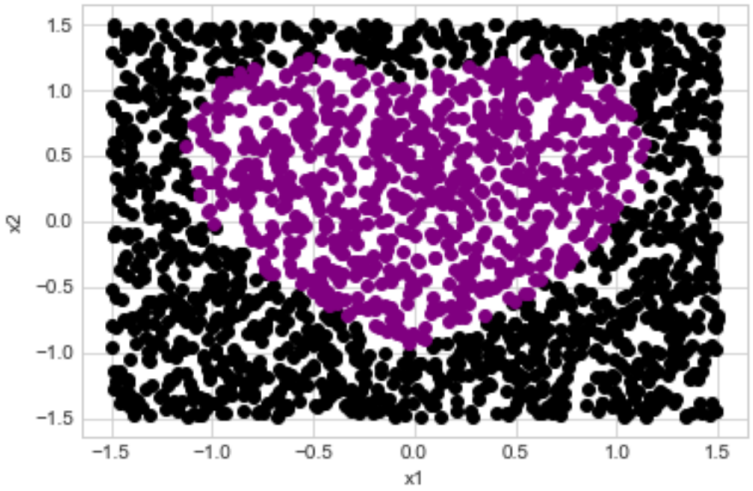
Decision Boundary for D_{32}



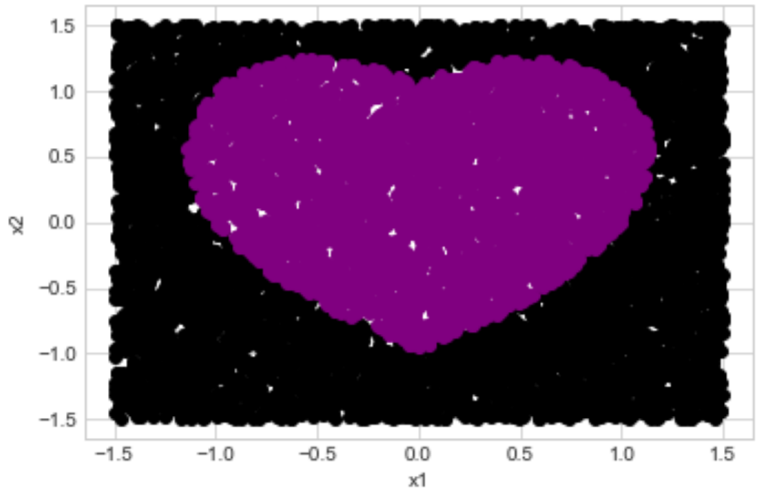
Decision Boundary for D_{128}



Decision Boundary for D_{512}



Decision Boundary for D_{2048}



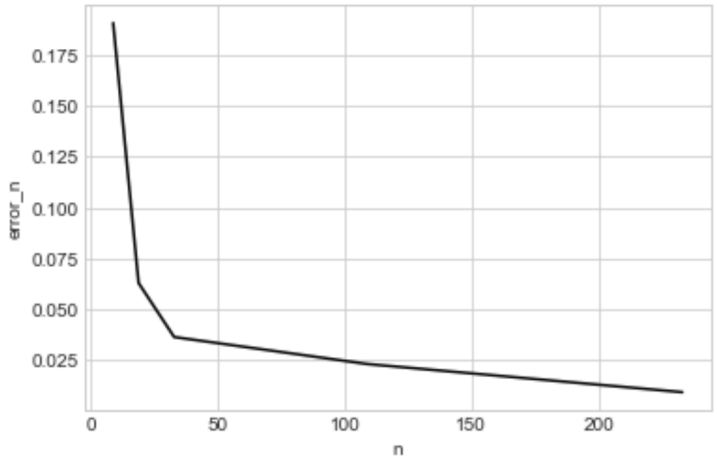
Decision Boundary for D_{8192}

Answer 3

After using `sklearn's` Decision Tree Classifier, Here were the number of nodes for each training sets:

Training Set	Number of Nodes
D_{32}	9
D_{128}	19
D_{512}	33
D_{2048}	109
D_{8192}	233

The number of nodes vs error plot for this as follows:



Number of nodes vs Error

We notice that the error decreases sharply as the nodes increase, and then flattens out. This is consistent with theoretical observations about decision trees as well.

Answer 4

For this question, let the bounds be $[0, 10]$. Moreover, the error used here for reporting is the MAE (Mean Absolute Error).
After sampling 100 points uniformly from this range, and performing Lagrange interpolation, here are the results:

Train Error: $1.2426298372857457e + 68$
Test Error: $3.5259340923635824e + 68$
Note: Although ideally the training error should be 0, floating point error is inducing some error here which is being magnified to this large extent because of coefficient of the order 44 in the Lagrange polynomial. Also, sampling 100 points is not ideal as this function is numerically unstable as stated in `sklearn`'s documentation.
After adding Gaussian noise with a mean of 0 and variance 1, here are the results:
Train Error: $2.285069442486491e + 70$
Test Error: $5.084587742799037e + 67$
With mean 0 and variance 10, the results are:
Train Error: $9.865928902888427e + 65$
Test Error: $3.563889962553354e + 65$
Finally, with mean 0 and variance 50, the results are:
Train Error: $9.680045818333048e + 55$
Test Error: $5.79185303030396e + 71$
Overall, it looks like the errors are large and consistent across test sets. The error does tend to go down by a couple of orders in magnitude as the variance increases, but go back up again when the variance is 50.