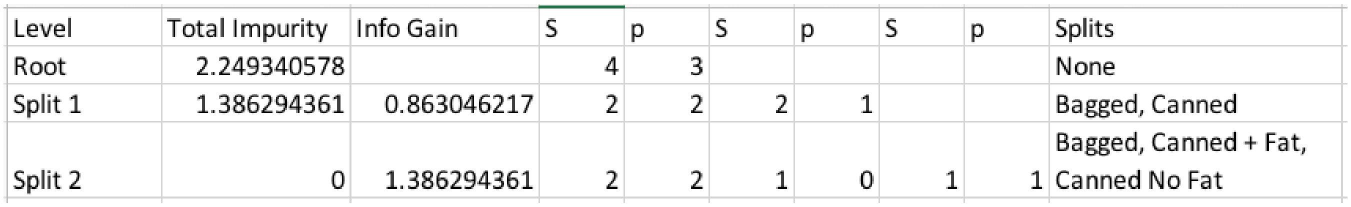
CS155 Set 3

Timothy Liu

January 25, 2018

1 Problem 1

1.1 Problem A



Level	Total Impurity	Info Gain	S	p	S	p	S	p	Splits
Root	2.249340578		4	3					None
Split 1	1.386294361	0.863046217	2	2	2	1			Bagged, Canned
Split 2	0	1.386294361	2	2	1	0	1	1	Bagged, Canned + Fat, Canned No Fat

Figure 1: Impurity at each step of decision tree using log entropy.

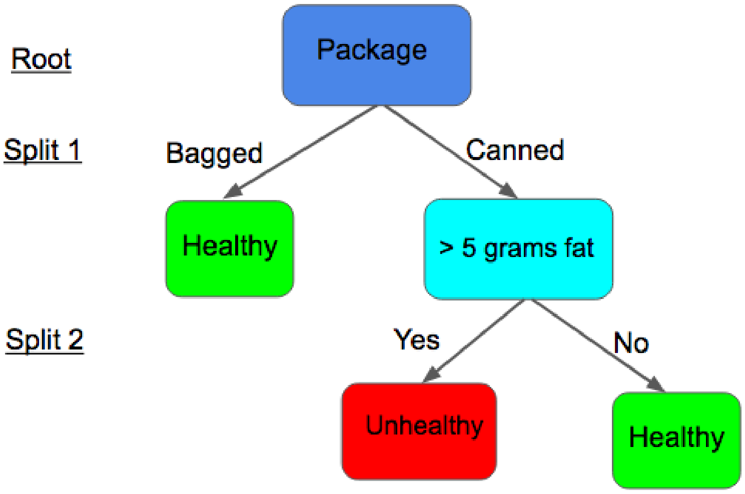


Figure 2: Illustration of decision tree.

1.2 Problem B

No, a decision tree is not always preferred for classification. There are some datasets which are easier to classify linearly but difficult to do with a decision tree.

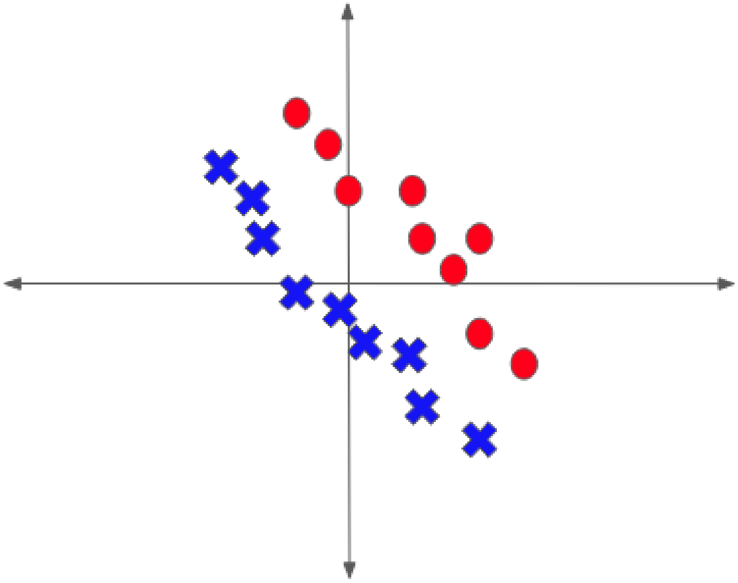


Figure 3: Dataset easy to separate with linear classifier and more complex for a decision tree.

1.3 Problem C

1.3.1 i

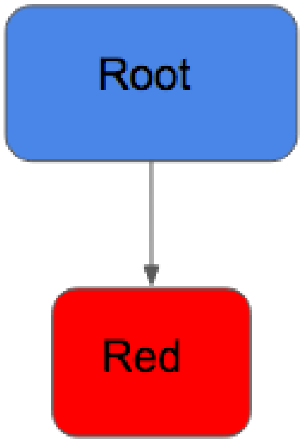


Figure 4: Decision tree trained using Gini index.

There are no splits that reduce the Gini index. At the root, half of the points are misclassified. All of the possible first splits will have two branches that have half of the points misclassified, so there is no reduction in the Gini Index. The decision tree may choose to label all of the points red or all of the points green. The classification error is $\frac{1}{2}$.

Yes. The problem with the Gini index is that it cannot break ties; the decision tree stalls because the first split results in two nodes that sum together with the same impurity as the root node. An impurity function where impurity scales with and falls with the number of points could generate the above decision tree. Such a function would not end with a tie, since the splits would have a smaller impurity because each node has a smaller number of points. The pro of such an approach is that it focuses more on the number of misclassified points rather than the fraction, so it places less weight on further splitting nodes that may have high impurity but sparse points. The con is that such an approach would have results dependent on the size of the training set.

1.3.2 ii

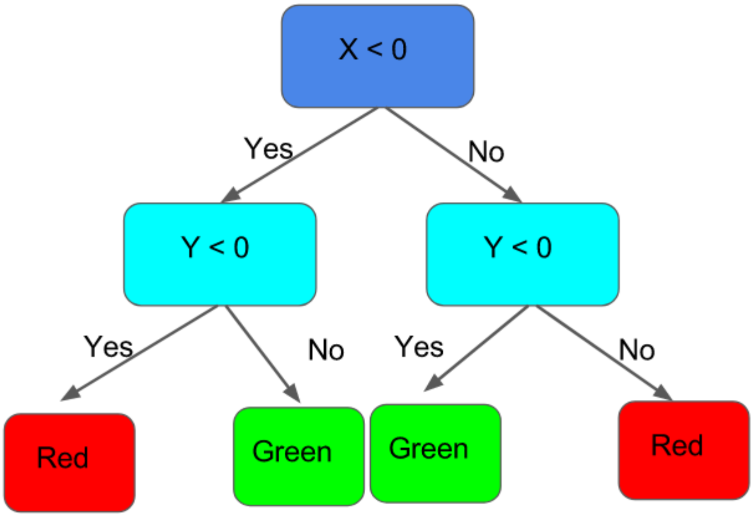


Figure 5: 2 level decision tree that perfectly categorizes the data.

1.3.3 iii

The most difficult 2D data set to train is a 10 by 10 checkerboard array where each point is classified differently from the points directly left, right, above, and below it. 98 internal nodes are needed to perfectly classify such a set. Since each split in the final tree will only contain a single point, a node is needed to split off every point. One node is the root node, and an additional 98 internal nodes are needed to split the remaining points.

1.4 Problem D

The worst case scenario for training time is $O(N \times D)$. There are up to N-1 ways to split a dataset with N points, since a decision boundary can be between any two adjacent points. This process must be repeated across all D dimensions, so the worse case is the product $(N-1) \times D$ which asymptotically approaches $O(N \times D)$.

2 Problem 2

2.1 Problem A

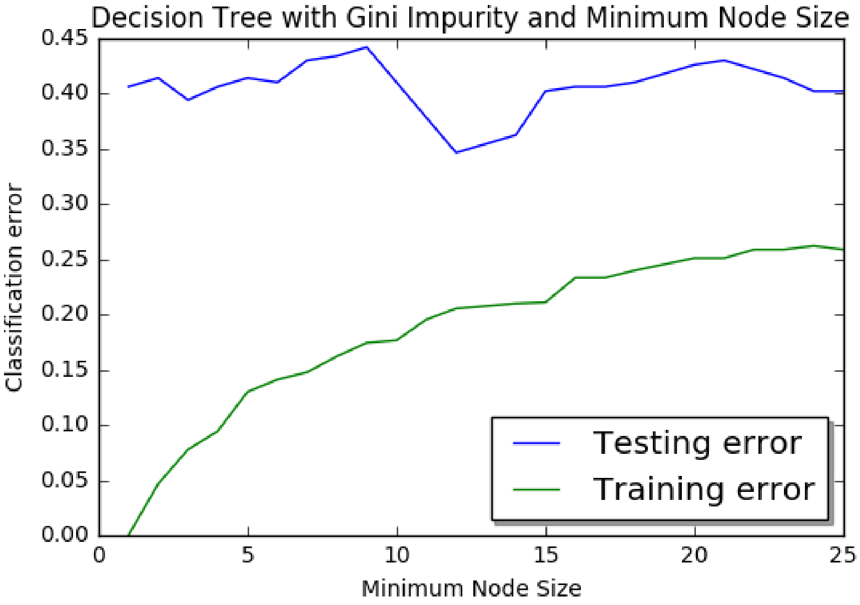


Figure 6: Decision tree with varying minimum leaf node size.

As the minimum node size increases, the training error generally increases. However, the testing error hits a minimum at a minimum node size of 13.

2.2 Problem B

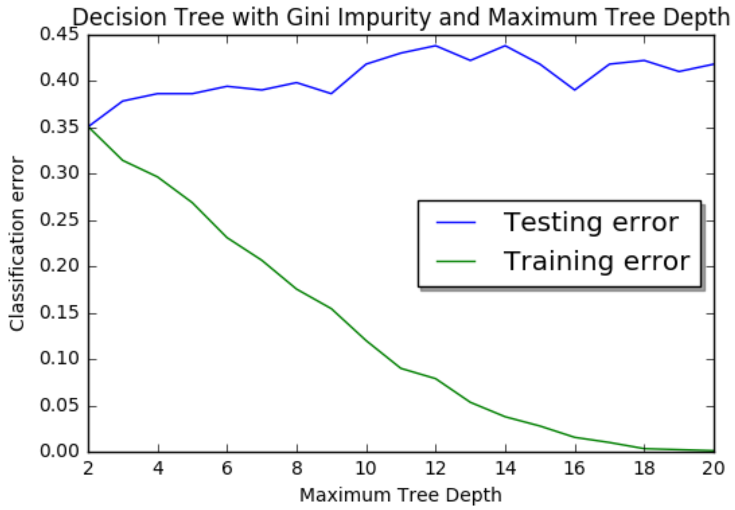


Figure 7: Decision tree with varying maximum tree depth.

As maximum tree depth increases, the training error falls to zero. However, the testing error is lowest when the maximum tree depth is only 2.

2.3 Problem C

Training error is minimized when the minimum node size is 13 and the maximum tree depth is 2. Early stopping increases the testing performance of a decision tree model. When the stopping condition is relaxed (minimum node size goes to 1 and maximum depth increases) the resultant model has poor testing performance. The test error generally increases as the maximum tree depth constraint is relaxed and as the minimum node size falls.

2.4 Problem D

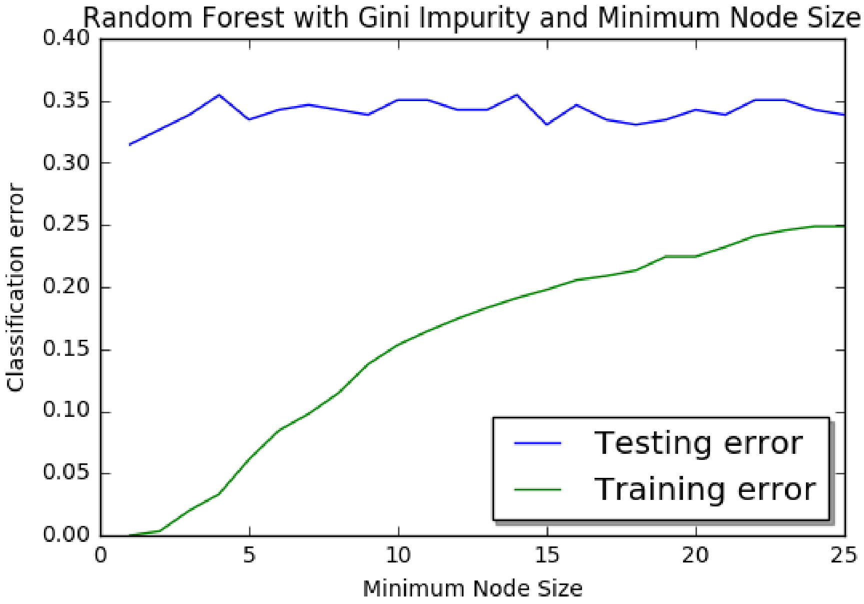


Figure 8: Decision tree with varying minimum leaf node size.

2.5 Problem E

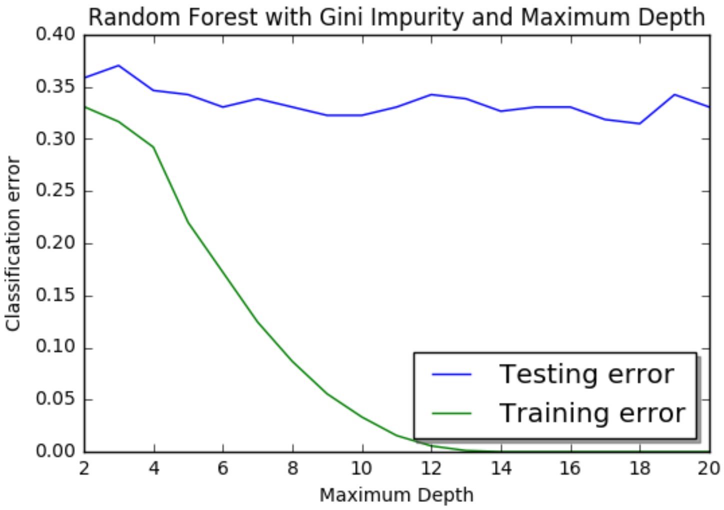


Figure 9: Decision tree with varying maximum tree depth.

2.6 Problem F

Training error is minimized when the minimum node size is 1 and the maximum tree depth is 18. Early stopping seems to have little effect on the performance of the random tree. The testing error is fairly constant for both types of early stopping.

2.7 Problem G

The curves for decision trees and random forests have pretty similar general shapes. The decision trees seem to have higher training error as the early stopping constraint is relaxed, but the effect is fairly small. The errors for the random forests are also generally lower than for decision trees. For reasons unknown....

3 Problem 3

3.1 Problem A

We will show that exponential loss bounds 0/1 loss.

$$E = \frac{1}{N}\sum_{i=1}^{N} exp(-y_i f(x_i))$$

$$= \frac{1}{N}\sum_{i=1}^{N} exp\Big\{-y\{\sum_{t=1}^{n} a_t h_t(x)\}\Big\}$$

$$= \frac{1}{N}\sum_{i=1}^{N}\prod_{t=1}^{n} exp\{-y\alpha_t h_t(x)\}$$

When $f_(x)$ is negative, the product $-y_i f(x_i)$ is positive, so the exponential term $exp(-y_i f(x_i)$ is positive and greater than one. The product of positive numbers greater than one is also positive and greater than one, which bounds 0/1 loss for when $f(x) < 0$. When $f(x)$ is exactly 0, then the exponential term is 1 and the product across n is 1. This is equivalent to zero one loss when $f(x) = 0$. Finally, when $f(x) < 1$ the each exponential term must still be positive, so the product is also positive and thus greater than 0.

3.2 Problem B

$$D_{t+1}(i) = \frac{D_t(i)exp\{-\alpha_t y_i h_t(x_i)\}}{Z_t}$$

$$D_{t+1}(i) = \frac{D_{t-1}(i)exp\{-\alpha_{t-1} y_i h_{t-1}(x_i)\}}{Z_{t-1}} \frac{exp\{-\alpha_t y_i h_t(x_i)\}}{Z_t}$$

$$D_{t+1}(i) = \frac{D_{t-2}(i)exp\{-\alpha_{t-2} y_i h_{t-2}(x_i)\}}{Z_{t-2}} \frac{exp\{-\alpha_{t-1} y_i h_{t-1}(x_i)\}}{Z_{t-1}} \frac{exp\{-\alpha_t y_i h_t(x_i)\}}{Z_t} ....$$

$$D_{t+1}(i) = D_1(i) \prod_{t=1}^{T} \frac{exp\{-\alpha_t y_i h_t(x_i)\}}{Z_t}$$

$$D_{t+1}(i) = \frac{1}{N} \prod_{t=1}^{T} \frac{exp\{-\alpha_t y_i h_t(x_i)\}}{Z_t}$$

3.3 Problem C

$$E = \frac{1}{N} \sum_{i=1}^{N} exp(-y_i f(x_i))$$

$$E = \sum_{i=1}^{N} \frac{1}{N} exp(-y_t f(x_i))$$

Substitute $f(x) = \sum_i^T \alpha_t h_t(x)$

$$E = \sum_{i=1}^{N} \frac{1}{N} exp(-y_i \sum_{t}^{T} \alpha_t h_t(x_i)))$$

$$E = \sum_{i=1}^{N} \frac{1}{N} exp(-\sum_{t}^{T} y_i \alpha_t h_t(x_i)))$$

3.4 Problem D

$$E = \sum_{i=1}^{N} \frac{1}{T} exp(-\sum_{t}^{T} y_i \alpha_t h_t(x_i)))$$

$$= \frac{1}{N} \sum_{i=1}^{T} \prod_{t=1}^{n} exp\{-y \alpha_t h_t(x_i)\}$$

$$= \sum_{i=1}^{T} D_t \prod_{t=1}^{T} Z_t$$

$$E = \prod_{t=1}^{T} Z_t$$

3.5 Problem E

$$Z_t = \sum_{i=1}^{N} D_t(i) exp(-\alpha_t y_i h_t(x_i))$$

We split this into where the points are correctly and incorrectly classified:

$$Z_t = (1 - \epsilon_t) \sum_{i=1}^{N} D_t(i) exp(-\alpha_t y_i h_t(x_i)) + \epsilon_t \sum_{i=1}^{N} D_t(i) exp(-\alpha_t y_i h_t(x_i))$$

$$Z_t = (1 - \epsilon_t) exp(-\alpha_t) + \epsilon_t exp(\alpha_t)$$

3.6 Problem F

We take the derivative of Z_t with respect to α_t, set the result to zero, and solve for α_t.

$$\frac{\partial Z_t}{\partial \alpha_t} = 0 = -(1 - \epsilon_t) exp(-\alpha_t) + \epsilon_t exp(\alpha_t)$$

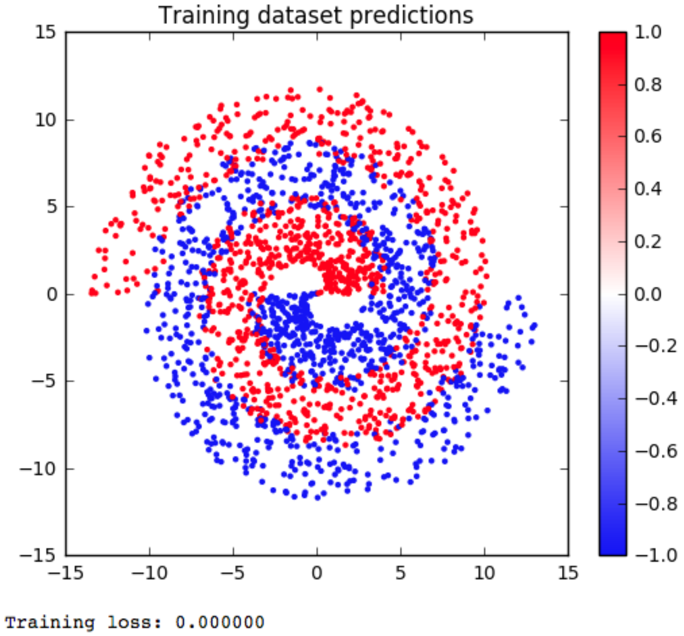
$$\frac{1 - \epsilon_t}{\epsilon_t} exp(-\alpha_t) = exp(\alpha_t)$$

$$\frac{1 - \epsilon_t}{\epsilon_t} = exp(2\alpha_t)$$

$$\alpha_t = \frac{1}{2} ln(\frac{1 - \epsilon_t}{\epsilon_t})$$

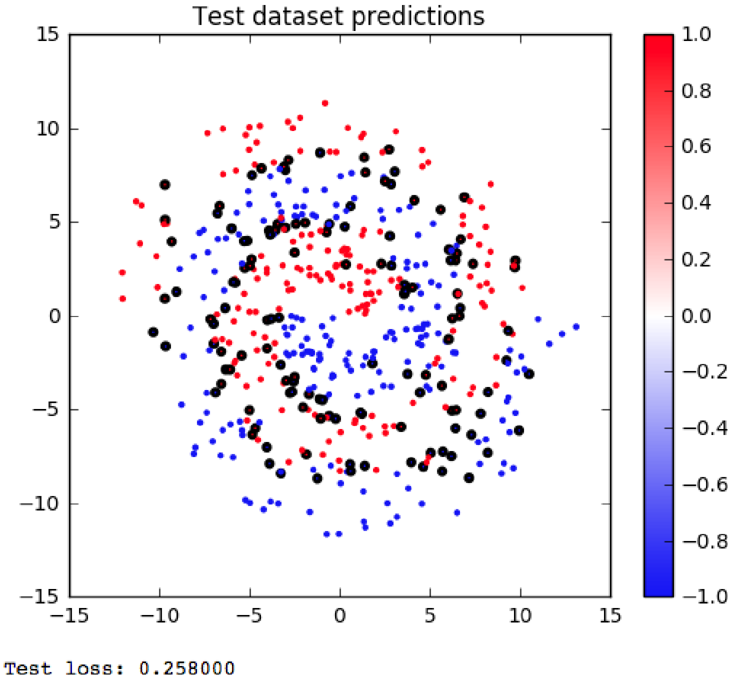
3.7 Problem G

Gradient Boosting



```
Training loss: 0.000000
```

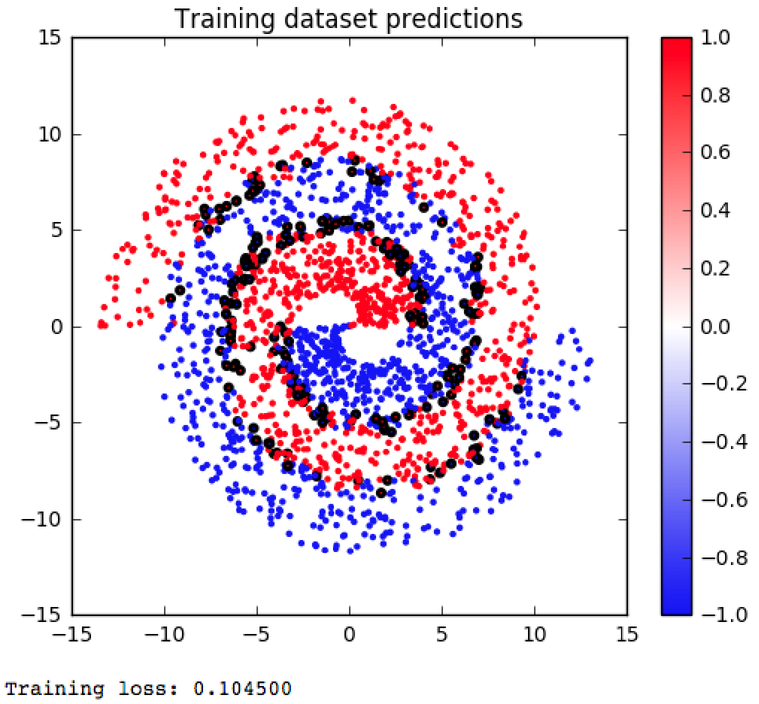
Figure 10: Training dataset for gradient boosting



```
Test loss: 0.258000
```

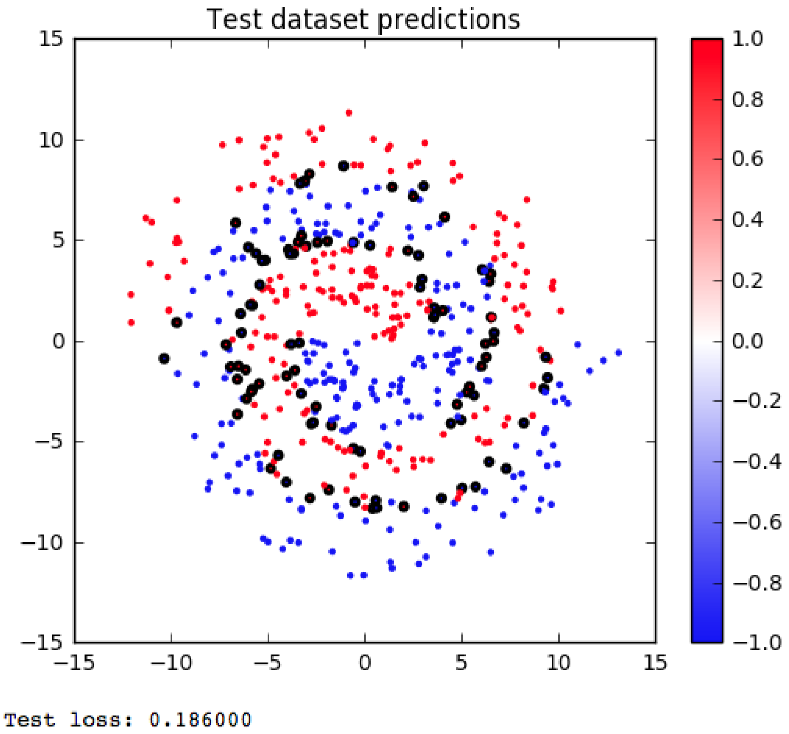
Figure 11: Test dataset for gradient boosting

Adaboost



```
Training loss: 0.104500
```

Figure 12: Training dataset for adaptive boosting.



```
Test loss: 0.186000
```

Figure 13: Testing dataset for adaptive boosting

3.8 Problem H

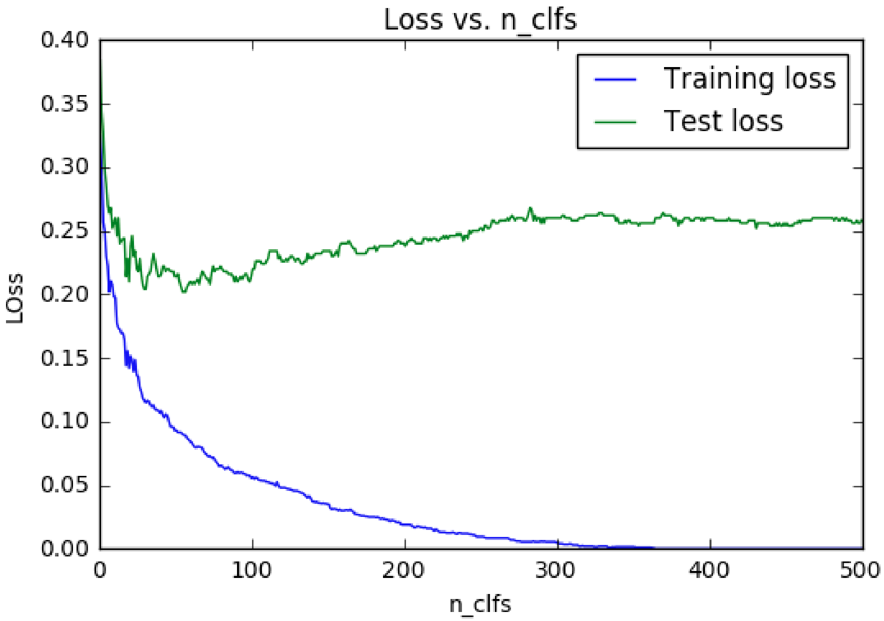


Figure 14: Learning curve for gradient boosting

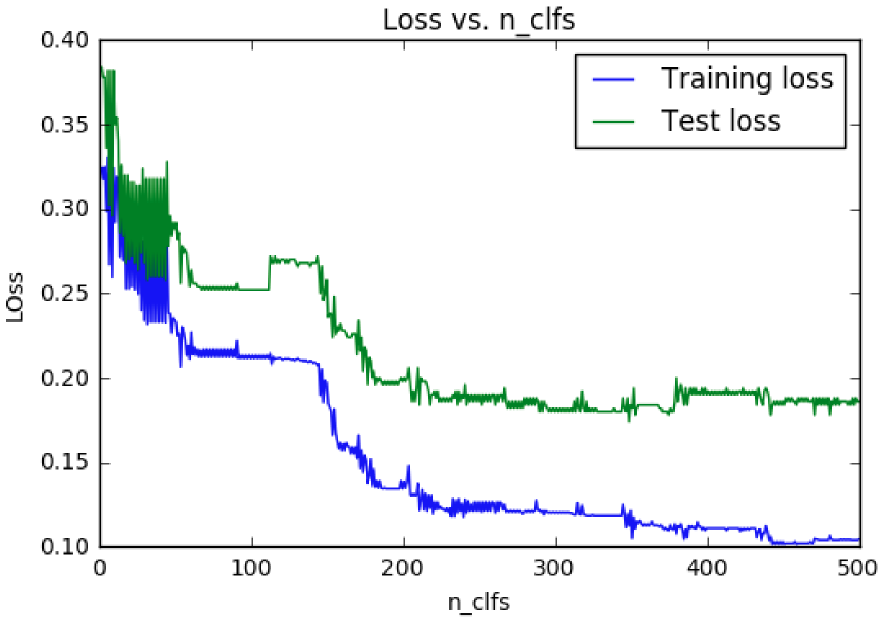


Figure 15: Learning curve for adaptive boosting.

The training loss for gradient boosting starts at about 0.35 and falls steadily towards zero as the number of classifiers increases. The test loss for gradient boosting begins at about the same level, falls to a minimum of about 0.2 when there are about 40 classifiers, and then increases as the number of classifiers increases. As the number of classifiers increases, overfitting occurs. The curves are fairly smooth.

The training loss for adaboost starts at about 0.32 and falls to 0.21 as the number of classifiers rises to about 50. The training loss levels off and then falls again when the number of classifiers reaches 150. After that, the training loss falls quickly to 0.12, and then falls more slowly towards 0.1. The test loss behaves similarly. The test loss begins at about 0.37, falls to 0.25 at 50 classifiers, stabilizes, then falls after 150 classifiers. Afterwards the test loss stabilizes at about 0.2. The adaboost curve is much more jagged and has sharper variations than the gradient boosting curve.

3.9 Problem I

The final training error of gradient boosting is about 0 and the final testing error is 0.26. The final training error of Adaboost is about 0.1 and the testing error is 0.18. The adaptive boosting algorithm performed better on the classification dataset.

3.10 Problem J

The dataset weights are the largest near the center of each spiral arm away from the boundaries while the weights are the smallest at the boundaries between red and blue.