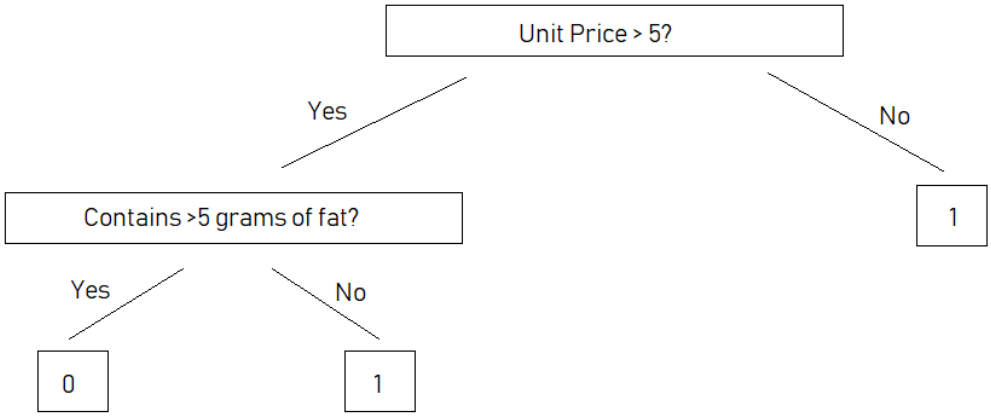
Jessica Wang
Machine Learning and Data Mining
Set 3

I. Decision Trees

Problem A.



In the tree shown above, 1 indicates healthy and 0 indicates unhealthy.
1st split: L(S) = 2.2493
2nd split: L(S) = 1.3863
3rd split: L(S) = 0.0
The first impurity reduction is approximately 0.86 and the second impurity reduction is approximately 1.39.

See Jupyter Notebook for code on calculating entropy, with natural log, as the impurity measure.

At the root:
S_1 = {No, Yes, Yes, Yes} so P_{S1} = ¾ or 0.75
Entropy = 4 * 0.5623 = **2.2493**

At the first level:
Entropy when splitting on 'Package Type':
Bagged: S_1 = {Yes, Yes} so P_{S1} = 1
Canned: S_2 = {No, Yes} so P_{S2} = ½ or 0.5
Entropy = (2 * 0) + (2 * 0.6931) = **1.3863**
Entropy when splitting on 'Unit Price > 5':
Yes: S_1 = {No, Yes} so P_{S1} = 0.5
No: S_2 = {Yes, Yes} so P_{S2} = 1
Entropy = (2 * 0.6931) + (2 * 0) = **1.3863**

Entropy when splitting on 'Contains > 5 grams of fat':
Yes: S_1 = {No, Yes} so P_{S1} = 0.5
No: S_2 = {Yes, Yes} so P_{S2} = 1
Entropy = (2 * 0.6931) + (2 * 0) = **1.3863**

Since all of the entropies were calculated to be the same, any of the splits can be used. Let us split on 'Unit Price > 5'. Since the stopping criterion in this problem is governed by no impurity in the leaves, S_2 does not need to be split further since it is pure.

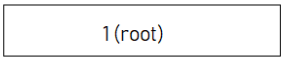
At the second level:
Entropy when splitting on 'Package Type':
Bagged: S_1 = {Yes} so P_{S1} = 1
Canned: S_2 = {No} so P_{S2} = 0
Entropy = (1 * 0) + (1 * 0) = **0**
Entropy when splitting on 'Contains > 5 grams of fat':
Yes: S_1 = {No} so P_{S1} = 0
No: S_2 = {Yes} so P_{S2} = 1
Entropy = (1 * 0) + (1 * 0) = **0**

Problem B.

Compared to a linear classifier, a decision tree is not always preferred for classification problems.

A simple linear classifier can easily find the decision boundary in the example shown above but decision trees require complex axis partitioning.

Problem C.i.



Using the Gini index as the impurity measure:

At the root:
P_{S1} = 1/2 or 0.5
Entropy = 4 * 0.5 = **2.0**

At the first level:

Impurity when splitting on 'X':

X = 1: S_1 = {Yes, No} so P_{S1} = ½ or 0.5

X = -1: S_2 = {No, Yes} so P_{S2} = ½ or 0.5

Entropy = (2 * 0.5) + (2 * 0.5) = **2.0**

Impurity when splitting on 'Y':

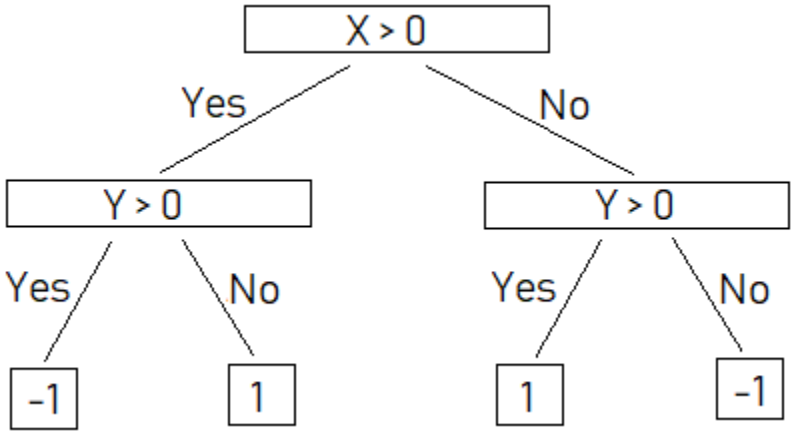
Y = 1: S_1 = {Yes, No} so P_{S1} = ½ or 0.5

Y = -1: S_2 = {No, Yes} so P_{S2} = ½ or 0.5

Entropy = (2 * 0.5) + (2 * 0.5) = **2.0**

We observe that there is no such splitting which would allow for a reduction in impurity which satisfies the stopping criterion of the problem. Everything would be classified as plus or everything would be classified as minus. The classification error, which is the number of misclassified points divided by the number of total points is 0.5.

Problem C. ii.



In the diagram above, "1" indicates "+" and "-1" indicates "-" as given by the diagram in the problem set.

There is an impurity measure given by multiplying the number of positives by the number of negatives or using the Gini index but substituting $|S'|^2$ for the denominator. This impurity measure, a slight modification to the Bernoulli Variance given in lecture, would lead to the same decision tree I drew. The pro is that everything would be classified correctly but the con is that the model would overfit each time.

Problem C. iii.

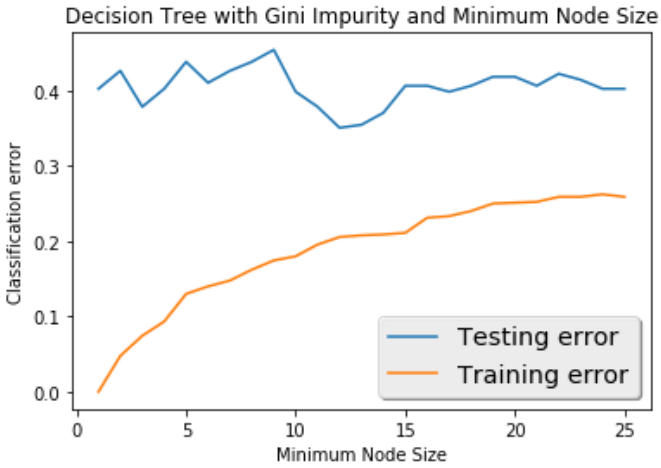
The largest number of unique thresholds necessary to achieve zero classification training error is 99. A decision tree with zero classification error must have as many leaf nodes as data points in the worst case. Therefore, since each data point must be split into its own leaf and a binary tree with L leaf nodes has n = 2L -1 nodes so the decision tree must have 99 internal nodes.

Problem D.

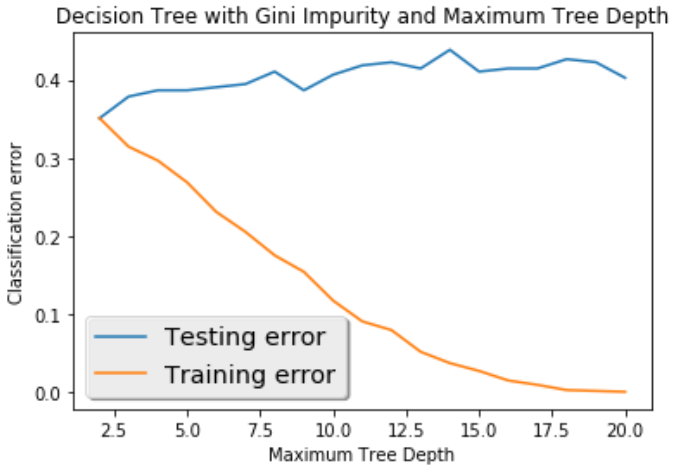
The worst-case complexity of the number of possible splits would be O(DN). Since there are N data points, there are N – 1 possible positions to split them, each of which can be split with one of the D features. While there are an infinite number of possible decision boundaries since we are using continuous features, there are not an infinite number of boundaries that result in unique child sets.

II. Overfitting Decision Trees

Problem A.



Problem B.

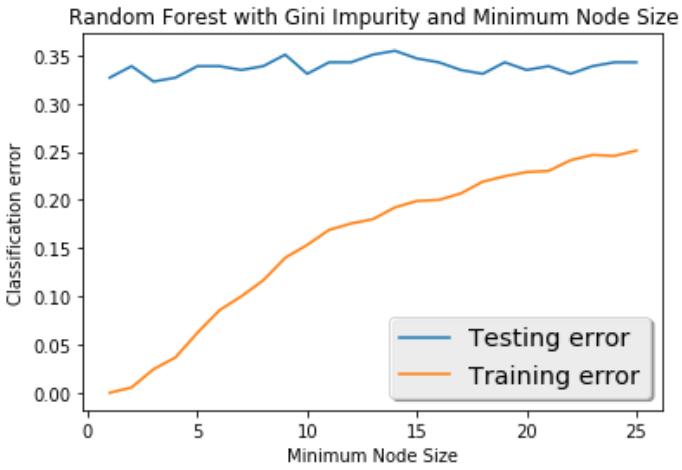


Problem C.

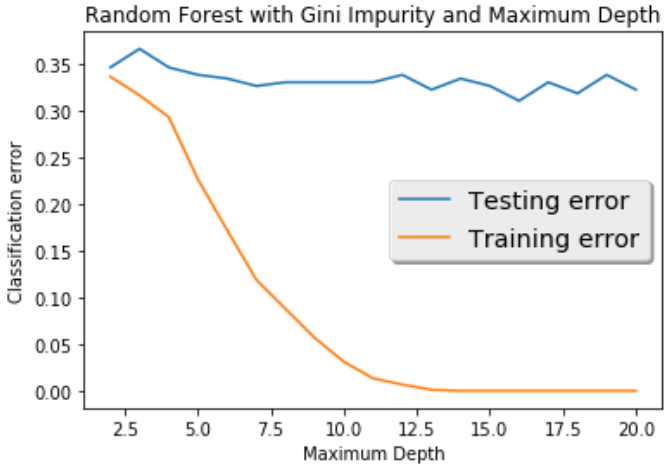
The test error is minimized when the parameter value, min_samples_leaf, is set to 12. The test error is minimized when the parameter value, max_depth, is set to 2. Early stopping helps prevent overfitting but if we stop too early, we begin to underfit since the plots show that the training error and test error will then begin to rise. For a decision tree model where we are concerned with the minimal leaf node sizes, early stopping between 10 to 15 leaves will

cause the testing error to decrease and training error to increase. We observe this through the first plot. On the other hand, for a decision tree model concerned with the maximum tree depth, the model needs to use less splits and stop earlier when the tree depth is lower. In this case, early stopping between 1 to 2 depths allows training error to rise and testing error to fall. Regardless, early stopping prevents overfitting.

Problem D.



Problem E.



Problem F.

The test error is minimized when the parameter value, min_samples_leaf, is set to 1. The test error is minimized when the parameter value, max_depth, is set to 12. For a random forest model where we are concerned with the minimal leaf node sizes, early stopping does not help much since according to the plot, the testing error stays pretty much constant. For a random forest model where we are concerned with the maximum depth, early stopping does not help much since the testing error stays pretty much constant.

Problem G.

Decisions trees are prone to overfitting from high variance particularly when the decision tree has a lot of depth. Random forests reduce variance by training on different subsets of data –

random forests are like bagged decision trees. As such, the decision tree model plots show a lot more fluctuation in test error than the plot generated for those using random forests although, in general, the shapes of the test error curve look rather similar.

III. The AdaBoost Algorithm

Problem A.

We let $h_t: \mathbb{R}^m \to \{-1, 1\}$ be the weak classifier obtained at step t and let α_t be its weight. Recall that the final classifier is

$$H(x) = sign\big(f(x)\big) = sign\,(\sum_{i=1}^{T} \alpha_t h_t(x))$$

We suppose $\{(x_1, y_1), \dots, (x_N, y_N)\} \subset \mathbb{R}^m \to \{-1, 1\}$ is our training dataset. We would like to show that the training set error of the final classifier can be bounded from above if an exponential loss function is used:

$$E = \frac{1}{N}\sum_{i=1}^{T} \exp(-y_i f(x_i)) \geq \frac{1}{N}\sum_{i=1}^{T} \mathbb{I}(H(x_i) \neq y_i),$$

where $\mathbb{I}$ is the indicator function.

In other words, we would like to show that for each data point in our training dataset, (x_i, y_i)

$$\exp\big(-y_i f(x_i)\big) \geq \mathbb{I}(H(x_i) \neq y_i)$$

We observe that there are only two cases: (1) y_i and $f(y_i)$ agree in sign and (2) y_i and $f(x_i)$ disagree in sign. Let us consider each case.

(1) y_i and $f(y_i)$ agree in sign

$$\mathbb{I}(H(x_i) \neq y_i) = 0$$
$$\exp\big(-y_i f(x_i)\big) = \exp(a)$$

where a is some negative number since y_i and $f(y_i)$ agree in sign which means $y_i f(x_i)$ must be a positive value and therefore, $-y_i f(x_i) = a = some\ negative\ value$. However, we know that $\exp(a) \geq 0$ which implies

$$\exp\big(-y_i f(x_i)\big) \geq \mathbb{I}(H(x_i) \neq y_i)$$

(2) y_i and $f(y_i)$ disagree in sign

$$\mathbb{I}(H(x_i) \neq y_i) = 1$$
$$\exp\big(-y_i f(x_i)\big) = \exp(b)$$

where b is some positive number since y_i and $f(y_i)$ disagree in sign which means $y_i f(x_i)$ must be a negative value and therefore, $-y_i f(x_i) = b = some\ positive\ value$

However, we know that $\exp(b) \geq 1$ for all positive values, b since $\exp(0) = 1$ which implies

$$\exp\big(-y_i f(x_i)\big) \geq \mathbb{I}(H(x_i) \neq y_i)$$

We have shown that in both cases,

$$\exp\big(-y_i f(x_i)\big) \geq \mathbb{I}(H(x_i) \neq y_i)$$

and as such, we can conclude that:

$$E = \frac{1}{N}\sum_{i=1}^{T} \exp(-y_i f(x_i)) \geq \frac{1}{N}\sum_{i=1}^{T} \mathbb{I}(H(x_i) \neq y_i),$$

where $\mathbb{I}$ is the indicator function.

Problem B.

We would like to find $D_{T+1}(i)$ in terms of Z_t, a_t, x_i, y_i and $h_t(x_i)$ where T is the last timestep and $t \in \{1, \ldots, T\}$. We recall that Z_t is the normalization factor for distribution D_{T+1}:

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

To update the weights, we observe that a recursive formula was provided from lecture:

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

We had defined the base case $D_1(i) = \frac{1}{N}$ for all $i = \{1, \ldots, N\}$ so that each data point is weighted equally. If we recursively update our weight, we can arrive at the following formula for D_{T+1}:

$$D_{T+1} = D_1(i) \cdot \frac{\exp(-\alpha_1 y_i h_1(x_i))}{Z_1} \frac{\exp(-\alpha_2 y_i h_2(x_i))}{Z_2} \ldots \frac{\exp(-\alpha_T y_i h_T(x_i))}{Z_T}$$

We observe that $Z_1 Z_2 \ldots Z_T = \prod_{t=1}^{T} Z_t$ and $\exp(-\alpha_1 y_i h_1(x_i)) \exp(-\alpha_2 y_i h_2(x_i)) \ldots \exp(-\alpha_T y_i h_T(x_i)) = \exp(-y_i \sum_{t=1}^{T} \alpha_t h_t(x_i))$ and we recall that $D_1(i) = \frac{1}{N}$ so we can rewrite our expression as:

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

Problem C.

We would like to show that $E = \sum_{i=1}^{N} \frac{1}{N} e^{\sum_{t=1}^{T} -\alpha_t y_i h_t(x_i)}$

We recall that

$$H(x) = sign(f(x)) = sign\left(\sum_{i=1}^{T} \alpha_t h_t(x)\right)$$

which means $f(x) = \sum_{i=1}^{T} \alpha_t h_t(x)$ and since

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

we can substitute to obtain our desired expression:

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

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

Problem D.

We would like to show that $E = \prod_{t=1}^{T} Z_t$. We recall that $\sum_{i=1}^{N} D_t(i) = 1$ because D is a distribution and we recall from lecture that:

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

Then we can rewrite the expression to obtain $D_t(i)$ as:

$$D_t(i) = D_1(i) \prod_{j=1}^{t-1} \frac{\exp(-\alpha_j y_i h_j(x_i))}{Z_j}$$

And we can further rewrite the expression to obtain Z_t as:

$$Z_t = \sum_{i=1}^{N} \left(\prod_{j=1}^{T-1} \frac{\exp\left(-\alpha_j y_i h_j(x_i)\right)}{Z_j} \cdot D_1(i) \right) \cdot \exp(-\alpha_T y_i h_T(x_i))$$

$$Z_t = \sum_{i=1}^{N} \left(\prod_{j=1}^{T-1} \frac{1}{Z_j} \right) \cdot \left(\prod_{j=1}^{T-1} \exp\left(-\alpha_j y_i h_j(x_i)\right) \right) \cdot D_1(i) \cdot \exp(-\alpha_T y_i h_T(x_i))$$

$$Z_T = \sum_{i=1}^{N} \left(\prod_{j=1}^{T-1} \frac{1}{Z_j} \right) \cdot \left(\prod_{j=1}^{T} \exp\left(-\alpha_j y_i h_j(x_i)\right) \right) \cdot D_1(i)$$

$$Z_T = \sum_{i=1}^{N} \left(\prod_{j=1}^{T-1} \frac{1}{Z_j} \right) \cdot \exp\left(-y_i \sum_{j=1}^{T} \alpha_j h_j(x_i) \right) \cdot D_1(i)$$

We recall that

$$H(x) = sign\big(f(x)\big) = sign\left(\sum_{i=1}^{T} \alpha_t h_t(x) \right)$$

to rewrite our expression as

$$Z_T = \sum_{i=1}^{N} \left(\prod_{j=1}^{T-1} \frac{1}{Z_j} \right) \cdot \exp\big(-y_i f(x_i)\big) \cdot D_1(i)$$

$$\left(\prod_{j=1}^{T-1} \frac{1}{Z_j} \right) Z_T = \sum_{i=1}^{N} \left(\exp\big(-y_i f(x_i)\big) \cdot D_1(i) \right)$$

$$\prod_{t=1}^{T} Z_t = \sum_{i=1}^{N} \left(\exp\big(-y_i f(x_i)\big) \cdot D_1(i) \right)$$

We recall that $D_1(i) = \frac{1}{N}$ so we have

$$\prod_{t=1}^{T} Z_t = \frac{1}{N} \sum_{i=1}^{N} \exp\big(-y_i f(x_i)\big)$$

And since

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

We have shown that:

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

Problem E.

We would like to show that the normalizer Z_t can be written as

$$Z_t = (1 - \varepsilon_t)\exp(-a_t) + \varepsilon_t \exp(a_t)$$

where ε_t is the training set error of weak classifier h_t for the weighted dataset:

$$\varepsilon_t = \sum_{i=1}^{N} D_t(i)\, \mathbb{I}\,(h_t(x_i) \neq y_i)$$

We observe that there are only two cases: (1) $h_t(x_i) \neq y_i$ and (2) $h_t(x_i) = y_i$. Let us consider each case.

(1) $h_t(x_i) \neq y_i$ implies that $h_t(x_i)y_i = -1$

Since we always normalize our weights such that they sum to 1:

$$\varepsilon_t = \sum_{i=1}^{N} D_t(i) = 1$$

Using our definition of the normalizer, we can substitute in our value and observe:

$$Z_t = (1 - \varepsilon_t)\exp(-a_t) + \varepsilon_t \exp(a_t)$$
$$Z_t = \exp(a_t)$$

We can check that this value agrees with our previous definition of Z_t

$$Z_T = \sum_{i=1}^{N} D_t(i) \cdot \exp\left(-\alpha_t y_i h_t(x_i)\right)$$
$$Z_T = \sum_{i=1}^{N} D_t(i) \cdot \exp(\alpha_t)$$
$$Z_T = \exp(\alpha_t)$$

The two values agree so we see that

$$Z_t = (1 - \varepsilon_t)\exp(-a_t) + \varepsilon_t \exp(a_t)$$

is valid when $h_t(x_i) \neq y_i$

(2) $h_t(x_i) = y_i$ implies that $h_t(x_i)y_i = 1$

Using our equation for the training set error of weak classifier h_t

$$\varepsilon_t = 0$$

Using our definition of the normalizer, we can substitute in our value and observe:

$$Z_t = (1 - \varepsilon_t)\exp(-a_t) + \varepsilon_t \exp(a_t)$$
$$Z_t = \exp(-a_t)$$

We can check that this value agrees with our previous definition of Z_t

$$Z_T = \sum_{i=1}^{N} D_t(i) \cdot \exp\left(-\alpha_t y_i h_t(x_i)\right)$$

$$Z_T = \sum_{i=1}^{N} D_t(i) \cdot \exp(-\alpha_t)$$

Again, since we always normalize our weights such that they sum to 1, we observe:

$$Z_T = \exp(-\alpha_t)$$

The two values agree so we see that

$$Z_t = (1 - \varepsilon_t)\exp(-a_t) + \varepsilon_t \exp(a_t)$$

is valid when $h_t(x_i) = y_i$

Problem F.

We would like to show that choosing α_t greedily to minimize Z_t at each iteration leads to the choices in AdaBoost:

$$a_t^* = \frac{1}{2}\ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right)$$

We had previously shown that the normalizer Z_t can be written as

$$Z_t = (1 - \varepsilon_t)\exp(-a_t) + \varepsilon_t \exp(a_t)$$

and now we will minimize Z_t with respect to a_t

$$\frac{\delta Z_t}{\delta a_t} = (-a_t)(1 - \varepsilon_t)\exp(-a_t) + a_t \varepsilon_t \exp(a_t)$$

$$\frac{\delta Z_t}{\delta a_t} = 0$$

$$(-a_t)(1 - \varepsilon_t)\exp(-a_t) + a_t \varepsilon_t \exp(a_t) = 0$$

$$(a_t)(1 - \varepsilon_t)\exp(-a_t) = a_t \varepsilon_t \exp(a_t)$$

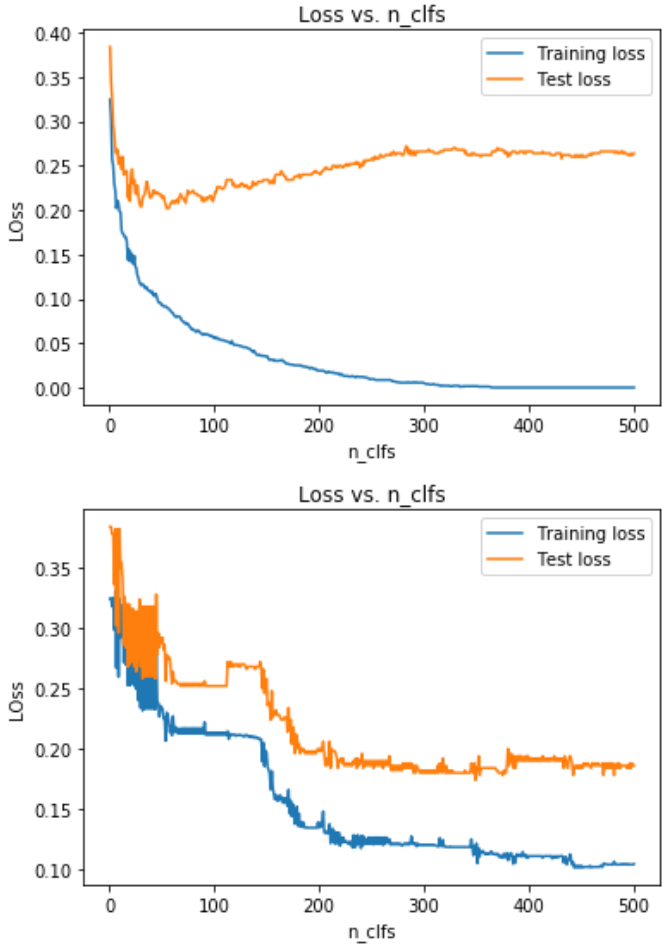
$$\exp(2a_t) = \frac{1 - \varepsilon_t}{\varepsilon_t}$$

$$a_t^* = \frac{1}{2}\ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right)$$

Therefore, we observe that choosing α_t greedily to minimize Z_t at each iteration leads to the choices in AdaBoost:

$$a_t^* = \frac{1}{2}\ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right)$$

Problem G.



Problem H.

The loss curve for gradient boosting is shown on the first plot and the loss curve for AdaBoost is shown on the second plot. The loss curve for gradient boosting is smoother since it fits on the residual. The final test error at the end of all the iterations was smaller for AdaBoost and larger for gradient boosting because gradient boosting overfits since according to the plot, the training loss was zero at the end.

Problem I.

The final test loss value using gradient boosting is 0.264 and the final test loss value using Adaboost is 0.186. Therefore, we see that Adaboost performed better on the classification dataset because its test error was lower.

Problem J.

For AdaBoost, the dataset weights are largest closer to the decision boundary and smaller for dataset weights further away from the decision boundary. Additionally, the dataset weights are larger as the number of iterations increase.