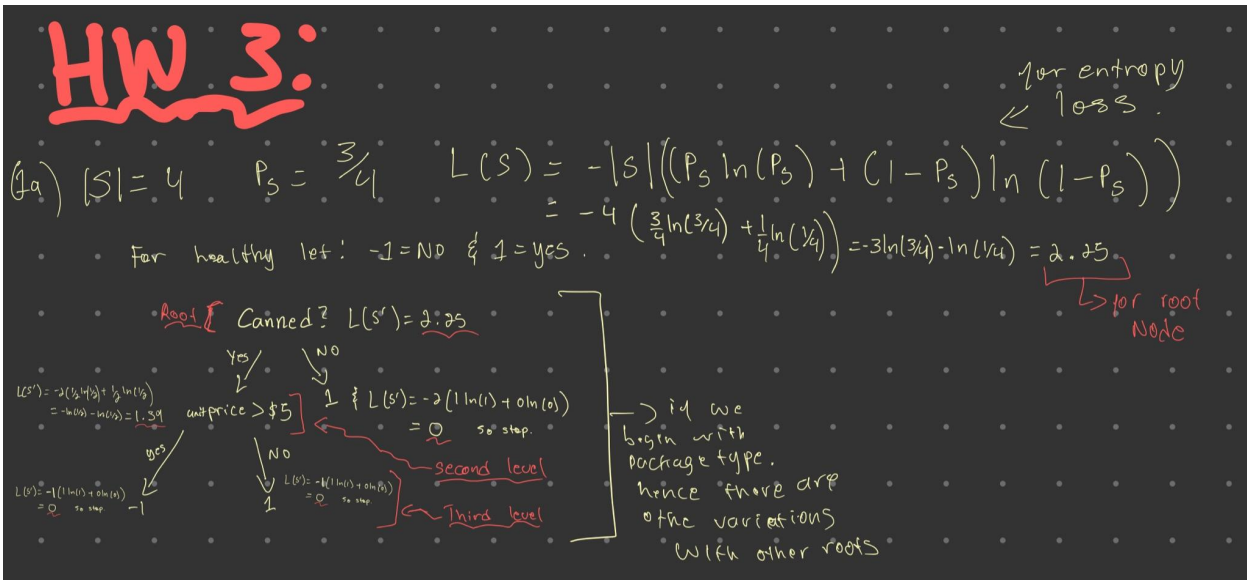
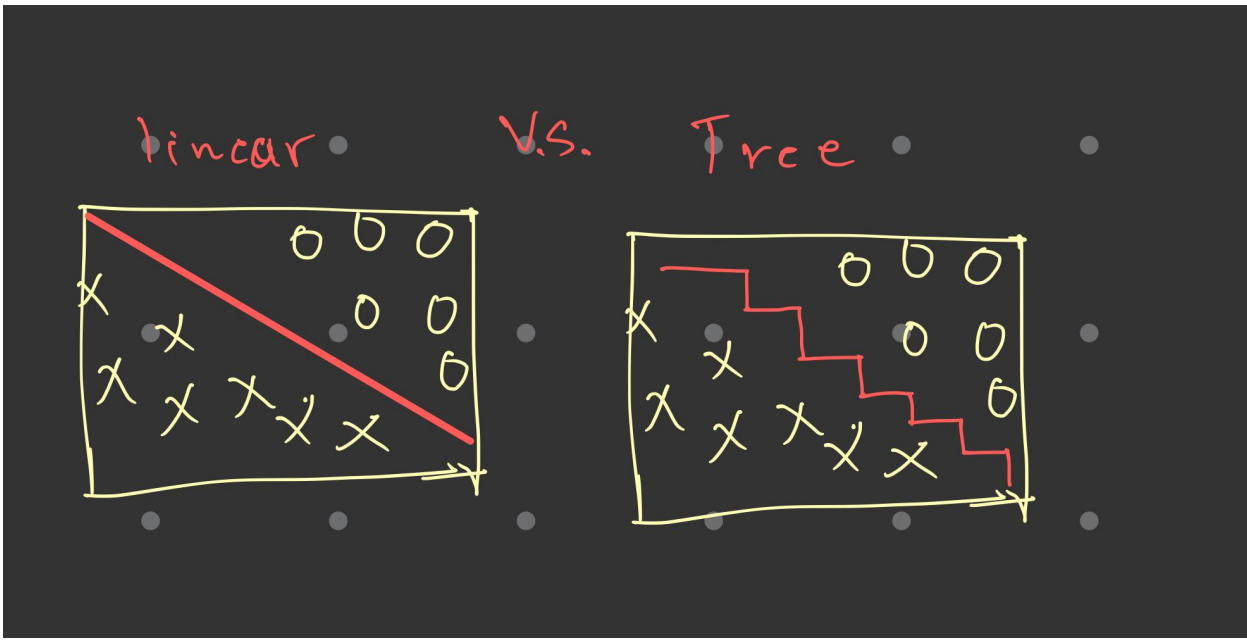
1. **Decision Trees:**

a.

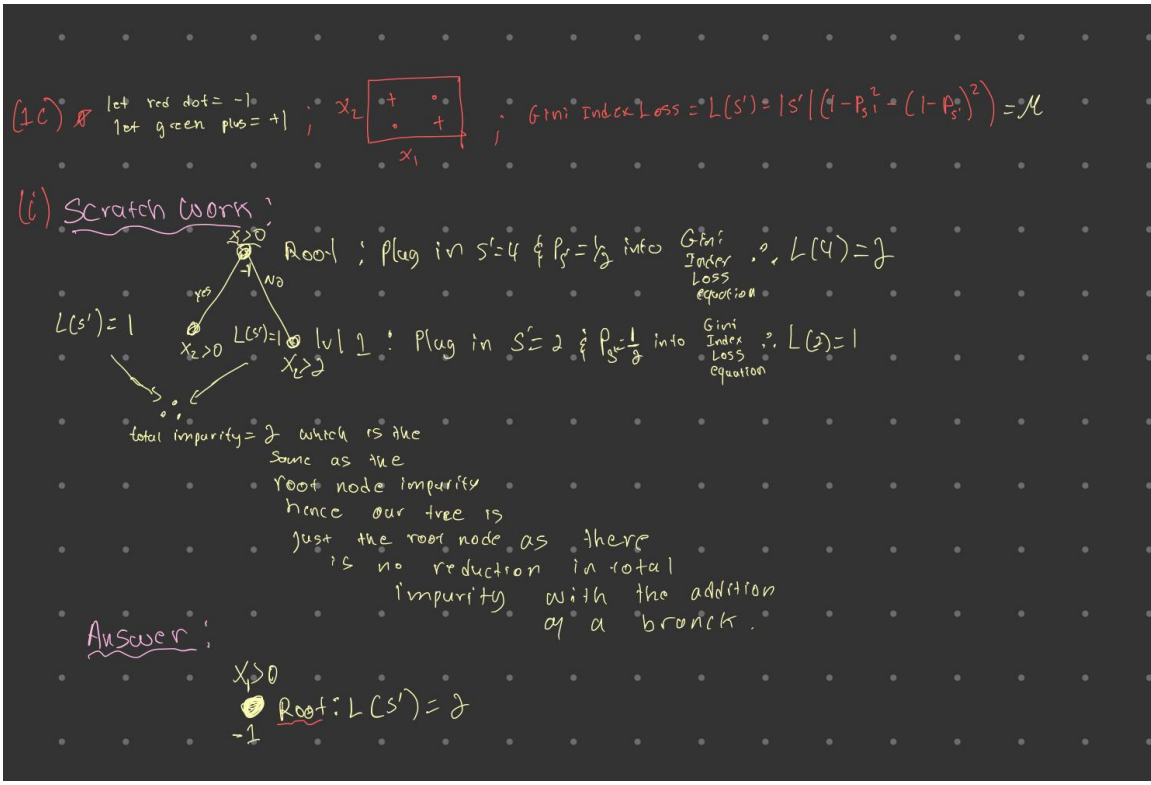


b. A decision is not always preferred over a linear classifier for classification because decision trees are always axis aligned hence they cannot easily classify diagonal boundaries as seen below where a simple linear SVM or regression can easily separate the data compared to an overly complex decision tree.



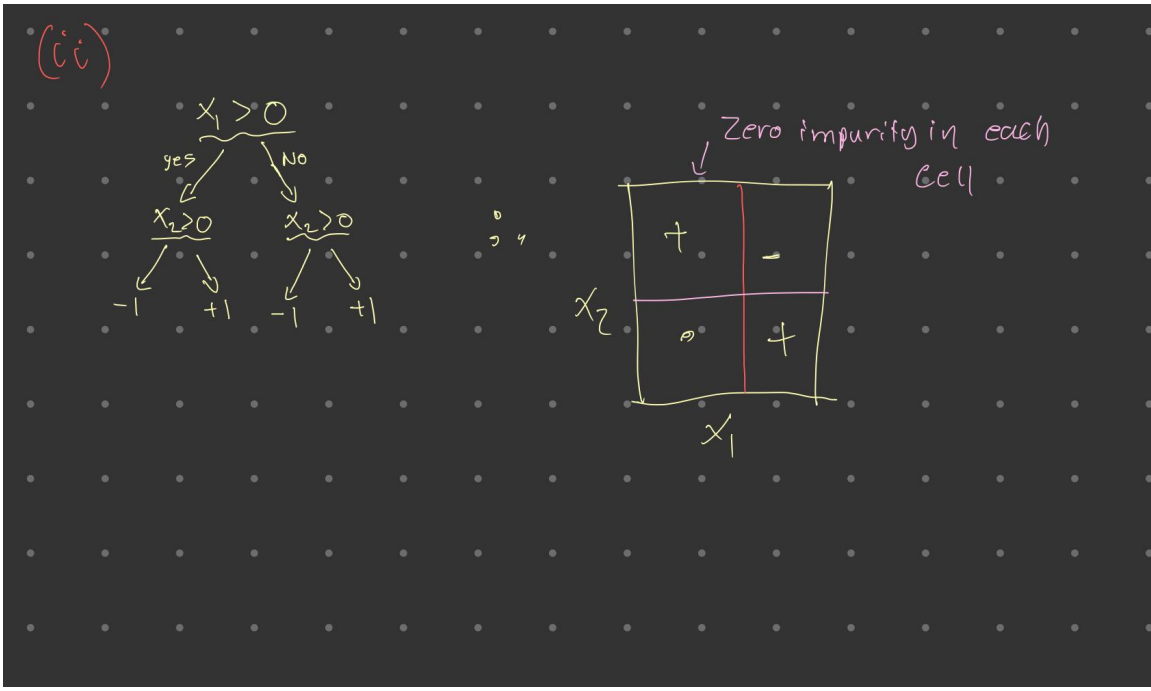
c.

i.



The resulting tree is just the root node due the scratch work above showing that there is no reduction in total impurity with the addition of a level hence our tree is simply the root node with either - 1 or 1 with a 5-% of each (I chose -1) . The classification error as defined in the problem is: (number of misclassified points) / (number of total points) = 2 / 4 = 1 / 2.

ii.



An impurity measure that would lead lead a top-down greedy induction with the same stopping condition to produce the tree we have drawn would be (|positive| * |negative|) /sqrt(|s'|) therefore the impurity at the root will be (2*2)/2 = 2; 2*(1*1)/sqrt(2) = 2/sqrt(2) and the total impurity on level 3 would be 2 * (1 * 0)/ 1 = 0 which is less than the root but we can also stop after that split as we have an impurity if 0 hence giving us the above diagram. A con is that we are more likely to succumb to overfitting because this measure will split data whereas other measures would not have hence making it bad for generalization. A pro is that our measure will split our data even if the percentage of correctly classified and correctly misclassified points before and after the split as in the previous problem where we couldnt keep splitting and here we could.

iii. The largest number of unique thresholds to achieve zero classification training error would be 99 splits because looking at smaller datasets we see that the maximum number of ways to split 1 point is zero, split 2 points requires one unique line to

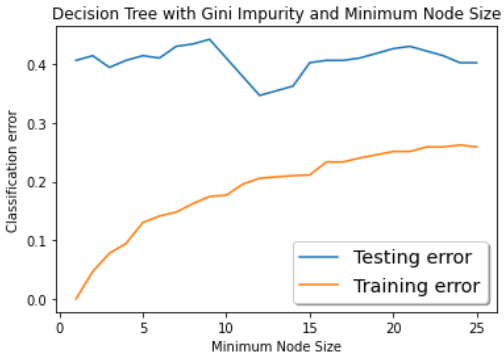
split them, 3 points have a maximum of 2 unique splits, 4 points have a maximum of 3 unique splits, 5 points have a maximum of 4 unique splits. Hence continuing this pattern we see that the maximum number of unique splits for N points will always be N - 1 hence 99. This is obvious because it takes a minimum of 99 splits until each point will be in its own box hence no further splits will be unique hence making that the maximum number of unique splits.

d. The worst-case time complexity of the number of possible splits we must consider in top-down greedy induction is O(N * D) because in the worst-case scenario, we would have to consider every possible combination of N data points with D continuous features, which would result in N * D possible splits. To determine the best split, we would need to evaluate each possible split, which takes O(1) time, and compare the impurity reduction for each split. Hence, the total complexity becomes O(N*D).

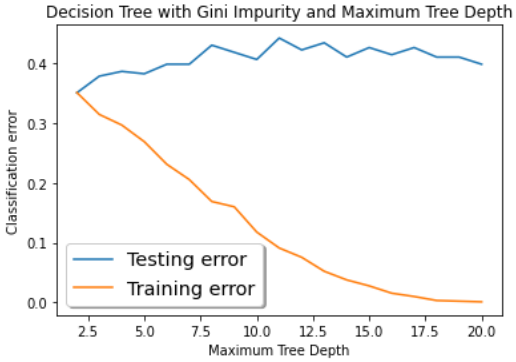
2. **Overfitting Decision Trees:**

See Code for 2

a.



(Extra Credit)



b. (i)

For the minimal leaf node size, the minimal node size of 12 best minimizes the test error. Early stopping allows us to avoid overfitting and underfitting because as seen when the minimum node size is less than 12 then the testing error rises as training error drops due to overfitting as our decision tree is fitting the noise. For minimum node sizes greater than 12, we see the testing error rises as training error increases due to underfitting as our model no longer fits the training set well enough.

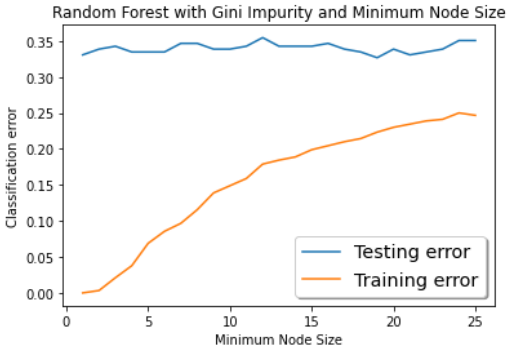
(Extra Credit)

(ii)

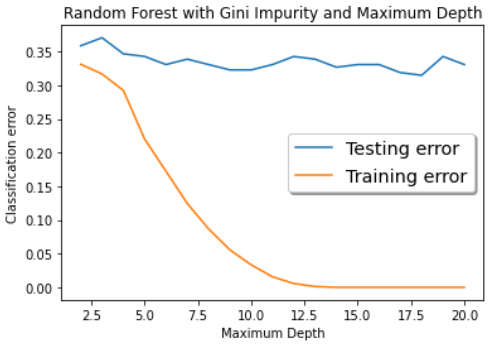
For The maximum tree depth, we see that a depth of 2 best minimizes the test error. Here early stopping helps prevent overfitting because we see that training error decreases as test error increases as the

maximum tree depth increases towards 25 hence meaning that with each additional layer, we start fitting noise hence overfitting.

c.



(Extra Credit)



d. (i)

For minimal leaf node size, the minimal node size of 19 had the lowest testing error. However, as we see the minimum node size does not have a significant impact on random forests with Gini Impurity as the testing error seems to remain constant and barely fluctuates with increased minimum node sizes thus early stopping does not have a significant impact on the performance of a random forest model. This is probably a result of the randomness at each node which makes random forests not susceptible to overfitting.

(Extra Credit)

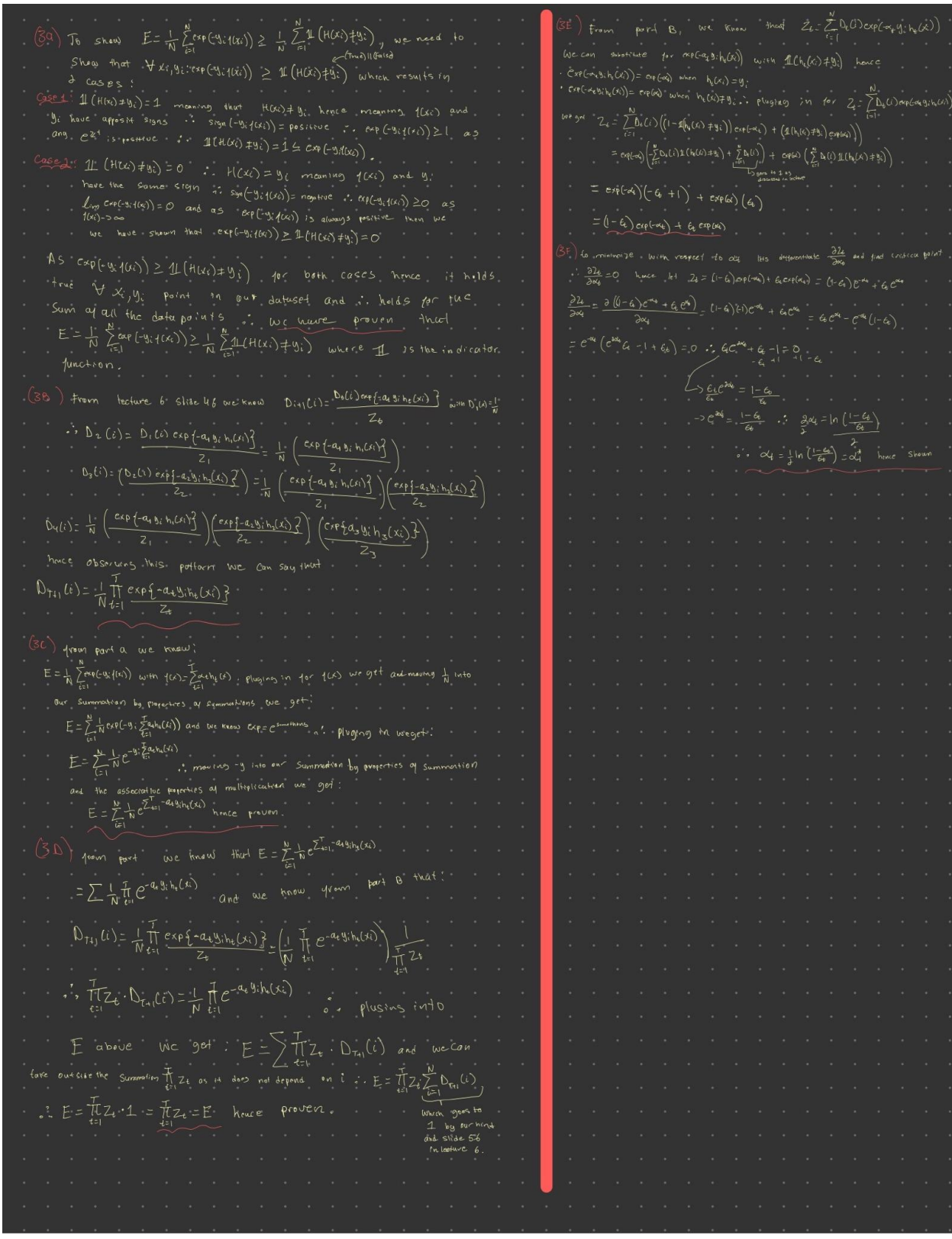
(ii) For maximum depth, the maximum depth size of 18 had the lowest testing error. However, as we see the maximum depth size does not have a significant impact on random forests with Gini Impurity as the testing error seems to remain constant and barely fluctuates with increased minimum node sizes. This is probably a result of the

randomness at each node which makes random forests not susceptible to overfitting.

e. In terms of testing error the curves of random forest fluctuate way less compared to the decision tree plot. However, the in sample error trends are similar as both increase with increased minimum node sizes and maximum depth. The random forest also reaches a lower testing error, however the decision tree dips to around the same testing error with a minimum node size of 12 hence allowing it to best benefit from early stopping as previously explained. Also, its high testing error could be a result of overfitting due to its fluctuating pattern.
f. See above
g. See above

3. **The AdaBoost Algorithm:**

   a. (A - F)



(3A) To show $E = \frac{1}{N}\sum_{i=1}^{N}\exp(-y_i f(x_i)) \geq \frac{1}{N}\sum_{i=1}^{N}\mathbb{1}(H(x_i)\neq y_i)$, we need to show that $\forall x_i, y_i : \exp(-y_i f(x_i)) \geq \mathbb{1}(H(x_i)\neq y_i)$ (true) (false) which results in 2 cases:

Case 1: $\mathbb{1}(H(x_i)\neq y_i) = 1$ meaning that $H(x_i)\neq y_i$ hence meaning $f(x_i)$ and y_i have opposite signs $\therefore$ sign$(-y_i f(x_i))$ = positive $\therefore \exp(-y_i f(x_i)) \geq 1$ as any e^{x^2} is positive $\therefore \mathbb{1}(H(x_i)\neq y_i) = 1 \leq \exp(-y_i f(x_i))$

Case 2: $\mathbb{1}(H(x_i)\neq y_i) = 0$ $\therefore H(x_i) = y_i$ meaning $f(x_i)$ and y_i have the same sign $\therefore$ sign$(-y_i f(x_i))$ = negative $\therefore \exp(-y_i f(x_i)) \geq 0$ as $\lim_{f(x_i)\to\infty}\exp(-y_i f(x_i)) = 0$ and as $\exp(-y_i f(x_i))$ is always positive then we have shown that $\exp(-y_i f(x_i)) \geq \mathbb{1}(H(x_i)\neq y_i) = 0$

As $\exp(-y_i f(x_i)) \geq \mathbb{1}(H(x_i)\neq y_i)$ for both cases hence it holds true $\forall x_i, y_i$ point in our dataset and $\therefore$ holds for the sum of all the data points $\therefore$ we have proven that $E = \frac{1}{N}\sum_{i=1}^{N}\exp(-y_i f(x_i)) \geq \frac{1}{N}\sum_{i=1}^{N}\mathbb{1}(H(x_i)\neq y_i)$ where $\mathbb{1}$ is the indicator function.

(3B) From lecture 6 slide 46 we know $D_{t+1}(i) = \frac{D_t(i)\exp\{-\alpha_t y_i h_t(x_i)\}}{Z_t}$ with $D_1(i) = \frac{1}{N}$

$\therefore D_2(i) = \frac{D_1(i)\exp\{-\alpha_1 y_i h_1(x_i)\}}{Z_1} = \frac{1}{N}\left(\frac{\exp\{-\alpha_1 y_i h_1(x_i)\}}{Z_1}\right)$

$D_3(i) = \left(\frac{D_2(i)\exp\{-\alpha_2 y_i h_2(x_i)\}}{Z_2}\right) = \frac{1}{N}\left(\frac{\exp\{-\alpha_1 y_i h_1(x_i)\}}{Z_1}\right)\left(\frac{\exp\{-\alpha_2 y_i h_2(x_i)\}}{Z_2}\right)$

$D_4(i) = \frac{1}{N}\left(\frac{\exp\{-\alpha_1 y_i h_1(x_i)\}}{Z_1}\right)\left(\frac{\exp\{-\alpha_2 y_i h_2(x_i)\}}{Z_2}\right)\left(\frac{\exp\{-\alpha_3 y_i h_3(x_i)\}}{Z_3}\right)$

hence observing this pattern we can say that

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

(3C) From part a we know:

$E = \frac{1}{N}\sum_{i=1}^{N}\exp(-y_i f(x_i))$ with $f(x) = \sum_{t=1}^{T}\alpha_t h_t(x)$. Plugging in for $f(x)$ we get and moving $\frac{1}{N}$ into our summation by properties of summations we get:

$E = \sum_{i=1}^{N}\frac{1}{N}\exp(-y_i\sum_{t=1}^{T}\alpha_t h_t(x_i))$ and we know $\exp = e^{\text{something}}$ $\therefore$ plugging in we get:

$E = \sum_{i=1}^{N}\frac{1}{N}e^{-y_i\sum_{t=1}^{T}\alpha_t h_t(x_i)}$ $\therefore$ moving $-y$ into our summation by properties of summation and the associative properties of multiplication we get:

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

(3D) From part c we know that $E = \sum_{i=1}^{N}\frac{1}{N}e^{\sum_{t=1}^{T}-\alpha_t y_i h_t(x_i)}$

$= \sum\frac{1}{N}\prod_{t=1}^{T}e^{-\alpha_t y_i h_t(x_i)}$ and we know from part B that:

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

$\therefore \prod_{t=1}^{T}Z_t \cdot D_{T+1}(i) = \frac{1}{N}\prod_{t=1}^{T}e^{-\alpha_t y_i h_t(x_i)}$ $\therefore$ plugging into E above we get: $E = \sum_{i}\prod_{t=1}^{T}Z_t \cdot D_{T+1}(i)$ and we can take outside the summation $\prod_{t=1}^{T}Z_t$ as it does not depend on i $\therefore E = \prod_{t=1}^{T}Z_t\sum_{i=1}^{N}D_{T+1}(i)$ which goes to 1 by our hint and slide 56 in lecture 6.

$\therefore E = \prod_{t=1}^{T}Z_t \cdot 1 = \prod_{t=1}^{T}Z_t = E$ hence proven.

(3E) From part B, we know that $Z_t = \sum_{i=1}^{N}D_t(i)\exp(-\alpha_t y_i h_t(x_i))$

We can substitute for $\exp(-\alpha_t y_i h_t(x_i))$ with $\mathbb{1}(h_t(x_i)\neq y_i)$ hence

- $\exp(-\alpha_t y_i h_t(x_i)) = \exp(-\alpha_t)$ when $h_t(x_i) = y_i$
- $\exp(-\alpha_t y_i h_t(x_i)) = \exp(\alpha_t)$ when $h_t(x_i)\neq y_i$ $\therefore$ plugging in for $Z_t = \sum_{i=1}^{N}D_t(i)\exp(-\alpha_t y_i h_t(x_i))$ we get $Z_t = \sum_{i=1}^{N}D_t(i)\left((1-\mathbb{1}(h_t(x_i)\neq y_i))\exp(-\alpha_t) + (\mathbb{1}(h_t(x_i)\neq y_i))\exp(\alpha_t)\right)$

$= \exp(-\alpha_t)\left(-\sum_{i=1}^{N}D_t(i)\mathbb{1}(h_t(x_i)\neq y_i) + \sum_{i=1}^{N}D_t(i)\right) + \exp(\alpha_t)\left(\sum_{i=1}^{N}D_t(i)\mathbb{1}(h_t(x_i)\neq y_i)\right)$ (the $\sum D_t(i)$ goes to 1 as it's a distribution we initialize)

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

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

(3F) To minimize with respect to α_t first differentiate $\frac{\partial Z_t}{\partial\alpha_t}$ and find critical point $\therefore \frac{\partial Z_t}{\partial\alpha_t} = 0$ hence let $Z_t = (1-\epsilon_t)\exp(-\alpha_t) + \epsilon_t\exp(\alpha_t) = (1-\epsilon_t)e^{-\alpha_t} + \epsilon_t e^{\alpha_t}$

$\frac{\partial Z_t}{\partial\alpha_t} = \frac{\partial\left((1-\epsilon_t)e^{-\alpha_t} + \epsilon_t e^{\alpha_t}\right)}{\partial\alpha_t} = (1-\epsilon_t)(-1)e^{-\alpha_t} + \epsilon_t e^{\alpha_t} = \epsilon_t e^{\alpha_t} - e^{-\alpha_t}(1-\epsilon_t)$

$= e^{-\alpha_t}\left(e^{2\alpha_t}\epsilon_t - 1 + \epsilon_t\right) = 0$ $\therefore \epsilon_t e^{2\alpha_t} + \epsilon_t - 1 = 0$ ($-\epsilon_t$, $+1$, $-1+\epsilon_t$)

$\rightarrow \epsilon_t e^{2\alpha_t} = 1-\epsilon_t$

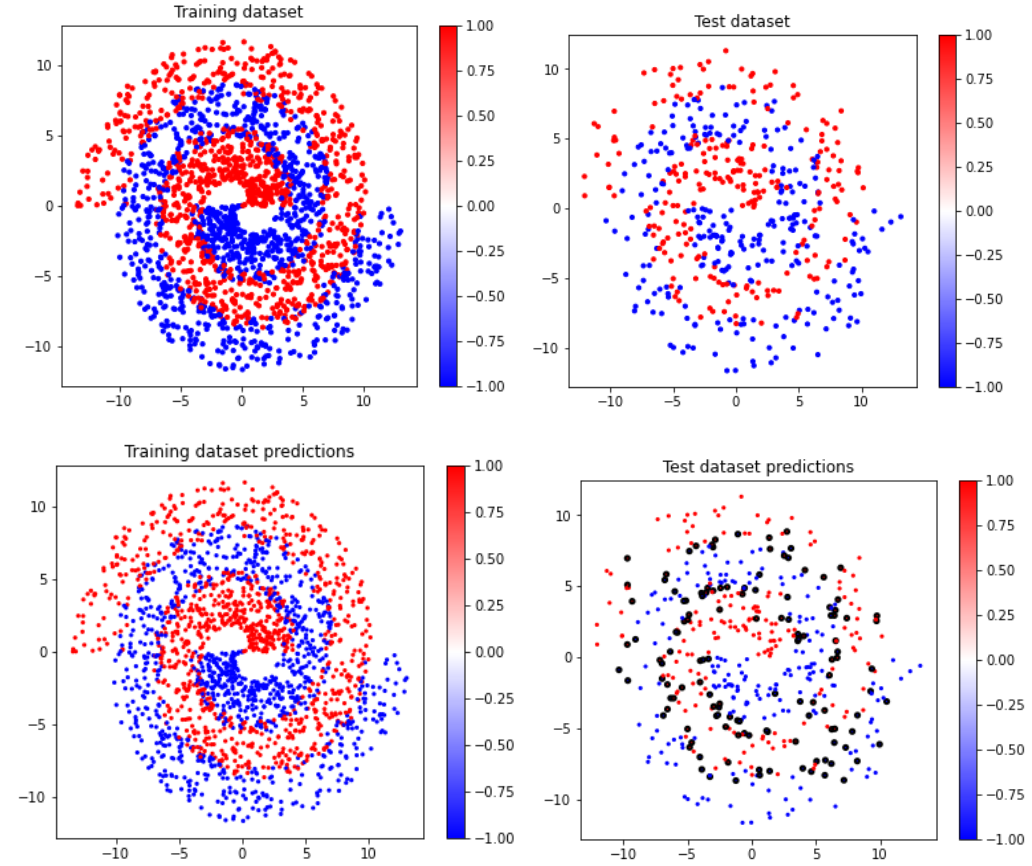
$\rightarrow e^{2\alpha_t} = \frac{1-\epsilon_t}{\epsilon_t}$ $\therefore \frac{2\alpha_t}{2} = \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$

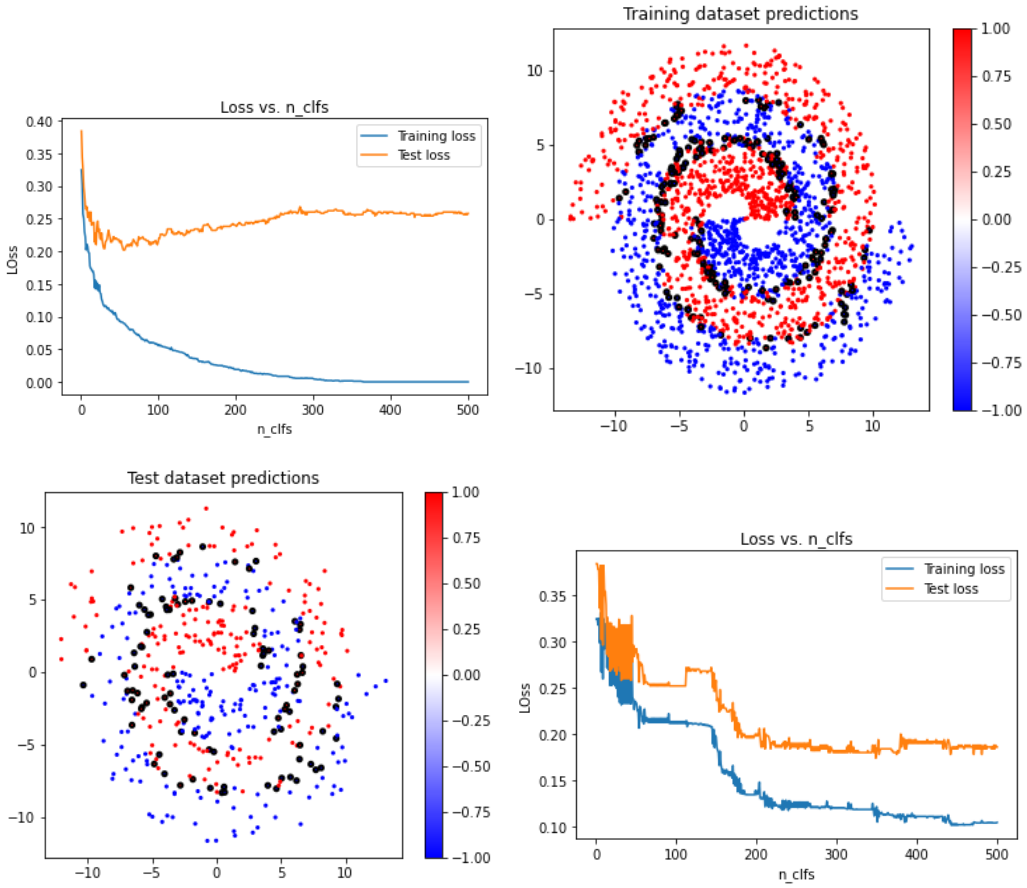
$\therefore \alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) = \alpha_t^{*}$ hence shown

b. All Graphs generated for question G:

[redacted]

See Code for prob3





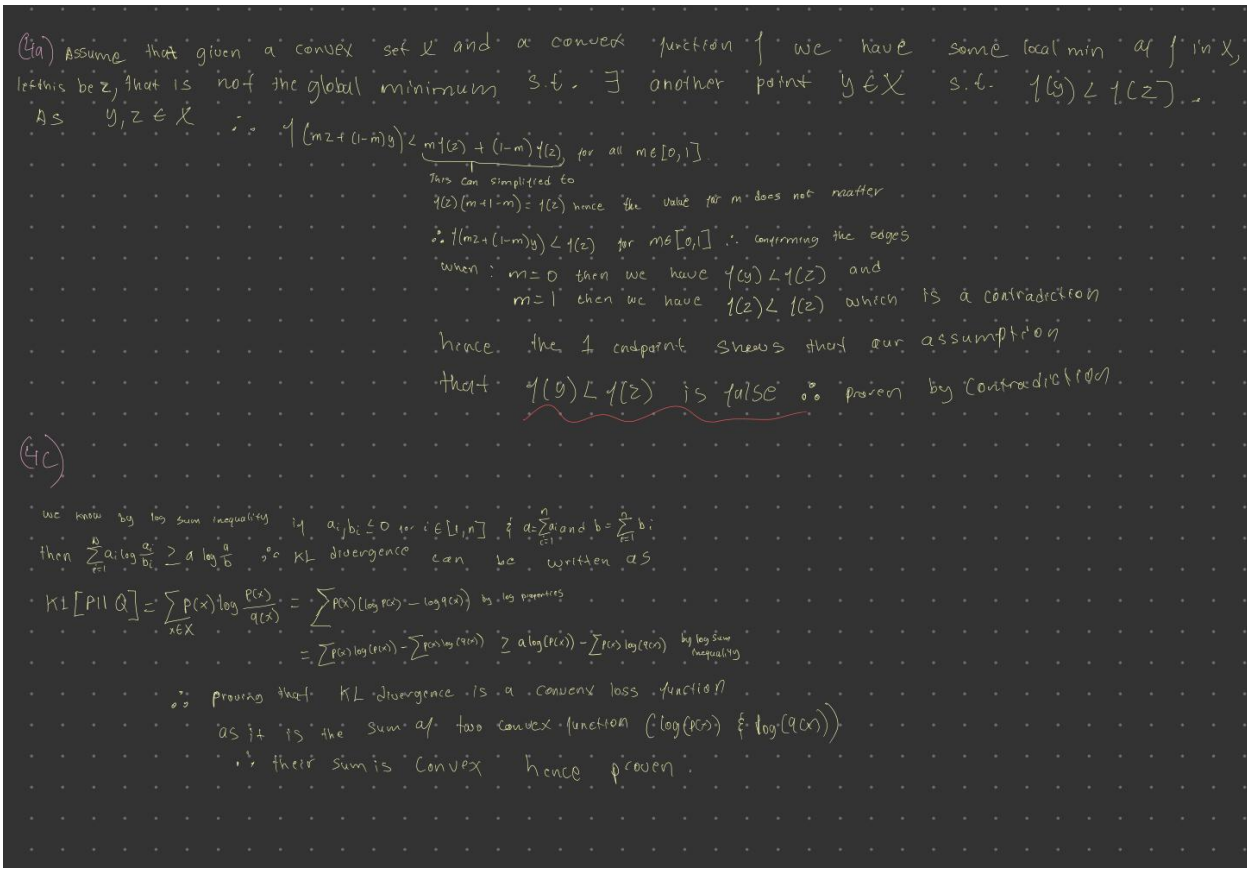
c. (H) Looking at the graphs of Loss vs. n_clfs for both, we see that both curves become smoother with increased n_clfs with gradient boosting testing error approaching a final value of around 0.25 and the AdaBoost testing approaches around 0.20. The gradient boost training loss approaches 0 while the AdaBoost training error approaches about 0.10. In terms of smoothness, the gradient loss is smoother overall however, the AdaBoost is smoother in certain regions such as n_clf range 50 - 150 where it looks completely flat. Also, the AdaBoost seems to be getting flatter while gradient Boost goes up and down towards the end.

d. (I) Gradient Boosting has a higher testing error and lower training error as seen in H hence as the AdaBoosting approaches a lower Test error compared to gradient boosting therefore making AdaBoosting a better generalizer than gradient boosting on the classification dataset as it generalizes better with a lower testing error.

e. (J) For AdaBoost, the dataset weights are the largest for misclassified points and smallest for correctly classified points.

4. **Convex Functions**

   a. (A & C)



   b. We want to find the parameters of our model that minimize the difference between the model predictions and the true labels i.e our loss function, which we are trying to minimize. If this loss function is convex, then any local minimum is also a global minimum, as seen in part A. This is desirable because training a learning model involves finding the optimal parameters, weight vector, by iteratively adjusting them based on the current loss. If the loss function is convex, then we can be confident that any local minimum we find will also be the global minimum, and that we have found the optimal parameters for the model. Furthermore, convex optimization problems can be efficiently solved using algorithms such as gradient descent, which makes it practical to find the global minimum of the loss function. Therefore, convex loss functions are desirable in training learning

models because they ensure that the optimal parameters can be found efficiently and with confidence.