Machine Learning & Data Mining
Set 3

Caltech CS/CNS/EE 155
January 26th, 2022

Dallas Taylor

Google Colab Links

- https://colab.research.google.com/drive/14NheKqbJZLCdvOFIdSTB3rz1FAK87X?usp=sharing
- https://colab.research.google.com/drive/1pUJUh0w0fty9IUccnhUt8HKIObKoIc0A?usp=sharing

Dallas Taylor

1 Decision Trees [30 Points]

Lecture 5

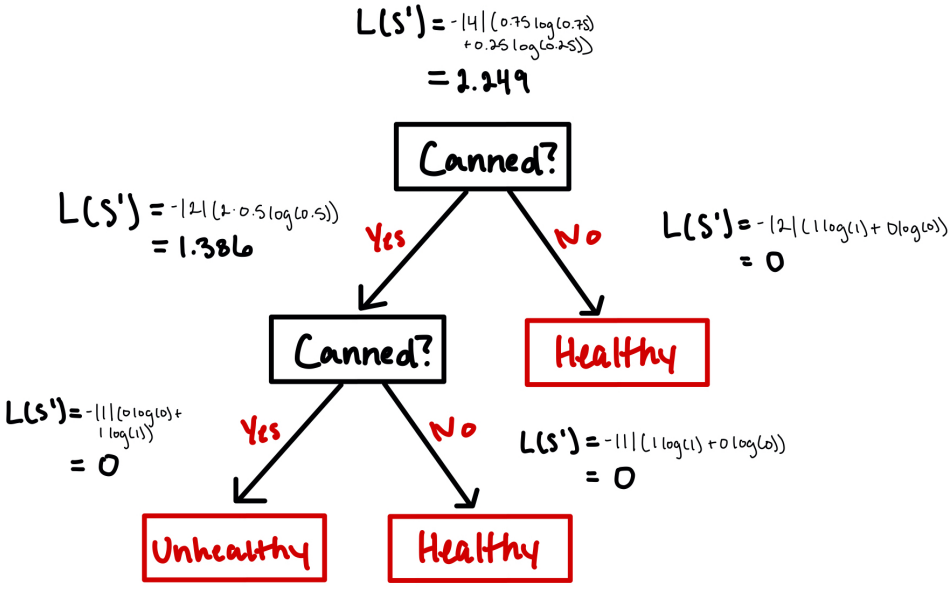
Question A [7 points]: Consider the following data, where given information about some food you must predict whether it is healthy:

No.	Package Type	Unit Price $>$ \$5	Contains $>$ 5 grams of fat	Healthy?
1	Canned	Yes	Yes	No
2	Bagged	Yes	No	Yes
3	Bagged	No	Yes	Yes
4	Canned	No	No	Yes

Train a decision tree by hand using top-down greedy induction. Use *entropy* (with natural log) as the impurity measure. Since the data can be classified without error, the stopping criterion will be no impurity in the leaves.

Submit a drawing of your tree showing the impurity reduction yielded by each split (including root) in your decision tree.

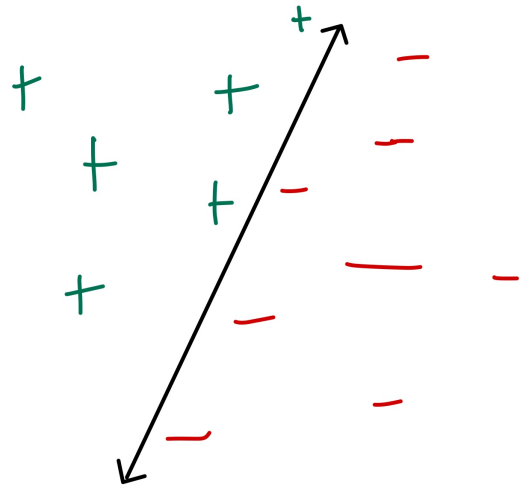
Solution A:

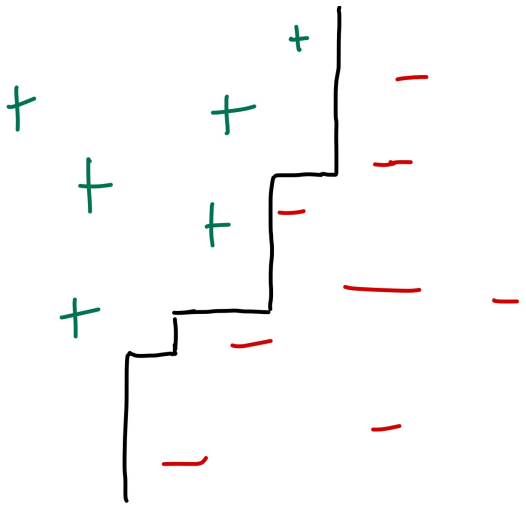


Note that here, healthy represents a value of 1 *and unhealthy represents a value of* 0.

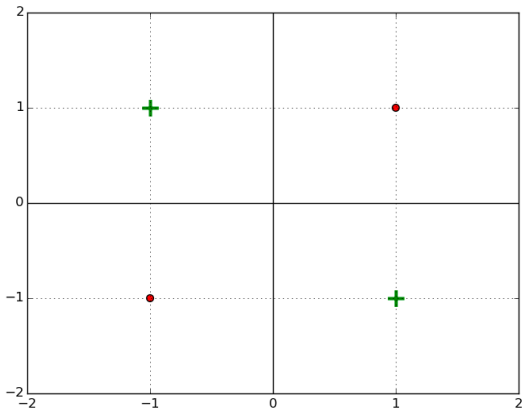
Question B [4 points]: Compared to a linear classifier, is a decision tree always preferred for classification problems? If not, draw a simple 2-D dataset that can be perfectly classified by a simple linear classifier but which requires an overly complex decision tree to perfectly classify.

Solution B: *A decision tree is not always preferred for classification problems. This can be seen in the following example, where we have data that is clearly easily linearly separable, but difficult to construct a decision tree. This is due to the fact that we must build our boundary decisions in our tree in parallel with the x and y axes.*



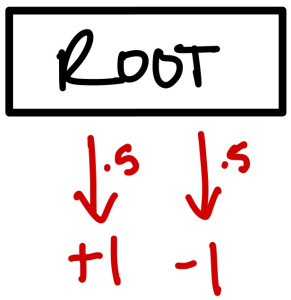


Question C [15 points]: Consider the following 2D data set:



i. [5 points]: Suppose we train a decision tree on this dataset using top-down greedy induction, with the Gini index as the impurity measure. We define our stopping condition to be if no split of a node results in any reduction in impurity. Submit a drawing of the resulting tree. What is its classification error ((number of misclassified points) / (number of total points))?

Solution C.i:

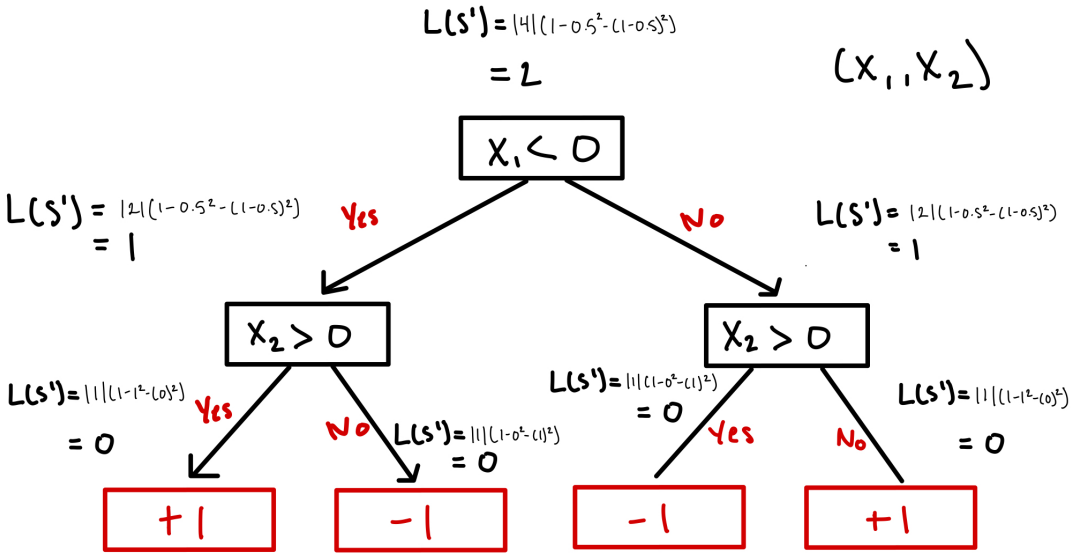


When considering our top-down greedy induction decision tree, there is no possible split to make in the data that results in a decrease in our impurity. Thus, we will produce the above tree, where we have just a root node who has a 50/50 probability of outputting either $+1, -1$*. Thus, it is clear that we have a classification error of* $\frac{2}{4} = 0.5$*.*

ii. [5 points]: Submit a drawing of a two-level decision tree that classifies the above dataset with zero classification error. (You don't need to use any particular training algorithm to produce the tree.)

Is there any impurity measure (i.e. any function that maps the data points under a particular node in a tree to a real number) that would have led top-down greedy induction with the same stopping condition to produce the tree you drew? If so, give an example of one, and briefly describe its pros and cons as an impurity measure for training decision trees in general (on arbitrary datasets).

Solution C.ii:



For the decision tree shown above, we have a classification error of $\frac{0}{4} = 0$. *A potential impurity measure that would have led our same top-down greedy induction to produce this same tree would be to take the natural exponent of our Gini index:*

$$L(S') \quad = \quad \exp\left(|S'|(1 - p_{S'}^2 - (1 - p_{S'})^2)\right)$$

Thus, the split described where $x_1 < 0$ *provides a decrease in our impurity measure (from* e^2 *to* $2e$*), and the two next splits where* $x_2 > 0$ *would also still provide our necessary impurity decrease as well (from* $2e$ *to* 2*). The pros of using this impurity measure on training decision trees is that we are able to allow data splits that would normally result in no impurity changes, allowing for potentially beneficial sets of data later in computation. However, using this altered impurity measure can also allow for over-fitting and over-generalization (we don't want to unnecessarily split data).*

---

iii. [5 points]: Suppose there are 100 data points in some 2-D dataset. What is the largest number of unique thresholds (i.e., internal nodes) you might need in order to achieve zero classification training error (on the training set)? Please justify your answer.

Solution C: *The largest number of unique thresholds that you might need in order to produce zero classification error on the training set is* 99*. This is due to the fact that if we imagine our data set to be similar to that given at the beginning of Question C, where we have alternating positive and negative points in a lattice (or even along a line), we would have to build a similar tree to the one I showed in C.ii, creating a threshold between each point, and thus* $n - 1 = 100 - 1 = 99$ *total thresholds.*

Question D [4 points]: Suppose in top-down greedy induction we want to split a leaf node that contains N data points composed of D continuous features. What is the worst-case complexity (big-O in terms of N and D) of the number of possible splits we must consider in order to find the one that most reduces impurity? Please justify your answer.

Note: Recall that at each node-splitting step in training a DT, you must consider all possible splits that you can make. 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 (which is what we mean by "split").

Solution D: *The worst case complexity of the number of possible splits we must consider is* $\mathcal{O}(D \cdot N)$*. This is due to the fact that we will be considering* $N - 1$ *splits in our worst case scenario, across our* D *continuous features.*

2 Overfitting Decision Trees [30 Points, EC 7 Points]

Lecture 5

In this problem, you will use the Diabetic Retinopathy Debrecen Data Set, which contains features extracted from images to determine whether or not the images contain signs of diabetic retinopathy. Additional information about this dataset can be found at the link below:

https://archive.ics.uci.edu/ml/datasets/Diabetic+Retinopathy+Debrecen+Data+Set

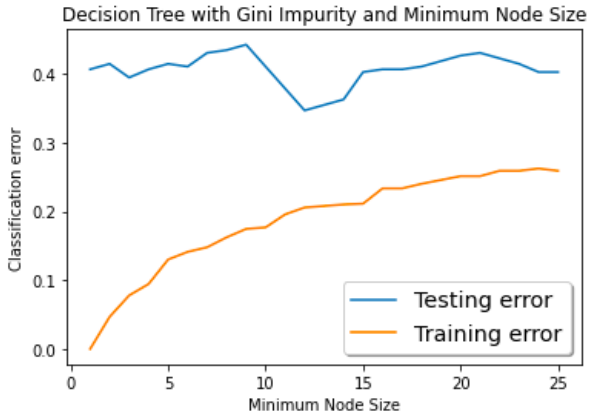
In the following question, your goal is to predict the diagnosis of diabetic retinopathy, which is the final column in the data matrix. Use the first 900 rows as training data, and the last 251 rows as validation data. Please feel free to use additional packages such as Scikit-Learn. Include your code in your submission.

Question A [10 points]: Choose one of the following from i or ii:

i. Train a decision tree classifier using Gini as the impurity measure and minimal leaf node size as early stopping criterion. Try different minimal leaf node sizes from 1 to 25 in increments of 1. Then, on a single plot, plot both training and test classification error versus leaf node size. To do this, fill in the `classification_err` and `eval_tree_based_model_min_samples` functions in the code template for this problem.

ii. Train a decision tree classifier using Gini as the impurity measure and maximal tree depth as early stopping criterion. Try different tree depths from 2 to 20 in increments of 1. Then, on a single plot, plot both training and test classification error versus tree depth. To do this, fill in the `eval_tree_based_model_max_depth` function in the code template for this problem.

Solution A:



Fix this Colab Link : https://colab.research.google.com/drive/14NheKqbJZLCdvOFIdSTB3rz1FAK87X?usp=sharing

Question B [6 points]: For either the minimal leaf node size or maximum depth parameters in the previous problem, which parameter value minimizes the test error? What effects does early stopping have on the performance of a decision tree model? Please justify your answer based on the plot you derived.

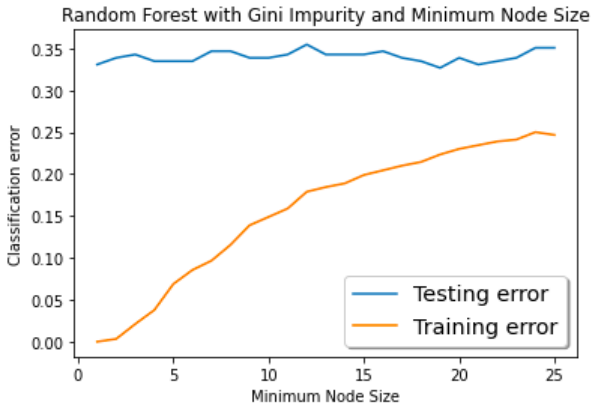
Solution B: *The parameter that minimizes the test error for minimal leaf node size is when we have a leaf node size of* 12*. It is also clear here that early stopping could be beneficial for this model, as once we pass our minimal leaf node size of* 12*, then both our training and testing error increases. Thus, stopping at this node size would prove beneficial.*

Question C [4 points]: Choose one of the following from i or ii:

i. Train a random forest classifier using Gini as the impurity measure, minimal leaf node size as early stopping criterion, and 1,000 trees in the forest. Try different node sizes from 1 to 25 in increments of 1. Then, on a single plot, plot both training and test classification error versus leaf node size.

ii. Train a random forest classifier using Gini as the impurity measure, maximal tree depth as early stopping criterion, and 1,000 trees in the forest. Try different tree depths from 2 to 20 in increments of 1. Then, on a single plot, plot both training and test classification error versus tree depth.

Solution C:



Question D [6 points]: For either the minimal leaf node size or maximum depth parameters tested, which parameter value minimizes the random forest test error? What effects does early stopping have on the performance of a random forest model? Please justify your answer based on the plot you derived.

Solution D: *The random forest test error is minimized at the minimal leaf node size of* 19*. It can be opposingly seen here, however, that early stopping would likely not be that beneficial for our dataset, since we have our training error smoothly increasing and our testing error stabilizing horizontally across all node sizes.*

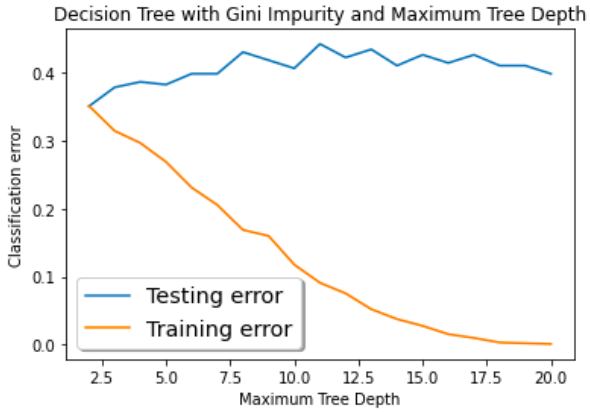
Question E [4 points]: Do you observe any differences between the curves for the random forest and decision tree plots? If so, explain what could account for these differences.

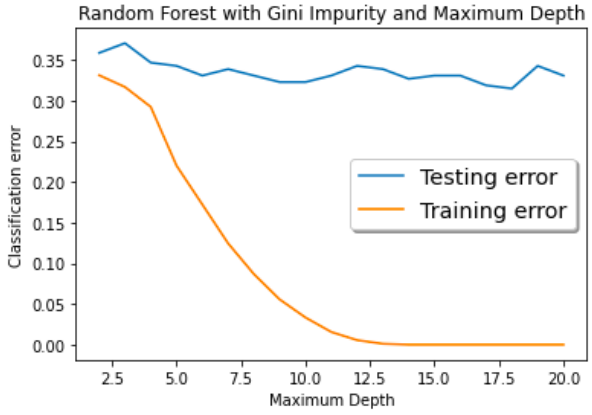
Solution E: *The curves are mostly similar in general trajectory between the decision tree and random forest plots. However, the random forest plot is much cleaner, with fewer deviations from the general trajectory. This could be accounted for by the number of estimators that factor into our random forest classifier. We can additionally see that our testing error is around* 0.05 *higher for our decision tree vs our random forest.*

Extra Credit [7 points total] :

Question F: [5 points, Extra Credit] Complete the other option for **Problem A** and **Problem C**.

Solution F:





Question G: [2 points, Extra Credit] For the stopping criterion tested in **Problem F**, which parameter value minimizes the decision tree and random forest test error respectively?

Solution G: *The parameter that minimizes the decision tree and random forest is maximum depth* $= 2, 18$*, respectively.*

3 The AdaBoost Algorithm [40 points]

Lecture 6

In this problem, you will show that the choice of the α_t parameter in the AdaBoost algorithm corresponds to greedily minimizing an exponential upper bound on the loss term at each iteration.

Question A [3 points]: 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

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

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

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

where $\mathbb{1}$ is the indicator function.

Solution A: *Let us further break down our training set error using $H(x)$. We know that the notation of $\mathbb{1}$ indicates that our sum will either by adding 1 if $H(x_i) \neq y_i$, and 0 otherwise. Furthermore, we know that when $H(x_i) \neq y_i$, then we have that $\mathbb{1}(H(x_i) \neq y_i) = 1$. Thus, we must also have that sign$(f(x_i)) \neq y_i$ and therefore $-y_i f(x_i) > 0$ and $e^{-y_i f(x_i)} > 1$. Thus, for this case, we know that our exponential loss is always greater than our $H(x)$ error.*

Additionally, we know that when $H(x_i) = y_i$, then we have that $\mathbb{1}(H(x_i) \neq y_i) = 0$. Thus, we must also have that sign$(f(x_i)) = y_i$ and therefore $-y_i f(x_i) < 0$ and $e^{-y_i f(x_i)} > 0$, by known exponential properties. Thus, for our other case, we know that our exponential loss is always greater than our $H(x)$ error.

Thus, it is clear that for all possible data points, we will be adding a larger value to our exponential loss sum versus our $H(x)$ sum, and we have shown that our final classifier can be bounded from above by our exponential loss function.

Question B [3 points]: Find $D_{T+1}(i)$ in terms of Z_t, α_t, x_i, y_i, and the classifier h_t, where T is the last timestep and $t \in \{1, \ldots, T\}$. Recall that Z_t is the normalization factor for distribution D_{t+1}:

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

Solution B: *We can write $D_{T+1}(i)$ to be the following:*

$$\begin{aligned}
D_{T+1}(i) &= \frac{D_T(i) \exp\left(-\alpha_T y_i h_T(x_i)\right)}{Z_T} \\
&= \frac{D_{T-1}(i) \exp\left(-\alpha_{T-1} y_i h_{T-1}(x_i)\right) \exp\left(-\alpha_T y_i h_T(x_i)\right)}{Z_T Z_{T-1}} \\
&= \frac{D_1(i) \exp\left(-\alpha_T y_i h_T(x_i)\right) \exp\left(-\alpha_{T-1} y_i h_{T-1}(x_i)\right) \cdots \exp\left(-\alpha_1 y_i h_1(x_i)\right)}{Z_T Z_{T-1} \cdots Z_1} \\
D_{T+1}(i) &= \frac{1}{N} \prod_{t=1}^{T} \frac{\exp\left(-\alpha_t y_i h_t(x_i)\right)}{Z_t}
\end{aligned}$$

Question C [2 points]: Show that $E = \sum_{i=1}^{N} \frac{1}{N} e^{\sum_{t=1}^{T} -\alpha_t y_i h_t(x_i)}$.

Solution C: *We are given that our error function E is defined as the following:*

$$\begin{aligned}
E &= \frac{1}{N} \sum_{i=1}^{N} \exp\left(-y_i f(x_i)\right) \\
&= \frac{1}{N} \sum_{i=1}^{N} \exp\left(-y_i \sum_{t=1}^{T} \alpha_t h_t(x_i)\right) \\
E &= \sum_{i=1}^{N} \frac{1}{N} e^{\sum_{t=1}^{T} -\alpha_t y_i h_t(x_i)},
\end{aligned}$$

as desired.

Question D [5 points]: Show that

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

Hint: *Recall that* $\sum_{i=1}^{N} D_t(i) = 1$ *because* D *is a distribution.*

Solution D: *We found our definition for* D_{T+1}*. Thus, we can perform the following operations:*

$$\begin{aligned}
D_{T+1} &= \frac{1}{N} \prod_{t=1}^{T} \frac{\exp\left(-\alpha_t y_i h_t(x_i)\right)}{Z_t} \\
\therefore D_{T+1} \prod_{t=1}^{T} Z_t &= \frac{1}{N} \prod_{t=1}^{T} \exp\left(-\alpha_t y_i h_t(x_i)\right) \\
\therefore \sum_{i=1}^{N} \left(D_{T+1} \prod_{t=1}^{T} Z_t \right) &= \sum_{i=1}^{N} \left(\frac{1}{N} \exp \sum_{t=1}^{T} -\alpha_t y_i h_t(x_i) \right) \\
\therefore 1 \cdot \prod_{t=1}^{T} Z_t &= \sum_{i=1}^{N} \frac{1}{N} \exp \sum_{t=1}^{T} -\alpha_t y_i h_t(x_i) \\
\therefore \textit{By Question C:} \quad E &= \prod_{t=1}^{T} Z_t
\end{aligned}$$

Question E [5 points]: Show that the normalizer Z_t can be written as

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

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

$$\epsilon_t = \sum_{i=1}^{N} D_t(i) \mathbb{1}(h_t(x_i) \neq y_i).$$

Solution E: *Let us begin with our given definition of Z_t and perform operations to simplify to our desired result. We additionally know that we are able to break down our function using the relationship between $h_t(x_i)$ and y_i.*

$$\begin{aligned}
Z_t &= \sum_{i=1}^{N} D_t(i) \exp\left(-\alpha_t y_i h_t(x_i)\right) \\
&= \sum_{i=1}^{N} D_t(i) \exp\left(-\alpha_t\right) \mathbb{1}(h_t(x_i) = y_i) + \sum_{i=1}^{N} D_t(i) \exp\left(\alpha_t\right) \mathbb{1}(h_t(x_i) \neq y_i) \\
&= \exp\left(-\alpha_t\right) \sum_{i=1}^{N} D_t(i) \mathbb{1}(h_t(x_i) = y_i) + \exp\left(\alpha_t\right) \sum_{i=1}^{N} D_t(i) \mathbb{1}(h_t(x_i) \neq y_i) \\
&= (1 - \epsilon_t) \exp\left(-\alpha_t\right) + \epsilon_t \exp\left(\alpha_t\right),
\end{aligned}$$

as desired.

Question F [2 points]: We derived all of this because it is hard to directly minimize the training set error, but we can greedily minimize the upper bound E on this error. Show that choosing α_t greedily to minimize Z_t at each iteration leads to the choices in AdaBoost:

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

Solution F: *We can see that by rearranging our answer from Question E, we can get the following:*

$$\begin{aligned}
Z_t &= (1-\epsilon_t)\exp(-\alpha_t) + \epsilon_t \exp(\alpha_t) \\
\therefore \frac{d}{d\alpha_t} &= -(1-\epsilon_t)\exp(-\alpha_t) + \epsilon_t \exp(\alpha_t) \\
\therefore 0 &= -(1-\epsilon_t)\exp(-\alpha_t) + \epsilon_t \exp(\alpha_t) \\
0 &= (\epsilon_t - 1 + \epsilon_t \exp(\alpha_t)^2)\exp(-\alpha_t) \\
0 &= \epsilon_t - 1 + \epsilon_t \exp(\alpha_t)^2 \\
\therefore -\epsilon_t \exp(\alpha_t)^2 &= \epsilon_t - 1 \\
\therefore \exp(\alpha_t)^2 &= \frac{\epsilon_t - 1}{-\epsilon_t} \\
\therefore \exp(\alpha_t)^2 &= \left(\frac{1-\epsilon_t}{\epsilon_t}\right) \\
\therefore \ln(\exp(\alpha_t)^2) &= \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) \\
\therefore 2\alpha_t* &= \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) \\
\therefore \alpha_t* &= \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right),
\end{aligned}$$

as desired.

Question G [14 points]: Implement the `GradientBoosting.fit()` and `AdaBoost.fit()` methods in the notebook provided for you. Some important notes and guidelines follow:

- For both methods, make sure to work with the class attributes provided to you. Namely, after `GradientBoosting.fit()` is called, `self.clfs` should be appropriately filled with the `self.n_clfs` trained weak hypotheses. Similarly, after `AdaBoost.fit()` is called, `self.clfs` and `self.coefs` should be appropriately filled with the `self.n_clfs` trained weak hypotheses and their coefficients, respectively.
- `AdaBoost.fit()` should additionally return an (N, T) shaped numpy array `D` such that `D[:, t]` contains D_{t+1} for each $t \in \{0, \ldots,$ `self.n_clfs`$\}$.
- For the `AdaBoost.fit()` method, **use the 0/1 loss** instead of the exponential loss.

- The only Sklearn classes that you may use in implementing your boosting fit functions are the DecisionTreeRegressor and DecisionTreeClassifier, not GradientBoostingRegressor.

Question H [2 points]: Describe and explain the behaviour of the loss curves for gradient boosting and for AdaBoost. You should consider two kinds of behaviours: the smoothness of the curves and the final values that the curves approach.

Solution H:

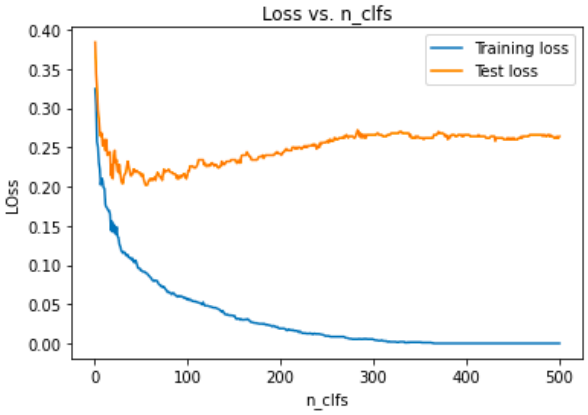


Figure 1: Gradient Boosting Loss Curves

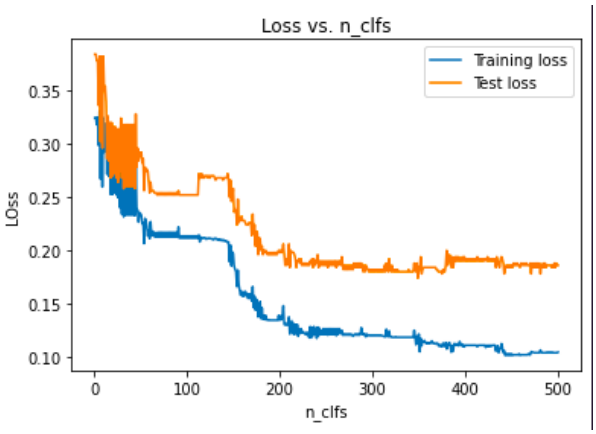


Figure 2: AdaBoost Loss Curves

We can see from the above figures that the loss curves between Gradient boosting and AdaBoost evolve slightly differently. Gradient boosting results in a steep decrease to around 0.25 *for the first* 25 *classifiers, and*

then stabilizes and splits such that our test loss is around 0.25 *higher than our training loss for the remaining classifiers. We can also see that our curves are very smooth, with testing and training stabilizing around* 0.25 *and* 0.0*, respectively.*

AdaBoost results in a similar decrease to around 0.25 within the first 100 *classifiers, where the test loss is somewhat higher than the training loss for all n_clifs* > 50*. We then (different from the gradient boosting loss) see another decrease with test and training loss stabilizing at around* 0.2 *and* 0.1*, respectively. We can also see that our loss curves are much less smooth than our gradient boosting curves.*

Colab Link : https://colab.research.google.com/drive/1pUJUh0w0fty9IUccnhUt8HKIObKoIc0A?usp=sharing

Question I [2 points]: Compare the final loss values of the two models. Which performed better on the classification dataset?

Solution I:

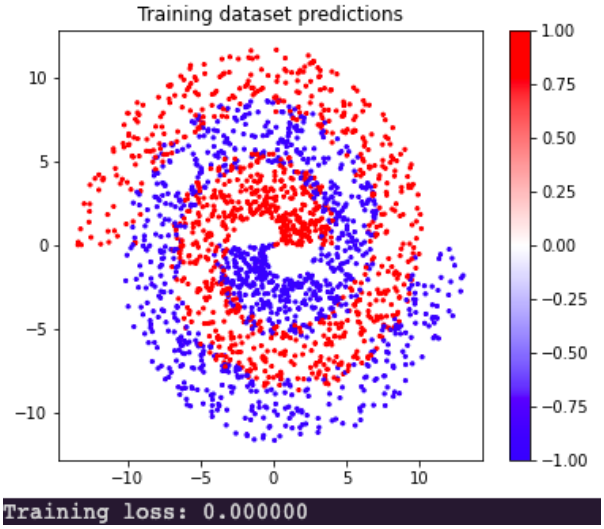


Figure 3: Gradient Boosting Final Training Loss

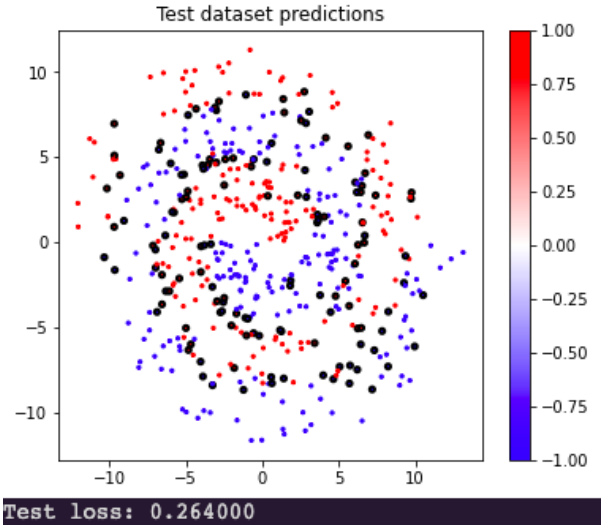


Figure 4: Gradient Boosting Final Testing Loss

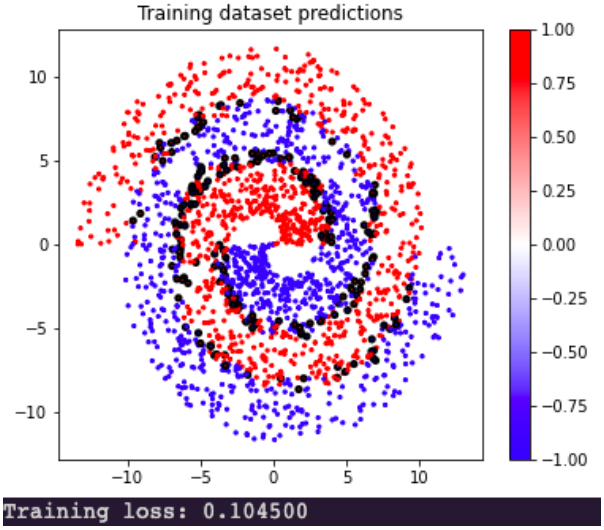


Figure 5: AdaBoost Final Training Loss

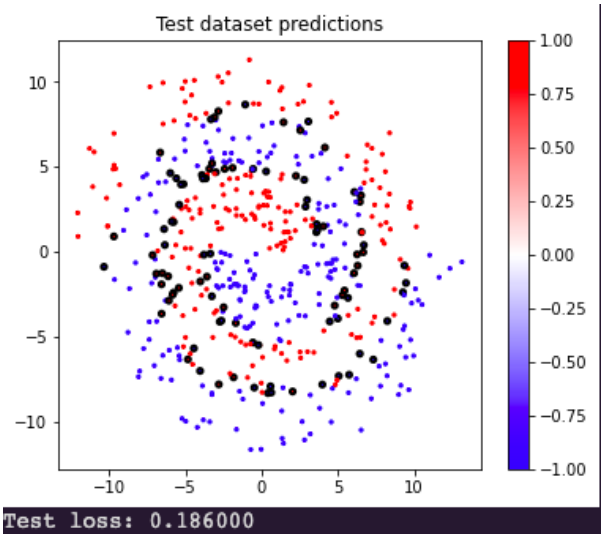


Figure 6: AdaBoost Final Testing Loss

It is clear from the above plots that gradient boosting performed better on the training data set, however AdaBoost performed better on the testing data set.

Question J [2 points]: For AdaBoost, where are the dataset weights the largest, and where are they the smallest? ***Hint:*** *Watch how the dataset weights change across time in the animation.*

Solution J: *It is clear that we have that our dataset weights are largest around misclassified points and they are the smallest around correctly classified points.*