Machine Learning & Data Mining
Set 3

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

Policies

- Due 9 PM PST, January 26th on Gradescope.
- You are free to collaborate on all of the problems, subject to the collaboration policy stated in the syllabus.
- In this course, we will be using Google Colab for code submissions. You will need a Google account.

Submission Instructions

- Submit your report as a single .pdf file to Gradescope, under "Set 3 Report".
- In the report, **include any images generated by your code** along with your answers to the questions.
- Submit your code by **sharing a link in your report** to your Google Colab notebook for each problem (see naming instructions below). Make sure to set sharing permissions to at least "Anyone with the link can view". **Links that can not be run by TAs will not be counted as turned in.** Check your links in an incognito window before submitting to be sure.
- For instructions specifically pertaining to the Gradescope submission process, see https://www.gradescope.com/get_started#student-submission.

Google Colab Instructions

For each notebook, you need to save a copy to your drive.

1. Open the github preview of the notebook, and click the icon to open the colab preview.
2. On the colab preview, go to File → Save a copy in Drive.
3. Edit your file name to "lastname_firstname_set_problem", e.g."yue_yisong_set3_prob2.ipynb"

1 Decision Trees [30 Points]

Relevant materials: Lecture 5

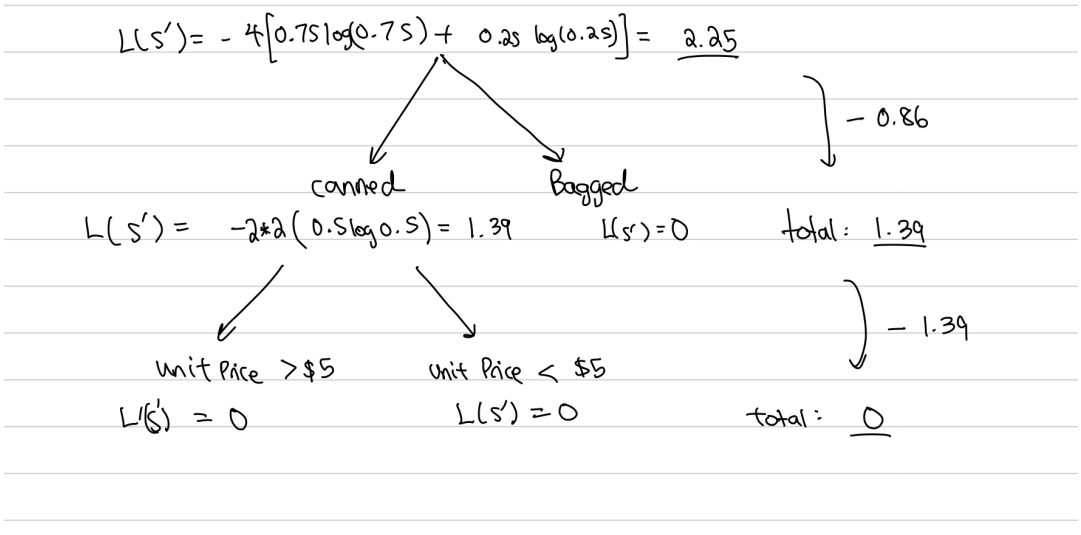
Problem A [7 points]: Consider the following data, where given information about some food you must predict whether it is healthy:

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.

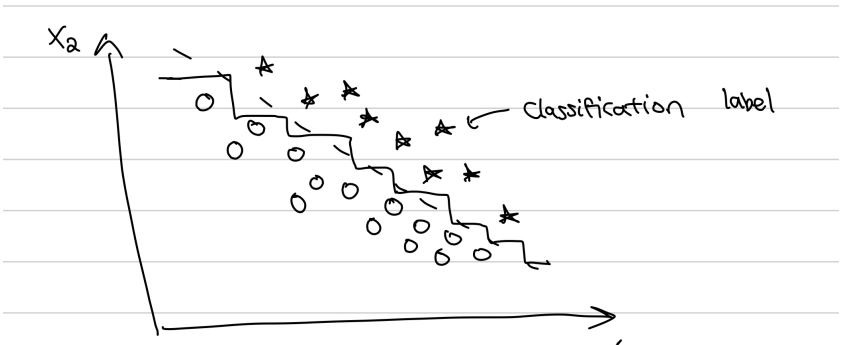
Submit a drawing of your tree showing the impurity reduction yielded by each split (including root) in your decision tree.

Solution A:



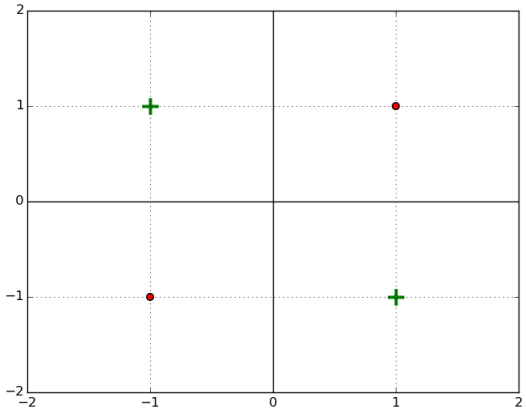
Problem B [4 points]: Compared to a linear classifier, is a decision tree always preferred for classification

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 better than a linear classifier, as exemplified above.*

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



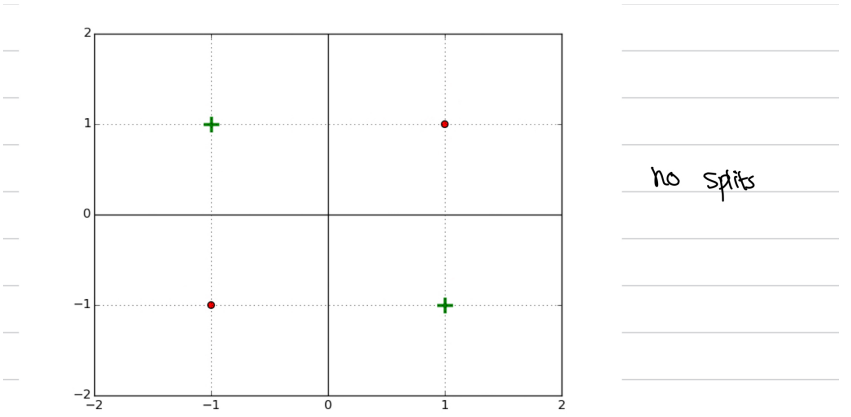
i. [5 points]: Suppose we train a decision tree on this dataset using top-down greedy induction, with the Gini index as the impurity measure. We define our stopping condition to be if no split of a node results in any reduction in impurity. Submit a drawing of the resulting tree. What is its classification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

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.)

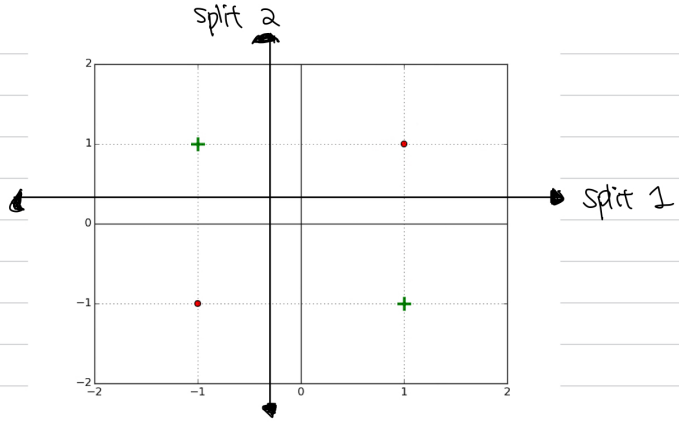
Is there any impurity measure (i.e. any function that maps the data points under a particular node in a tree to a real number) that would have led top-down greedy induction with the same stopping condition to produce the tree you drew? If so, give an example of one, and briefly describe its pros and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

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 classification error for the splits above is 50%.



We could define an new algorithm that uses the impurity to be the gini index but without stopping if there is no change in the error. This could find better solutions (as in the example above) but it could also lead to overfitting.

Taking inspiration from the way that the points are organized in the figure above, the 100 pts could be organized in some sort of alternating pattern along every dimension for difficult classification. At a minimum, each internal node would help correctly classify exactly one additional sample, except for the last one, which would classify the two remaining samples. Thus, in the worst case, there would be 99 internal nodes.

Problem 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 scenario occurs when all the dimensions are independent (there are no correlations in the data between dimensions). In each dimension, there would be on the order of N unique splits thus the number of splits to be considered would scale as D*N.*

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

Relevant materials: 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

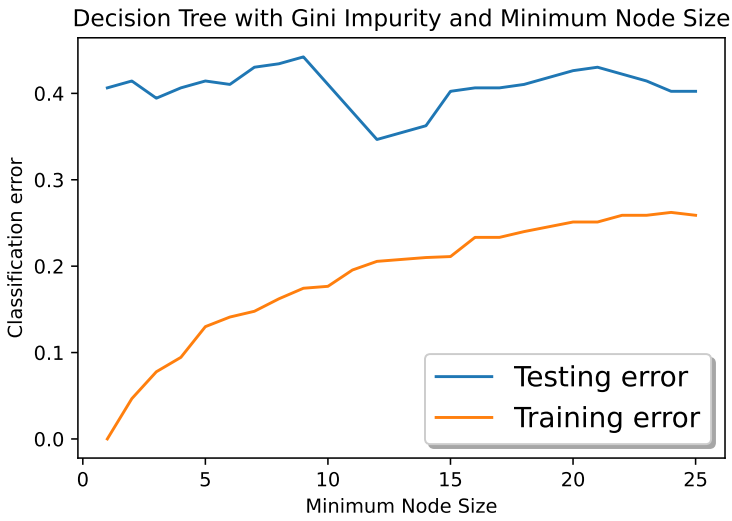
In the following question, your goal is to predict the diagnosis of diabetic retinopathy, which is the final column in the data matrix. Use the first 900 rows as training data, and the last 251 rows as validation data. Please feel free to use additional packages such as Scikit-Learn. Include your code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

Problem 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: *https://colab.research.google.com/drive/1d5_xyYNxEy9TbP0uPtmKEZ_z69R3b2DJ#scrollTo=yli92gNpDTKb*



Problem 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.

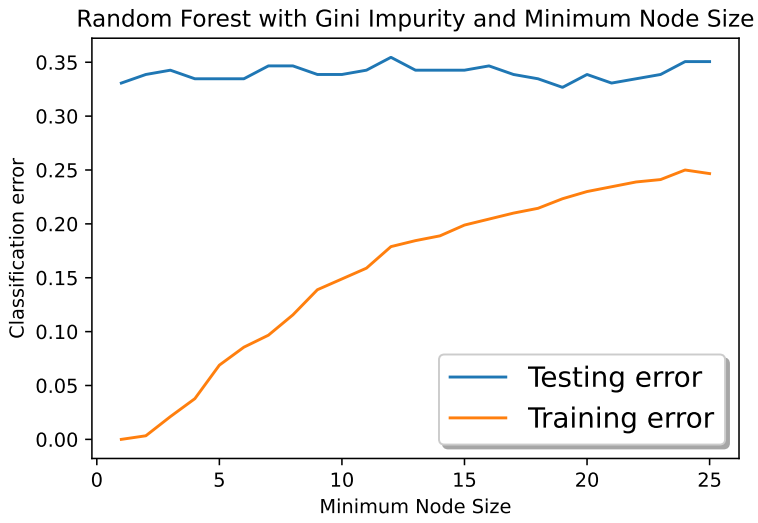
Solution B: *I will choose to analyze the minimal leaf node size. A minimum node size of around 12 minimizes the testing error. As we decrease the minimum node size, the model does not stop training as early, thus the training error decreases. When we stop training too early (minimum node size = 25), the test and train errors are both high due to high bias. When we stop the training too late (minimum node size = 1), the model is overfitting with high variance, as exemplified by low train error and high test error.*

Problem 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:



Problem 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: *Analyzing the minimal leaf node size, we can make the same conclusions as already described above. However, the mininum node size of around 19 seems to minimize the test error.*

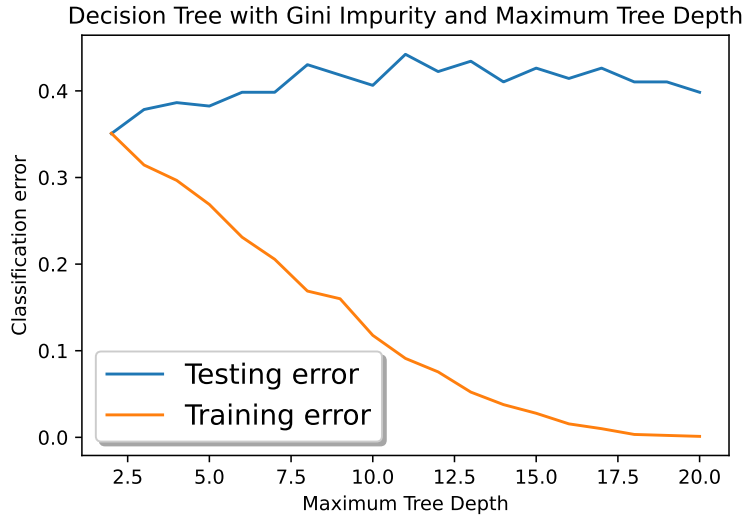
Problem E [4 points]: Do you observe any differences between the curves for the random forest and decision tree plots? If so, explain what could account for these differences.

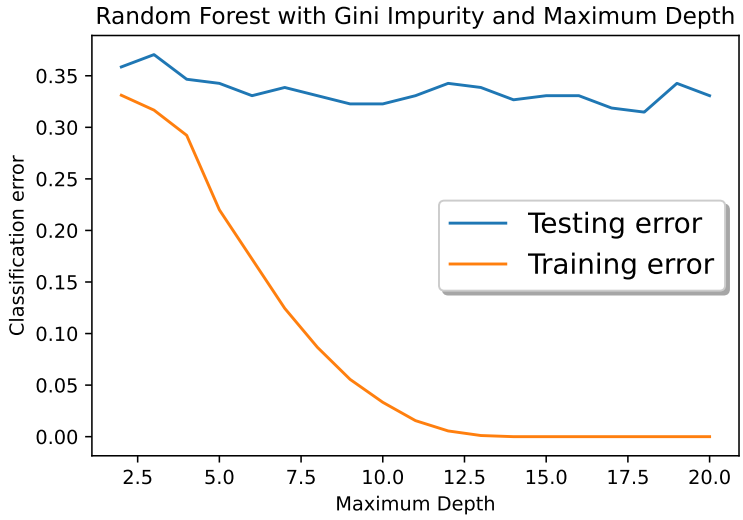
Solution E: *The random forest models generally have lower test error due to reduced bias and reduced variance from the bagging and random feature selection procedure. Furthermore, the minimum on the test error is not as pronounced because of variance reduction built-in to the random forest, which reduces the need for variance reduction from minimum node size early stopping.*

Extra Credit [7 points total] :

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

Solution F:





Problem 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 optimal maximum tree depth for the decision tree is 2. The optimal maximum tree depth is 18 for the random forest.*

3 The AdaBoost Algorithm [40 points]

Relevant materials: 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.

Problem 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: *We will show that* $\exp(-y_i f(x_i)) \geq \mathbb{1}(H(x_i) \neq y_i) \forall (x_i, y_i)$. $\mathbb{1}(H(x_i) \neq y_i)$ *is given by 1 if* $y_i f(x_i) < 0$ *and 0 if* $y_i f(x_i) > 0$. *It is easy to show that* $\exp(-y_i f(x_i)) \geq 1$ *for* $y_i f(x_i) < 0$ *and* > 0 *for* $y_i f(x_i) > 0$. *Thus, the inequality given by the summations is also true.*

Problem 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: *From lecture:*

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

Plugging this in recursively gives:

$$D_{T+1}(i) = D_1(i) \prod_{j}^{T-1} \frac{\exp\left(-\alpha_j y_i h_j(x_i)\right)}{Z_j} \tag{2}$$

Substituting $D_1(i) = \frac{1}{N}$ *gives:*

$$D_{T+1}(i) = \frac{\prod_j^T \frac{\exp\left(-\alpha_j y_i h_j(x_i)\right)}{Z_j}}{N} \tag{3}$$

where N is the number of samples.

Problem 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: *By definition,* $f(x) = \sum_{i=1}^{T} \alpha_t h_t(x)$. *Since* y_i *does not depend on* t*, we can pull it out of the summation in the expoenent. At the same time,* $\frac{1}{N}$ *does not depend on* i*, so we can pull that out of the summation, which reduces the expression to* $E = \frac{1}{N} \sum_{i=1}^{N} \exp(-y_i f(x_i))$.

Problem 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: *Substituting* $D_t(i)$ *into* Z_t*:*

$$Z_t = \frac{1}{N} \sum_{i=1}^{N} \left(\prod_{j}^{t-1} \frac{\exp\left(-\alpha_j y_i h_j(x_i)\right)}{Z_j} \right) \exp(-\alpha_t y_i h_t(x_i)) \tag{4}$$

$$Z_t = \frac{1}{N} \sum_{i=1}^{N} \left(\prod_{j}^{t-1} \frac{1}{Z_j} \right) \left(\prod_{j}^{t} \exp\left(-\alpha_j y_i h_j(x_i)\right) \right) \tag{5}$$

$$Z_t = \frac{1}{N} \sum_{i=1}^{N} \left(\prod_{j}^{t-1} \frac{1}{Z_j} \right) \exp \sum_{j}^{t} \left(-\alpha_j y_i h_j(x_i)\right) \tag{6}$$

$$\prod_{j}^{t} Z_j = \frac{1}{N} \sum_{i=1}^{N} \exp \sum_{j}^{t} \left(-\alpha_j y_i h_j(x_i)\right) \tag{7}$$

$$E = \prod_{j}^{T} Z_j \tag{8}$$

Problem 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:

$$Z_t = \sum_{i=1}^{N} D_t(i) \exp(-\alpha_t y_i h_t(x_i)) \tag{9}$$

First considering the case where $h_t(x_i) \neq y_i$:

$$Z_t = \sum_{i=1}^{N} D_t(i) \exp(\alpha_t) \tag{10}$$

$$Z_t = \exp(\alpha_t) \tag{11}$$

By definition,

$$\epsilon_t = \sum_{i=1}^{N} D_t(i) = 1 \tag{12}$$

$$Z_t = (0) \exp(-\alpha_t) + \exp(\alpha_t) = \exp(\alpha_t) \tag{13}$$

For the other case of $h_t(x_i) = y_i$:

$$Z_t = \sum_{i=1}^{N} D_t(i) \exp(-\alpha_t) = \exp(-\alpha_t) \tag{14}$$

using $\epsilon_t = 0$:

$$Z_t = \exp(-\alpha_t) + (0) \exp(\alpha_t) = \exp(-\alpha_t) \tag{15}$$

Problem 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:

$$\frac{\partial Z_t}{\partial \alpha_t} = -(1 - \epsilon_t) \exp(-\alpha_t) + \epsilon_t \exp(\alpha_t) = 0 \tag{16}$$

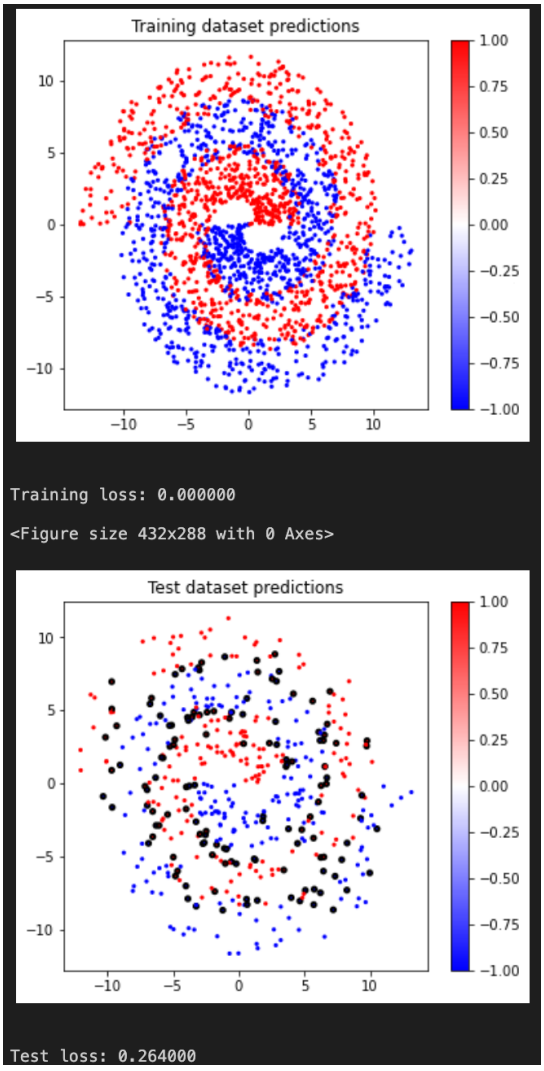
$$\frac{(1 - \epsilon_t)}{\epsilon_t} = \exp(2\alpha_t) \tag{17}$$

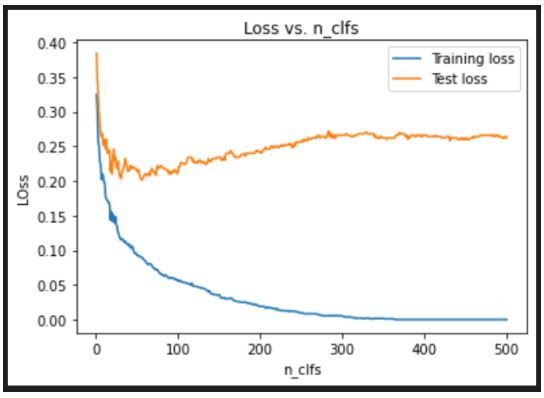
$$\alpha_t^* = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right) \tag{18}$$

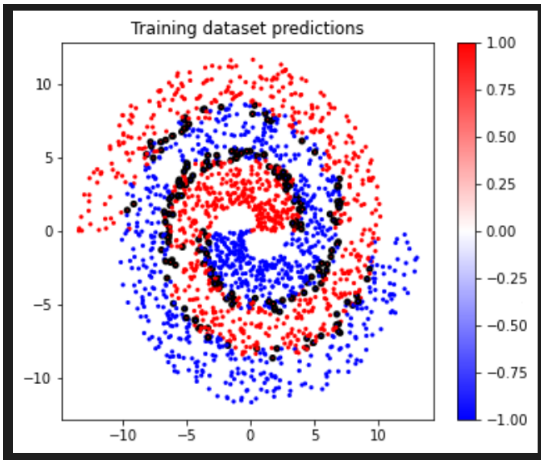
Problem G [14 points]: Implement the `GradientBoosting.fit()` and `AdaBoost.fit()` methods in the notebook provided for you. Some important notes and guidelines follow:

- 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.

Solution G: *https://colab.research.google.com/drive/1wZignuzs0exrP_8s3koG9WHtWS0GVaik#scrollTo=YpCMtATAD0jm*

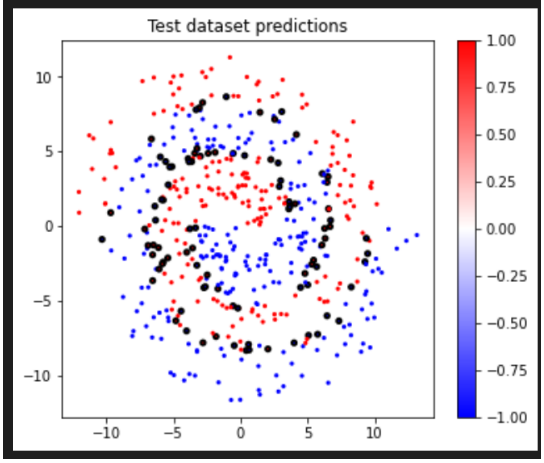

Training dataset predictions
Training loss: 0.000000
<Figure size 432x288 with 0 Axes>
Test dataset predictions
Test loss: 0.264000

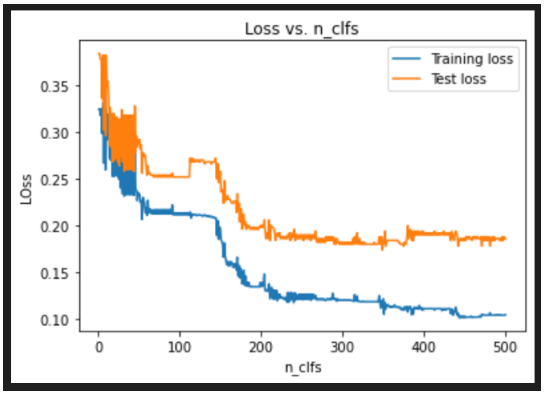

Loss vs. n_clfs
Training loss
Test loss
LOss
n_clfs




```
Training loss: 0.104500

<Figure size 432x288 with 0 Axes>
```





Problem 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: *Gradient boosting approaches a much lower training loss compared to adaboost, but adaboost achieves a lower test loss. Adaboost is a better model for this classification problem because the residuals in gradient boosting make more sense for regression problems. There is also evidence of overfitting in the gradient boosting model, as the test loss increases after a certain number of functions are fit.*

The gradient boosting loss curves are much smoother than those of adaboost. This is because in adaboost, the weights will jump back and forth as a point alternates between being correctly and incorrectly classified, whereas

in gradient boosting it uses continuous regression values.

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

Solution I: *Adaboost performed better for classification on the test dataset.*

Problem 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: *The weights are the largest for misclassified pts (those on the border of the spiral), while the weights are the smallest for consistently correctly classified pts (those that are towards the center of the spiral).*