1. The Gini impurity is $1-\sum_{k=1}^{K}\mu_k^2$. What is the maximum value of the Gini impurity among all possible $[\mu_1,\mu_2,\ldots,\mu_K$ that satisfies $\mu_k\geq 0$ and $\sum_{k=1}^{K}\mu_k=1$? Prove your answer.

$$\max G = 1-\sum_{k=1}^{k} u_k^2 \Leftrightarrow 1-\min\sum_{k=1}^{k} u_k^2$$

Lagrange multiplier

$$\min \sum_{k=1}^{k} u_k^2 \quad \text{s.t.} \sum_{k=1}^{k} u_k = 1 \Rightarrow \begin{cases} \text{let } J = \sum_{k=1}^{k} u_k^2 + \lambda\sum_{k=1}^{k} u_k \\ \text{Optimal } J \text{ is at } \frac{\partial J}{\partial u_k} = 0 \Rightarrow 2u_k = \lambda \Rightarrow u_k = \frac{-\lambda}{2} \end{cases}$$

$$\Rightarrow \text{Solving for } \lambda \Rightarrow \sum \frac{\lambda}{2} = 1 \Rightarrow \lambda = \frac{-2}{k}$$

$$\Rightarrow u_k = \frac{1}{k}$$

$$\Rightarrow \max \text{ of } \left\{G = 1-\sum_{k=1}^{k} u_k^2\right\} = 1-\frac{k}{k^2} = 1-\frac{1}{k}$$

2. Prove or disprove that the squared regression error when using binary classification, which is by definition $\mu_+(1-(\mu_+-\mu_-))^2+\mu_-(-1-(\mu_+-\mu_-))^2$ is simply a scaled version of the Gini impurity $1-\mu_+^2-\mu_-^2$.

$$\mu_+(1+(\mu_+^2+\mu_-^2-2\mu_+\mu_-)-(2\mu_+-2\mu_-))$$

$$+\mu_-(1+(\mu_+^2+\mu_-^2-2\mu_+\mu_-)+(2\mu_+-2\mu_-))$$

$$=1+\mu_+^2+\mu_-^2-2\mu_+\mu_- -2(\mu_+^2-\mu_+\mu_- -\mu_+\mu_- +\mu_-^2)$$

$$=1-\mu_+^2-\mu_-^2+2\mu_+\mu_- = 1-\mu_+^2-\mu_-^2+2\mu_+(1-\mu_+)$$

3. If bootstrapping is used to sample $N' = pN$ examples out of N examples and N is very large, argue that approximately $e^{-p} \cdot N$ of the examples will not be sampled at all.

Chance not getting picked $= (1 - \frac{P}{N})$ for 1 set

Chance of NEVER picked $= \lim_{k \to \infty} (1 - \frac{P}{N})^k$

since $e^x = \lim_{n \to \infty} (1 + \frac{x}{n})^n$, and N is very large

$\Rightarrow \lim_{k \to \infty} (1 - \frac{P}{N})^k \approx \lim_{k \to \infty} (1 - \frac{P}{k})^k = e^{-P}$

$\Rightarrow$ Expect amount $= P(x) \cdot N = e^{-P} \cdot N$ will never be sampled.

4. Consider a Random Forest G that consists of K binary classification trees $\{g_k\}_{k=1}^{K}$, where K is an odd integer. Each g_k is of test 0/1 error $E_{\text{out}}(g_k) = e_k$. Prove or disprove that $\frac{2}{K+1}\sum_{k=1}^{K} e_k$ upper bounds $E_{\text{out}}(G)$.

let $e_{k,i} = [g_k(x_i) \neq y_i]$

$$\left.\begin{aligned} e_{k,i} &= [g_k(x_i) \neq y_i] \\ \varepsilon_i &= \sum_{k=1}^{K} e_{k,i} \end{aligned}\right\} \Rightarrow E_{out} = \sum_{i=1}^{N} \frac{[\varepsilon_i \geq \frac{k+1}{2}]}{N}$$

$$\Rightarrow E_{out} \leq \frac{2}{k+1} \sum_{i=1}^{N} \frac{\varepsilon_i}{N} = \frac{2}{k+1} \sum_{i=1}^{N} \sum_{k=1}^{K} \frac{e_{k,i}}{N} = \frac{2}{k+1} \sum_{k=1}^{K} \sum_{i=1}^{N} \frac{e_{k,i}}{N}$$

$$\Rightarrow E_{out} \leq \frac{2}{k+1} \sum_{k=1}^{K} e_k$$

5. For the gradient boosted decision tree (with squared error), if a tree with only one constant node is returned as g_1, and if $g_1(\mathbf{x}) = 11.26$, then after the first iteration, all s_n is updated from 0 to a new constant $\alpha_1 g_1(\mathbf{x}_n) = 11.26\alpha_1$. What is α_1 in terms of all the $\{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$? Prove your answer.

$$\alpha_1 = \text{OneVarLinearReg}\left(\{(g_1(x_n), y_n - s_n)\}\right)$$

$$= \min_{\alpha_1} \frac{1}{N} \sum_{n=1}^{N} \left((y_n - s_n) - \alpha_1 g_1(x_n)\right)^2$$

$$E = \min_{\alpha_1} \frac{1}{N} \sum_{n=1}^{N} \left(y_n^2 + (11.26\alpha_1)^2 - 22.52\alpha_1 y_n \right)$$

$$\frac{\partial E}{\partial \alpha_1} = \min \frac{1}{N} \sum_{n=1}^{N} \left(253.5752\alpha_1 - 22.52 y_n\right)$$

$$= 11.26 \min \frac{1}{N} \sum \left(22.52\alpha_1 - 2y_n\right)$$

$$\alpha_1 = \sum_{n=1}^{N} \frac{y_n}{11.26}$$

6. For the gradient boosted decision tree (with squared error), after updating all s_n in iteration t using the steepest η as α_t, what is the value of $\sum_{n=1}^{N} s_n g_t(\mathbf{x}_n)$? Prove your answer.

$$S_n^t = S_n^{t-1} + \alpha_t g_t(X_n)$$

$$= S_n^{t-2} + \alpha_{t-1} g_{t-1}(X_n) + \alpha_t g_t(X_n)$$

$$\vdots$$

$$= \sum_{i=1}^{t} \alpha_t g_t(X_n)$$

$$\sum_{n=1}^{N} S_n g_t(X_n) = \sum_{n=1}^{N} \left(\sum_{i=1}^{t} \alpha_i g_i(X_n) \right) \cdot g_t(X_n)$$

7. If gradient boosting (with squared error) is coupled with squared-error polynomial regression (without regularization) instead of decision trees. Prove or disprove that the optimal $\alpha_1 = 1$.

$$E = \min_{\alpha_1} \sum ((y_n - \underbrace{s_n}_{=0}) - \alpha g_1(x_n))^2$$

$$= \min_{\alpha} \sum (y_n - \alpha g_1(x_n))^2 \quad \ldots \text{①}$$

$$g_1 : \min_{g} \sum (y_n - g(x_n))^2 \quad \ldots \text{②}$$

if $\alpha \neq 1 \Rightarrow E = \sum (y_n - g_1'(x_n))^2 \ldots$ ③ where $g_1'(x_n) = \alpha_1 g_1(x_n) \neq g_1(x_n)$

but g is optimal in minimizing ② $\Rightarrow$ ② $\leq$ ③, since $g_1'(x_n) \neq g_1(x_n)$

not equality $\Rightarrow$ ② < ③

$\Rightarrow$ then there exists g where E is better than ③

$\Rightarrow g'$ is not optimal $\rightarrow\leftarrow$

$\Rightarrow g'(x_n) = g(x_n) \Rightarrow \alpha_1 = 1$

8. Consider Neural Network with sign(s) instead of tanh(s) as the transformation functions. That is, consider Multi-Layer Perceptrons. In addition, we will take $+1$ to mean logic TRUE, and -1 to mean logic FALSE. Assume that all x_i below are either $+1$ or -1. Write down the weights w_i for the following perceptron

$$g_A(\mathbf{x}) = \text{sign}\left(\sum_{i=0}^{d} w_i x_i\right).$$

to implement

$$\texttt{OR}\,(x_1, x_2, \ldots, x_d)\,.$$

Explain your answer.

Using logic notation

$\overline{\bar{A} \cdot \bar{B}} = A + B$

$\overline{\bar{A} \cdot \bar{B} \cdot \bar{C}} = A + B + C$

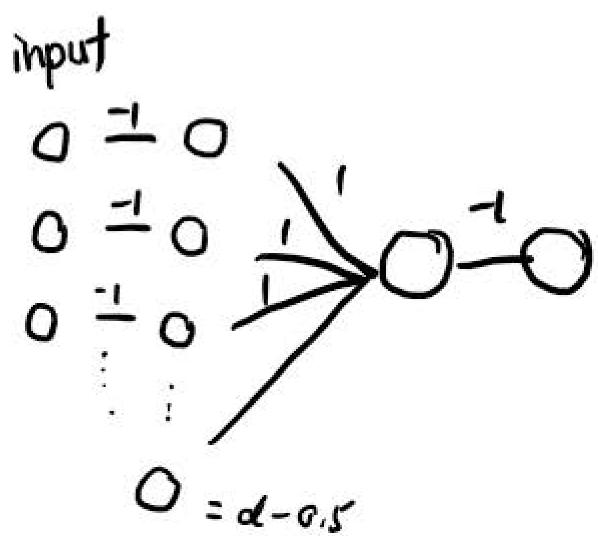
$\overline{\bar{A} \cdot \bar{B} \cdot \bar{C} \cdot \bar{D}} = A + B + C + D$

$\vdots$

First layer, NOT $\Rightarrow w_i = 1$

Second layer, AND $\rightarrow w_1 = 1$, $b = d - 0.5$

Third layer, NOT $\rightarrow w_1 = 1$



9. For a Neural Network with at least one hidden layer and $\tanh(s)$ as the transformation functions on all neurons (including the output neuron), when all the initial weights $w_{ij}^{(\ell)}$ are set to 0, what gradient components are also 0? Justify your answer.

Output layer : $\delta_i^{(L)} = -2\,(y_n - s_1^{(L)}) \cdot (x_i^{(L-1)})$

$$= -2\left(y_n - \sum \underbrace{w_{i1}^{(L)}}_{\to 0} x_i^{(L-1)}\right) \cdot (x_i^{(L-1)})$$

$$= -2\, y_n x_i^{(L-1)}$$ Not Zero

Other layers : $\delta_j^{(\ell)} = \sum_k \left(\delta_k^{(\ell+1)}\right)\left(\underbrace{w_{jk}^{(\ell+1)}}_{\to 0}\right)\left(\tanh'(s_j^{(\ell)})\right)$

$$= 0$$

Gradient Components on all but the last layer are 0 on the first run.

10. Multiclass Neural Network of K classes is typically done by having K output neurons in the last layer. For some given example $(\mathbf{x}, y)$, let $s_k^{(L)}$ be the summed input score to the k-th neuron, the joint "softmax" output vector is defined as

$$\mathbf{x}^{(L)} = \left[\frac{\exp(s_1^{(L)})}{\sum_{k=1}^{K} \exp(s_k^{(L)})}, \frac{\exp(s_2^{(L)})}{\sum_{k=1}^{K} \exp(s_k^{(L)})}, \ldots, \frac{\exp(s_K^{(L)})}{\sum_{k=1}^{K} \exp(s_k^{(L)})} \right].$$

It is easy to see that each $x_k^{(L)}$ is between 0 and 1 and the the components of the whole vector sum to 1. That is, $\mathbf{x}^{(L)}$ defines a probability distribution. Let's rename $\mathbf{x}^{(L)} = \mathbf{q}$ for short.

Define a one-hot-encoded vector of y to be

$$\mathbf{v} = [[\![y = 1]\!], [\![y = 2]\!], \ldots, [\![y = K]\!]].$$

The cross-entropy loss function for the Multiclass Neural Network, much like an extension of the cross-entropy loss function used in logistic regression, is defined as

$$e = -\sum_{k=1}^{K} v_k \ln q_k.$$

Prove that $\frac{\partial e}{\partial s_k^{(L)}} = q_k - v_k$ which is actually the $\delta_k^{(L)}$ that you'd need for backprop.

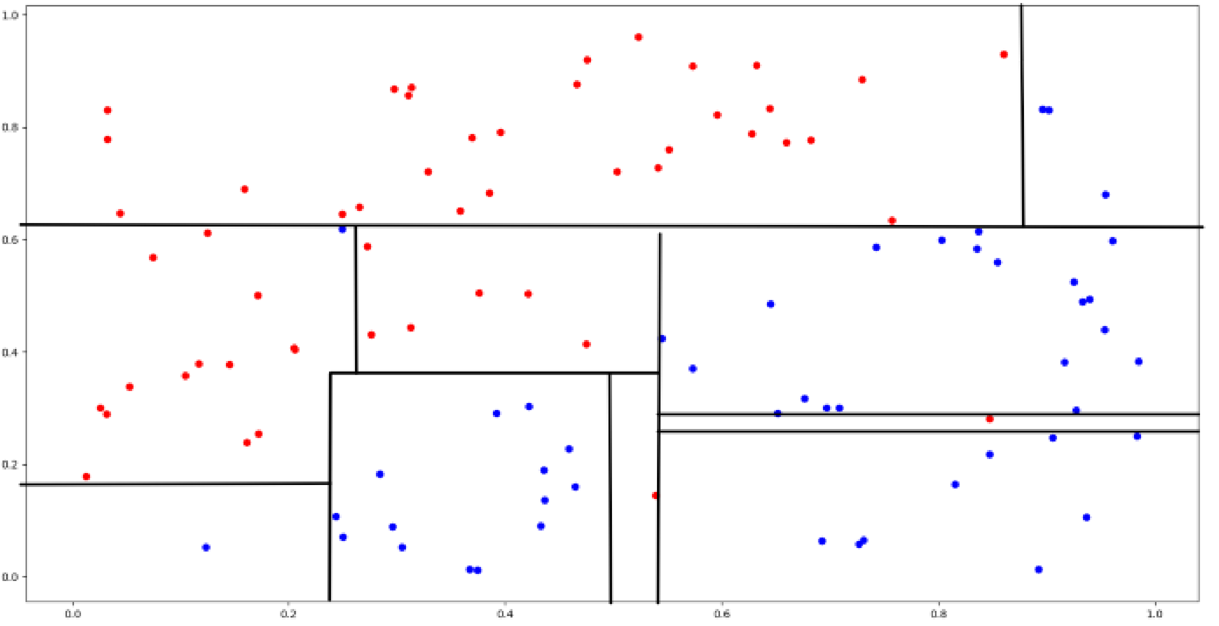
$$\frac{\partial e}{\partial s} = \frac{\partial(-\sum v_k \ln q_k)}{\partial s} = \frac{\partial\left(-\sum v_k \ln \frac{\exp s_k^L}{\sum \exp s_j^L}\right)}{\partial s} = \frac{\partial\left(-\sum v_k \left(s_k^L - \ln\left(\sum \exp s_j^L\right)\right)\right)}{\partial s}$$

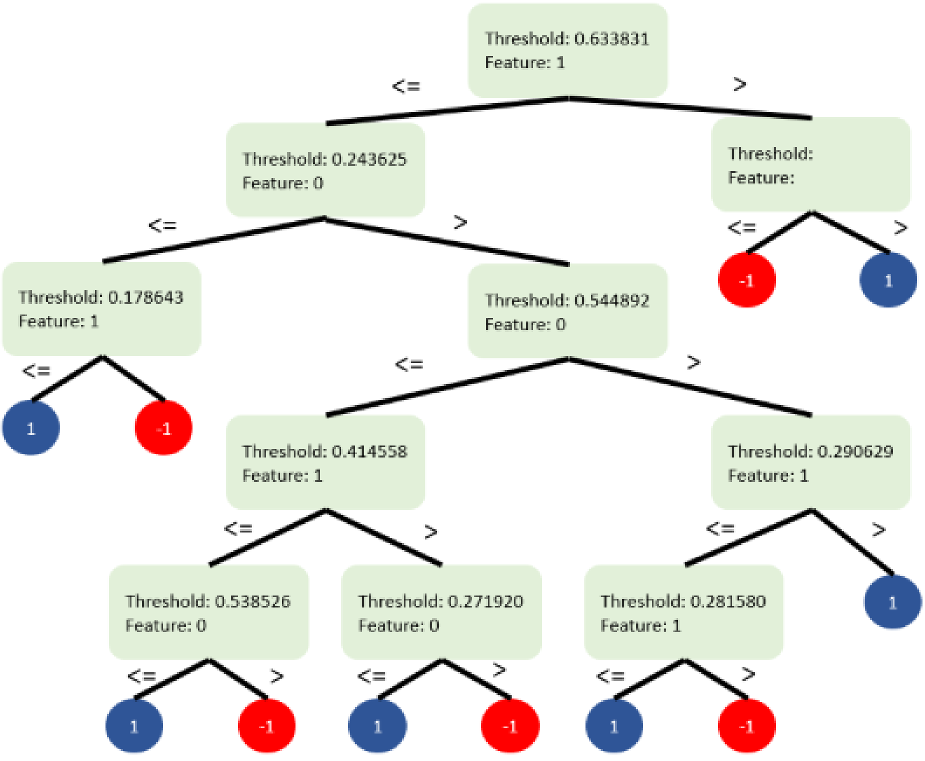
$$= v - \left(v \cdot \left(\frac{\exp s_k^L}{\sum \exp s_j^L}\right)\right) = -v + vq$$

From one hot encoding, $v_k = 1$ for δ_j^L if $k = j$

$$\Rightarrow \delta_k^L = 1 q_k - v_k = q_k - v_k$$

11. (*) Draw the resulting tree (by program or by hand, in any way easily understandable by the TAs).

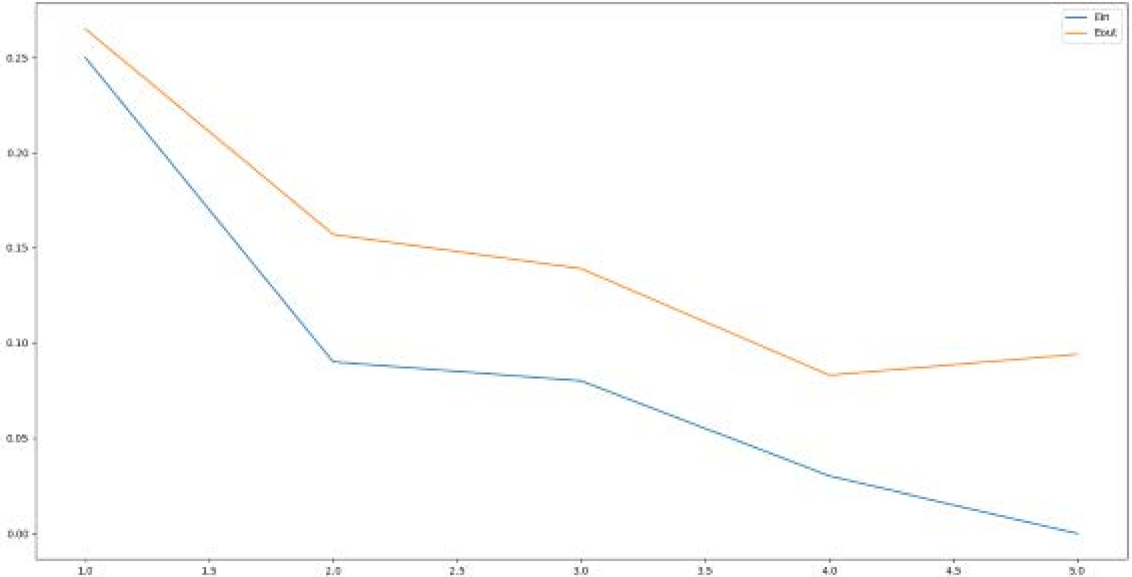




12. (*) Continuing from the previous problem, what is E_{in} and E_{out} (evaluated with 0/1 error) of the tree?

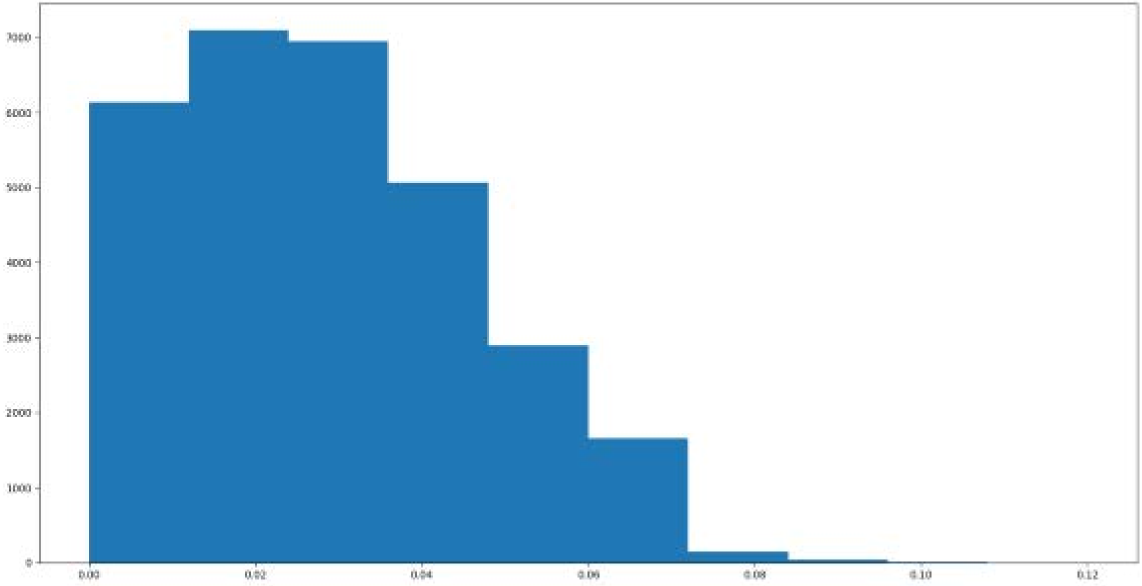
```
Ein:    0.0
Eout:   0.094
```

13. (*) Assume that the tree in the previous question is of height H. Try a simple pruning technique of restricting the maximum tree height to $H-1, H-2, \ldots, 1$ by terminating (returning a leave) whenever a node is at the maximum tree height. Call g_h the pruned decision tree with maximum tree height h. Plot curves of h versus $E_{\text{in}}(g_h)$ and h versus $E_{\text{out}}(g_h)$ using the 0/1 error in the same figure. Describe your findings.

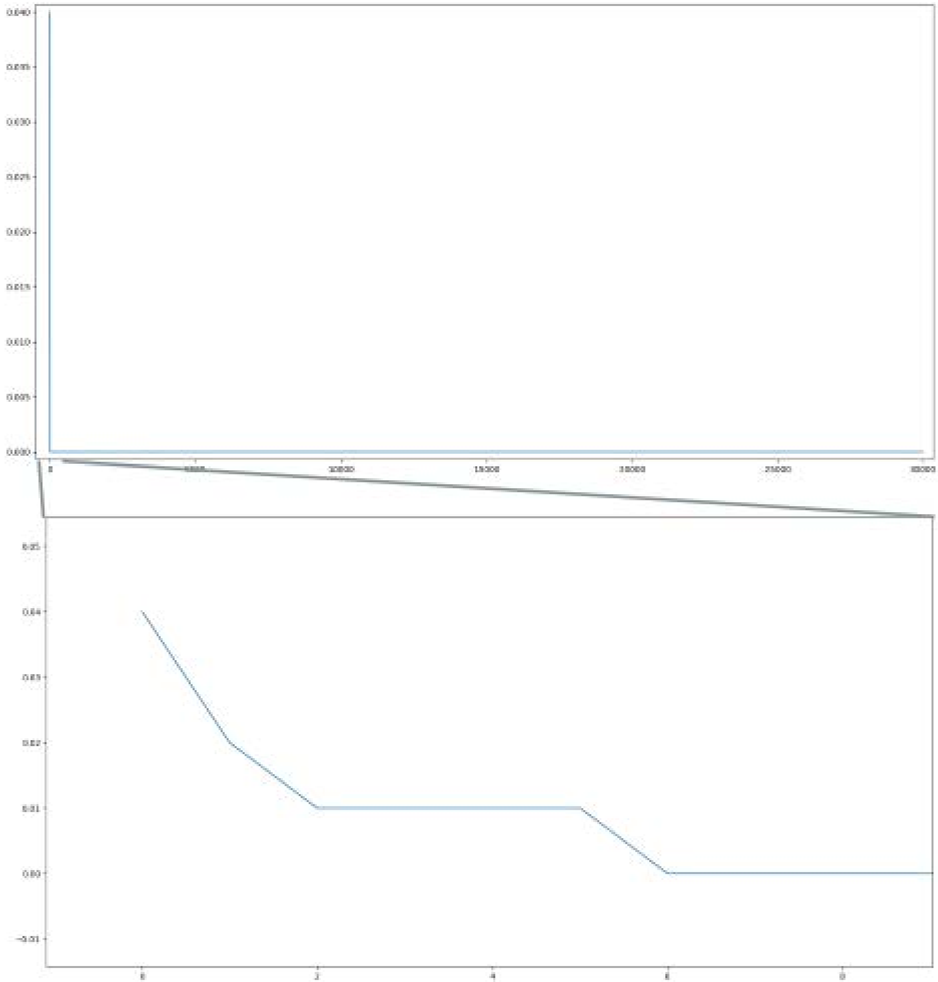


Ein continuously increases the more pruned the tree is, as the tree is designed to decrease Gini Impurity with every level.
Eout with a single level pruned has the lowest error, because the fully pruned tree isn't regularized, and so overfits the training model. Over pruned trees leads to not being able to separate the data properly

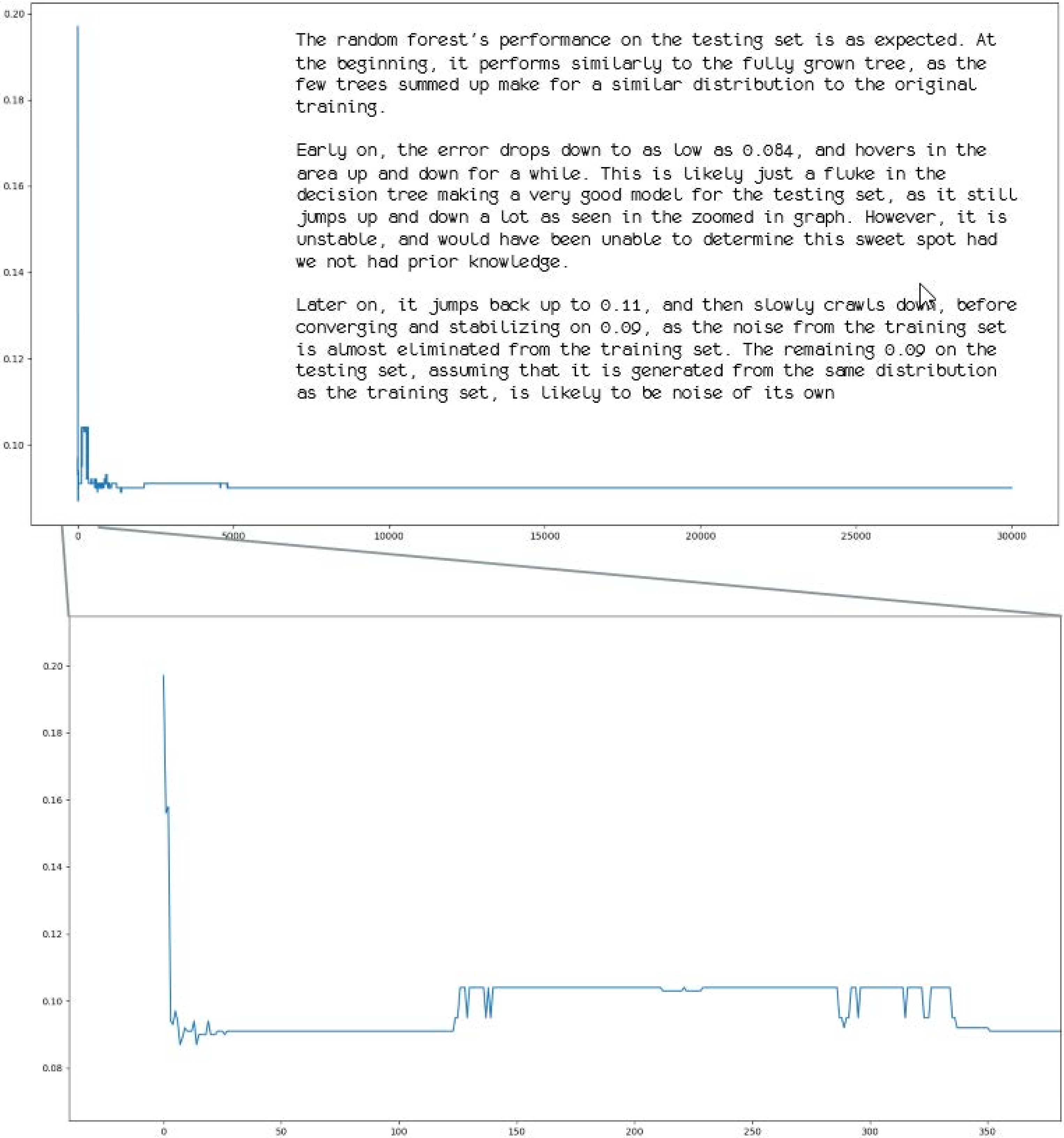
14. (*) Plot a histogram of $E_{\text{in}}(g_t)$ over the 30000 trees.



15. (*) Let G_t = "the random forest with the first t trees". Plot a curve of t versus $E_{\text{in}}(G_t)$.



16. (*) Continuing from Question 15, and plot a curve of t versus $E_{\text{out}}(G_t)$. Briefly compare with the curve in Question 15 and state your findings.



18. (10%) Prove that it is impossible to implement $\texttt{XOR}((x)_1, (x)_2, \ldots, (x_d))$ with any d-$(d-1)$-1 feed-forward neural network with $\text{sign}(s)$ as the transformation function.

For $XOR(x_1, x_2)$, NN: 2-1-1 is functionally the same as 2-1, effectively a single perceptron.

Since XOR is not linearly separable, it is impossible to mode with a single perceptron. - ①

By Q17, we know the minimum to solve is with a 2-2-1 net - ②

Let $XOR(x_1, x_2 \ldots x_d)$ require minimum of d-d-1 net to solve

Since $XOR(x_1, x_2, \ldots x_{d+1})$, is the same as $XOR(XOR(x_1, x_2 .. x_d), x_{d+1})$,

$XOR(x_1, x_2, \ldots x_{d+1})$ requires at least 1 more perceptron to out.

$\Rightarrow XOR(x_1, x_2 \ldots x_{d+1})$ requires at least $(d+1)$-$(d+1)$-1 net to solve

By induction, XOR is unsolvable by d-$(d-1)$-1 net.