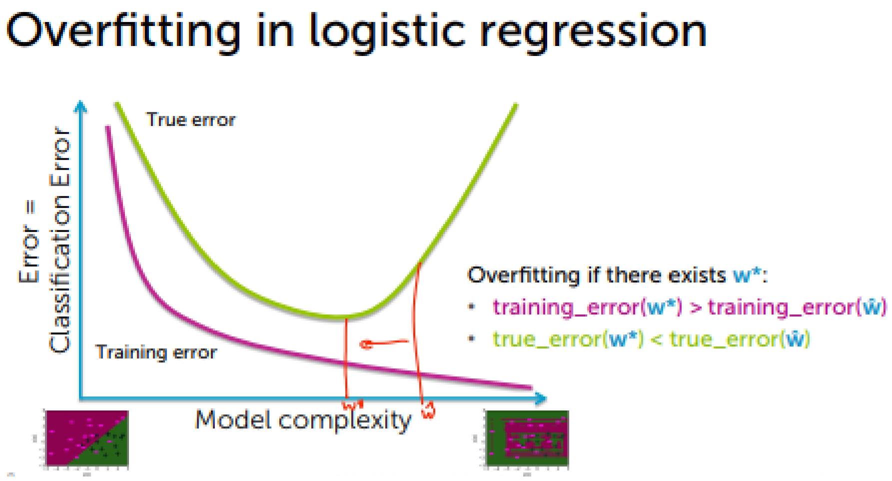
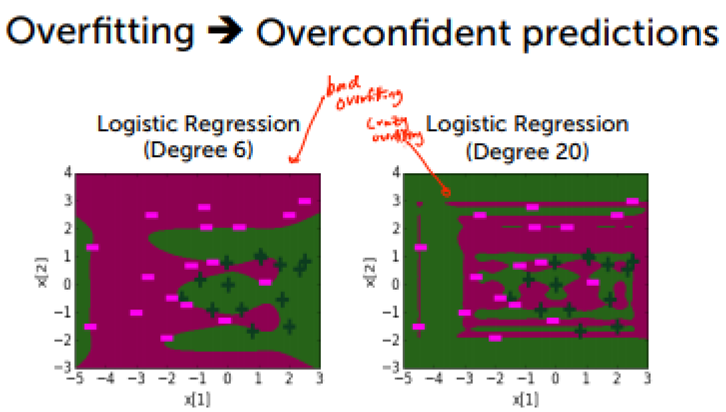
Overfitting in Decision Trees

Overfitting in logistic Regression

1. Overfitting occurs with increasing model complexity.
    - Overfitting if there exists w * :
        - training error(w *) > training_error(w-hat)
        - true_error(w *) < true_error(w-hat)
2. Overfitting leads to Overconfident predictions
    - Polynomial degree of the model increases the decision boundary become complex -> indication of overfitting.





Overfitting in Decision Trees

1. As the depth of the model increases, the training error becomes 0, case of **overfitting**.

Training Error reduces with depth - Since at each level/depth the feature with least classification/training error is choosen to split and make decision.

Means to avoid overfitting in Decision Trees

Occam's Razor

- **Principle of Occam's Razor : "Among competing hypothesis, the one with fewest assumptions should be selected."** - William of Occam, 13th Century

- **Occam's Razor for decision trees : When two trees have similar classification error on the validation set, pick the simpler one**.

Complexity	Train error	Validation error
Simple	0.23	0.24
Moderate	0.12	0.15
Complex	0.07	0.15
Super complex	0	0.18

Same validation error

pick

bad!

Overfit

- A simpler tree will have lower depth/level.

Decision tree learning problem modified

```
- **Find a "simple" decision tree with low classification error**.
```

Picking simpler trees

1. **Early Stopping** - Stop learning algorithm **before** tree becomes too complex.
2. **Pruning** - Simplify tree **after** learning algorithm terminates.

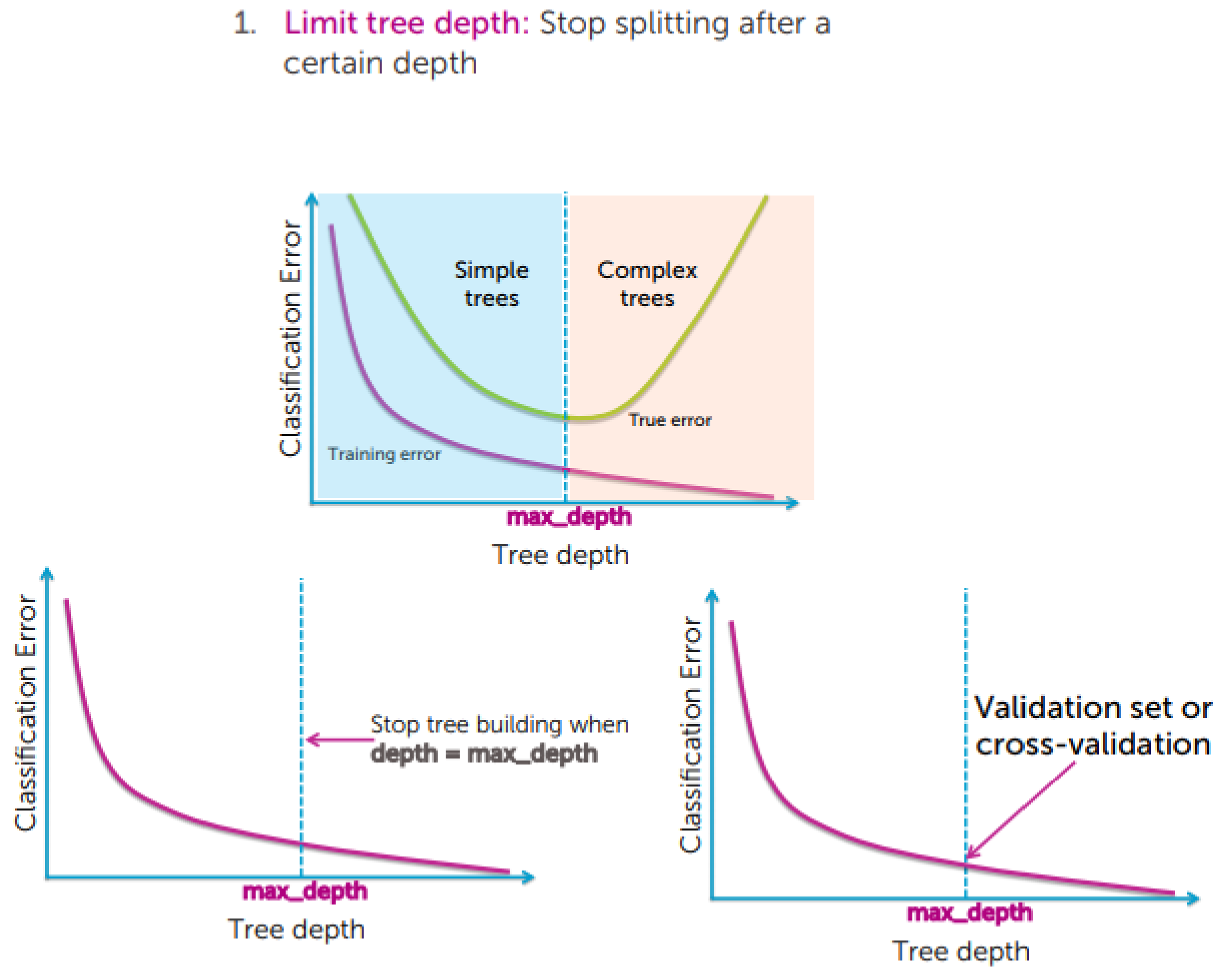
1. Early stopping in learning decision trees

- Deeper trees - increase model complexity and leads to overfitting.

3 conditions:

1. Limit the depth of the tree.

   - For a graph of **Classification Error** vs **Tree depth**.
     - Training error decreases as model complexity increases.
     - True error decreases till a certain point and there after increases.
     - The depth at which the generalization/true error is the least the a parameter called max_depth must be selected.
     - Everything to the left of the max_depth parameter are simpler trees and ecerything on the right of the max_depth parameter are complex trees.
     - Therefore since simpler trees are to be choosen -> **stop tree building when depth = max_depth** .
     - **The max_depth parameter selection must be done on the validation set or cross-validation**.

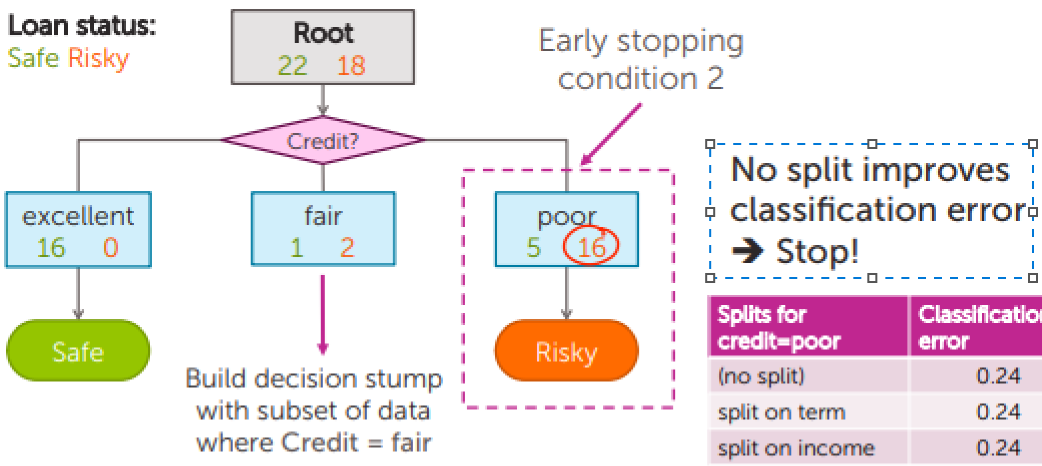


---

2. Use classification error to limit depth of tree

- In general if the classification error - based on which a feature is choosen to split the datatset further, provides poor or not much improvement (minimization) in the classification error then can stop splitting.
- Can employ a threshold parameter, in case the classifcation error is below the thresholf value then can ignore splitting based on that feature.
- Practical uses.

2. Classification error: Do not consider any split that does not cause a sufficient decrease in classification error

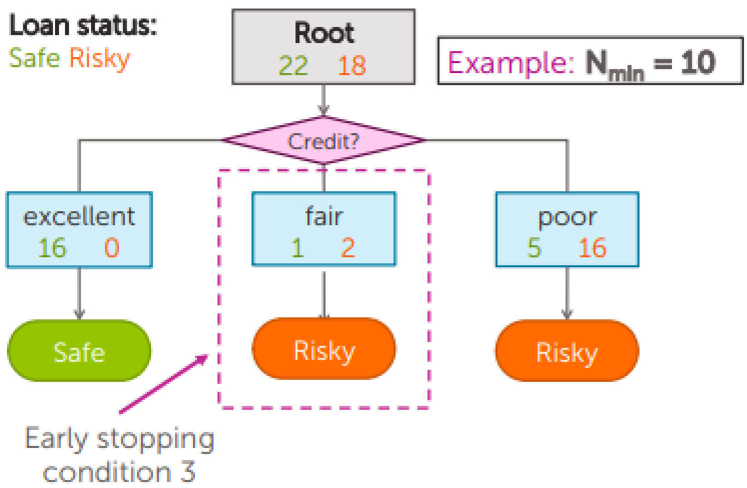


Splits for credit=poor	Classification error
(no split)	0.24
split on term	0.24
split on income	0.24

3. Datapoints contained in a node is too small - Stop

- If a node contains very few data-points then no need to further split it.
- Choose a threshold value (Nmin = 10, Nmin =100(large datatset)).
- Stop when the datapoints in a node <= Nmin.

Early stopping condition 3: Stop when data points in a node <= N_{min}



Greedy decision tree learning

- Step 1 - Start with an empty tree.
- Step 2 - Select a feature to split data.
- For each split of the tree:
  - Step 3 - If nothing more to do **(Stopping Conditions)** - make predictions.
  - Step 4 - Otherwise, go to Step 2 and continue (recurse) on this split.

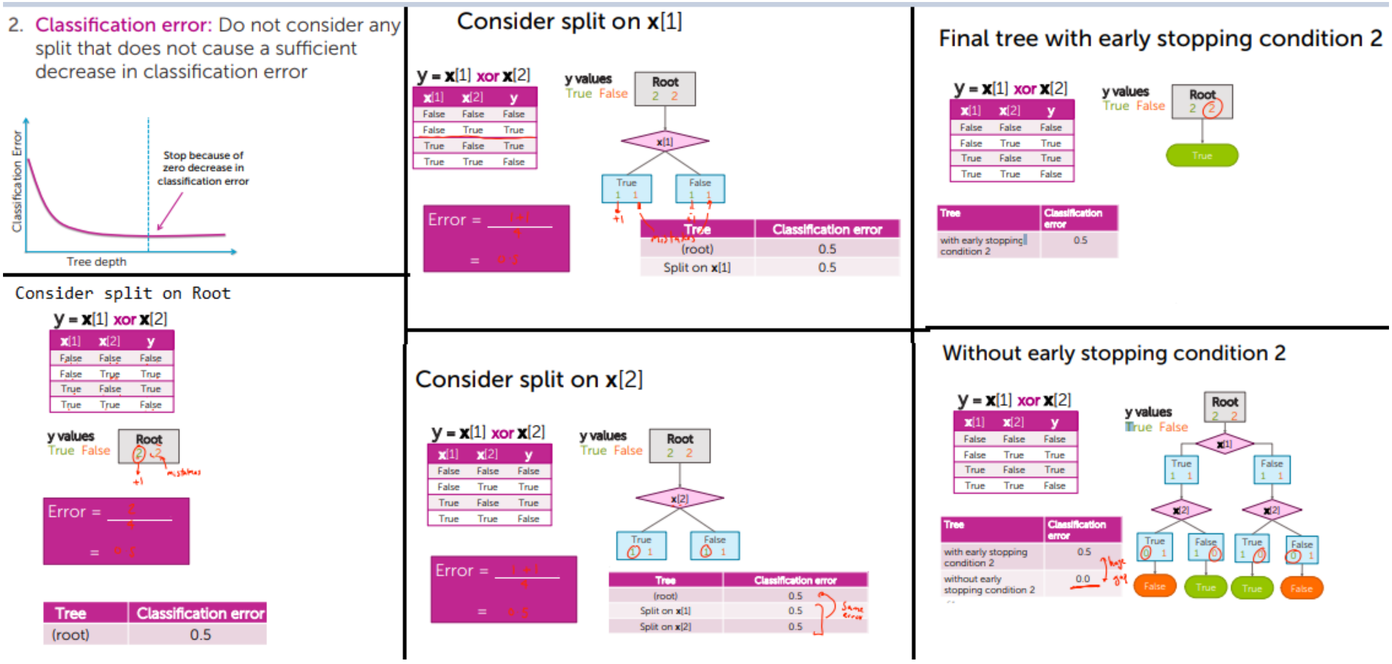
Stopping Conditions

```
1. When all data points agree to the true label y.
2. When no more features to split on exists.

(Early Stopping conditions)
1. Limit the tree depth - Stop splitting after a certain depth.
2. Classification error - Don't consider any split that doesn't have sufficient cl
assification error.
3. Minimum node 'size'- Do not split an intermediate node which contains too few d
ata points.
```

Pruning- Overfitting in Decision Trees

- The stopping conditions listed above have certain limitations.
- Early Stopping Condition 1 - **Limit the depth of the tree**.
  - Hard to know exactly when to stop.
  - Hard to get the max_depth parameter.
- Early stopping condition 2 - **No improvement in the classification error**
  - The classification error does not necessarily constantly decrease with increase in tree dept.
  - **Pros** - A reasonable heuristic for early stopping to avoid useless splits.
  - **Cons** - Too short sighted - Could miss "good" splits that may occur right after "useless" splits.



The scenarios are listed below

- At the root -> classification error -> 0.5
- When the dataset is split on feature 1 x[1] -> classification error -> 0.5
- When the dataset is split on feature 2 x[2] -> classification error -> 0.5
- Accord to stopping condition 2, if there is no improvement in the classification error, then the splitting process can stop.
- It can be note that -> With stopping condition 2 -> the splitting does not occur further than the root and has a **Classification Error = 0.5**.

Without the early stooping condition, if the tree was allowed to grow until the leaf nodes, then the training error would be absolutly 0.

Pruning

- Train a complex tree and then simplify it later.
- A simpler tree
  - has less depth or levels.
  - has fewer leaf nodes.

Need to balance "Simplicity" & "Predictive power"

- Need to balance between
  - Too complex, risk of overfitting.
  - Too simple, high classification error.

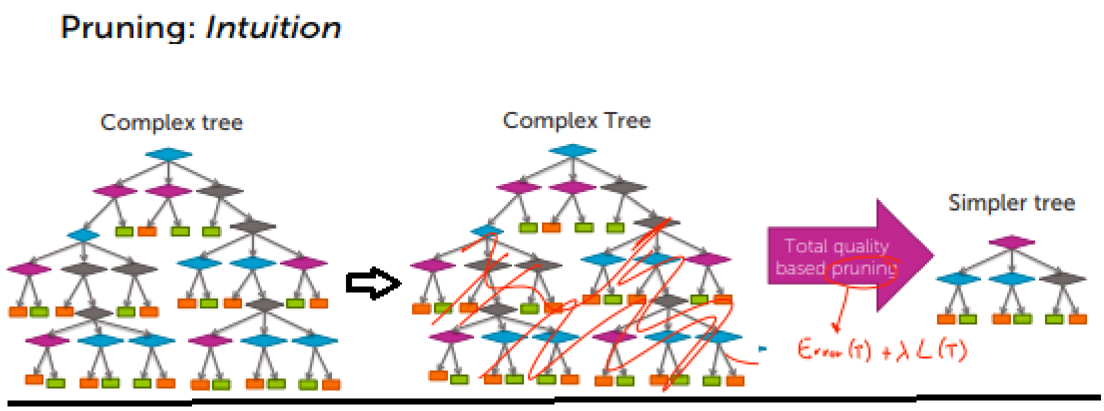
Desired Total quality format

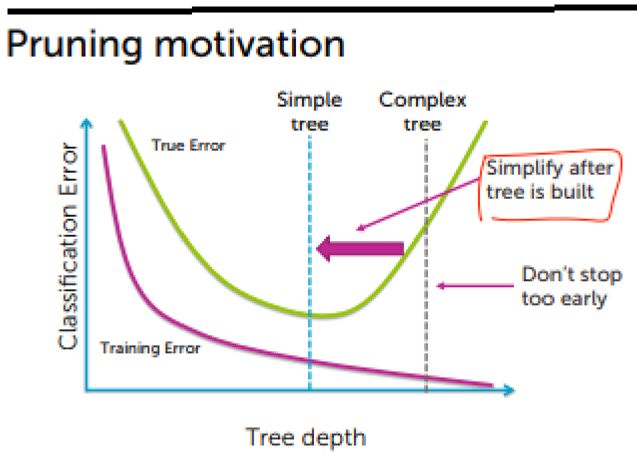
- Total cost = measure of fit + measure of complexity

```
(classification error)   (Large # -> likely overfit)
(Large # -> bad fit training data)
```

- Total cost = classification error + number of leaf nodes = Error(T) + L(T)
- Total cost C(T) = Error(T) + λ L(T)
- **λ - tuning parameter**
  - λ = 0 -> standard decision tree learning.
  - λ = infinity -> infinite penalty -> y-hat will be majority class. Prediction will be biased towards the majority class.
  - λ in between -> Balance the fit and complexity.

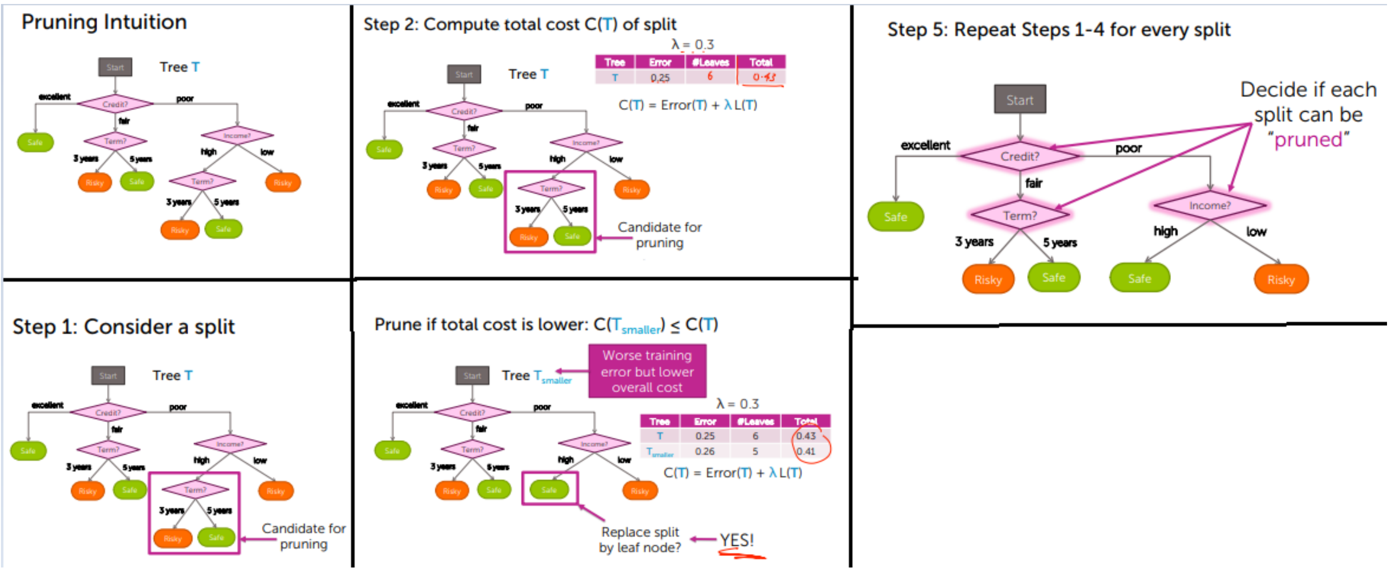
Use the total quality to simplify the trees.





Tree Pruning Algorithm

- It follows a bottom-top approach.
- Consider a tree split. (Fincancial data - tree)
- Step 1 : Consider a split (bottom - most feature)
- Step 2 : Compute the cost C(T) of the split
  - C(T) = Error(T) + λ L(T)
    - Consider λ = 0.3, Error = 0.25
    - C(T) = 0.43
- Step 3 : Retrace that split, and replace the split node and calculate the classification error on the newly formed tree (Tsmaller).
- Step 4 : If the Total cost for the **Tsmaller** is less than the **Tree (previous instance)** then replace it or else **undo the split on Tsmaller**.
  - In case Tsmaller classification error is lower then tree classification error than **Prune**. (In this case -> Worse training error but lower overall cost.)
- Step 5 : Repeat the above steps for every split.
  - Based on the classification error decide if the split can be pruned.



Decision tree pruning algorithm

- Start at bottom of tree T and traverse up, apply *prune_split* to each decision node M
- *prune_split(T,M):*
  1. Compute total cost of tree T using $C(T) = Error(T) + \lambda L(T)$
  2. Let $T_{smaller}$ be tree after pruning subtree below M
  3. Compute total cost complexity of $T_{smaller}$ $C(T_{smaller}) = Error(T_{smaller}) + \lambda L(T_{smaller})$
  4. If $C(T_{smaller}) < C(T)$, prune to $T_{smaller}$

Quiz

1. (True/False) When learning decision trees, smaller depth USUALLY translates to lower training error.

- ○ True
- ◉ False

- Learning decision trees with smaller depth usually translates to lower training error.

---

2. (True/False) If no two data points have the same input values, we can always learn a decision tree that achieves 0 training error.

- ◉ True
- ○ False

---

3. (True/False) If decision tree T1 has lower training error than decision tree T2, then T1 will always have better test error than T2.

- ○ True
- ◉ False

- If over/unerfit the test error will be different.

---

4. Which of the following is true for decision trees?

- ○ Model complexity increases with size of the data.
- ◉ Model complexity increases with depth.
- ○ None of the above

---

5. Pruning and early stopping in decision trees is used to

- ◉ combat overfitting
- ○ improve training error
- ○ None of the above

- "Pruning is a technique in machine learning that reduces the size of decision trees by removing sections of the tree that provide little power to classify instances. Pruning reduces the complexity of the final classifier, and hence improves predictive accuracy by the reduction of overfitting."

---

6. Which of the following is NOT an early stopping method?

- ○ Stop when the tree hits a certain depth
- ○ Stop when node has too few data points (minimum node "size")
- ◉ Stop when every possible split results in the same amount of error reduction
- ○ Stop when best split results in too small of an error reduction

- "early stopping is a form of regularization used to avoid overfitting when training a learner with an iterative method, such as gradient descent."

---

7. Consider decision tree T1 learned with minimum node size parameter = 1000. Now consider decision tree T2 trained on the same dataset and parameters, except that the minimum node size parameter is now 100. Which of the following is always true?

- [x] The depth of T2 >= the depth of T1
- [x] The number of nodes in T2 >= the number of nodes in T1
- [ ] The test error of T2 <= the test error of T1
- [x] The training error of T2 <= the training error of T1

- This would be true for the training error. More nodes would reduce the size of remaining training errors. At worst, there will be more nodes with same total error on training.
- Test error may or may not be T2 <= T1. Overfitting is possible.

---

8. Questions 8 to 11 refer to the following common scenario:

Imagine we are training a decision tree, and we are at a node. Each data point is (x1, x2, y), where x1,x2 are features, and y is the label. The data at this node is:

x1	x2	y
0	1	+1
1	0	+1
0	1	+1
1	1	-1

What is the classification error at this node (assuming a majority class classifier)?

0.25

```
x1   x2   y
0    1    +1 (duplicate of row 3)
1    0    +1
0    1    +1 (duplicate of row 1)
1    1    -1

three values of y = +1, one value of y=-1.
error = 1/4 = 0.25

answer = 0.25  [CORRECT]

-------------
majority class classifier.
```

---

9. Refer to the scenario presented in Question 8.

If we split on x1, what is the classification error?

0.25

```
x1    x2    y
0     1     +1
1     0     +1
0     1     +1
1     1     -1

split on x1. conclude when x1=0, y=1. when x1=0, y=-1 with 1/4 error.

x1    x2    y
0     1     +1
0     1     +1
--
1     1     -1
1     0     +1 [misclassified]
error = 0.25   [CORRECT]
```

---

10. Refer to the scenario presented in Question 8.

If we split on x2, what is the classification error?

0.25

```
x1    x2    y
0     1     +1
0     1     +1
1     1     -1 [misclassified]
---
1     0     +1


error = 0.25
```

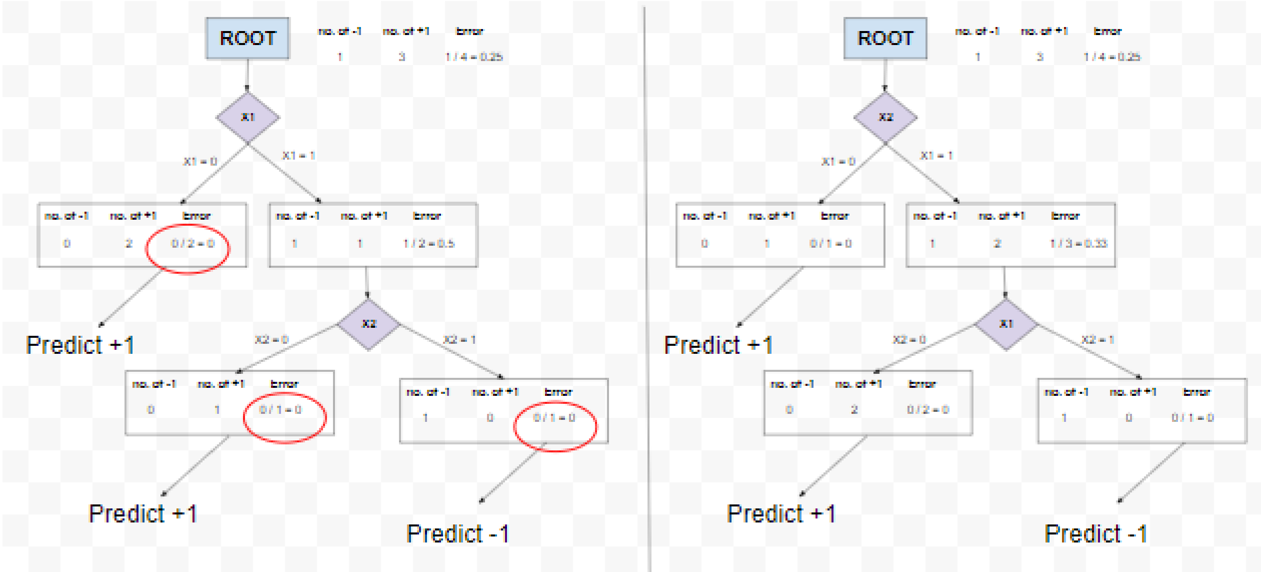
---

11. Refer to the scenario presented in Question 8.

If our parameter for minimum gain in error reduction is 0.1, do we split or stop early?

- ( ) Split
- (•) Stop early

- Stop early : since we can get 0.25 error if we stop after splitting on x1.


ROOT
no. of -1 no. of +1 Error
1 3 1 / 4 = 0.25
X1
X1 = 0
X1 = 1
no. of -1 no. of +1 Error
0 2 0 / 2 = 0
no. of -1 no. of +1 Error
1 1 1 / 2 = 0.5
Predict +1
X2
X2 = 0
X2 = 1
no. of -1 no. of +1 Error
0 1 0 / 1 = 0
no. of -1 no. of +1 Error
1 0 0 / 1 = 0
Predict +1
Predict -1
ROOT
no. of -1 no. of +1 Error
1 3 1 / 4 = 0.25
X2
X1 = 0
X1 = 1
no. of -1 no. of +1 Error
0 1 0 / 1 = 0
no. of -1 no. of +1 Error
1 2 1 / 3 = 0.33
Predict +1
X1
X2 = 0
X2 = 1
no. of -1 no. of +1 Error
0 2 0 / 2 = 0
no. of -1 no. of +1 Error
1 0 0 / 1 = 0
Predict +1
Predict -1

In [ ]:

In [ ]: