# Decision Trees

# Decision Tree in a Nutshell

Goal: build a tree of decisions to predict the class of an object

1. Recursively partition the training set with the goal of minimizing classification errors, using the “most” helpful attribute
2. Many methods to choose the attribute for partitioning
   - Maximize information gain (minimize entropy)
   - Minimize gini impurity

Let’s see an example.

# Example: Riding Mowers

- Goal: Classify 24 households as owning or not owning riding mowers
- Attributes = Income, Lot Size

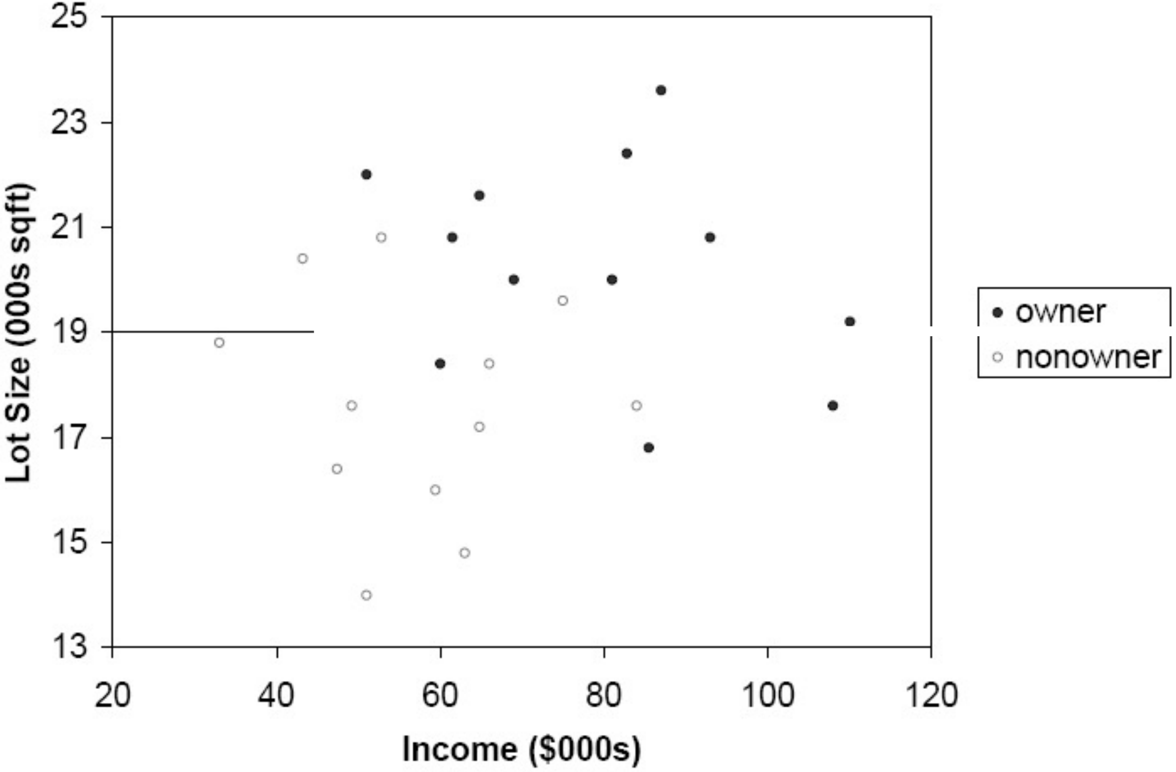
# Building a Tree

- We want to build a tree that tells us the difference between:
  - Owners: those who own a riding mower
  - Non-owners: those who do not

Training set

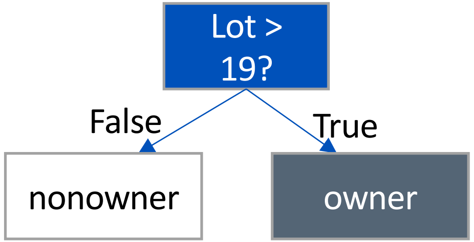
| Income | Lot_Size | Ownership |
|---|---|---|
| 60.0 | 18.4 | owner |
| 85.5 | 16.8 | owner |
| 64.8 | 21.6 | owner |
| 61.5 | 20.8 | owner |
| 87.0 | 23.6 | owner |
| 110.1 | 19.2 | owner |
| 108.0 | 17.6 | owner |
| 82.8 | 22.4 | owner |
| 69.0 | 20.0 | owner |
| 93.0 | 20.8 | owner |
| 51.0 | 22.0 | owner |
| 81.0 | 20.0 | owner |
| 75.0 | 19.6 | non-owner |
| 52.8 | 20.8 | non-owner |
| 64.8 | 17.2 | non-owner |
| 43.2 | 20.4 | non-owner |
| 84.0 | 17.6 | non-owner |
| 49.2 | 17.6 | non-owner |
| 59.4 | 16.0 | non-owner |
| 66.0 | 18.4 | non-owner |
| 47.4 | 16.4 | non-owner |
| 33.0 | 18.8 | non-owner |
| 51.0 | 14.0 | non-owner |
| 63.0 | 14.8 | non-owner |

# Here is the Data set



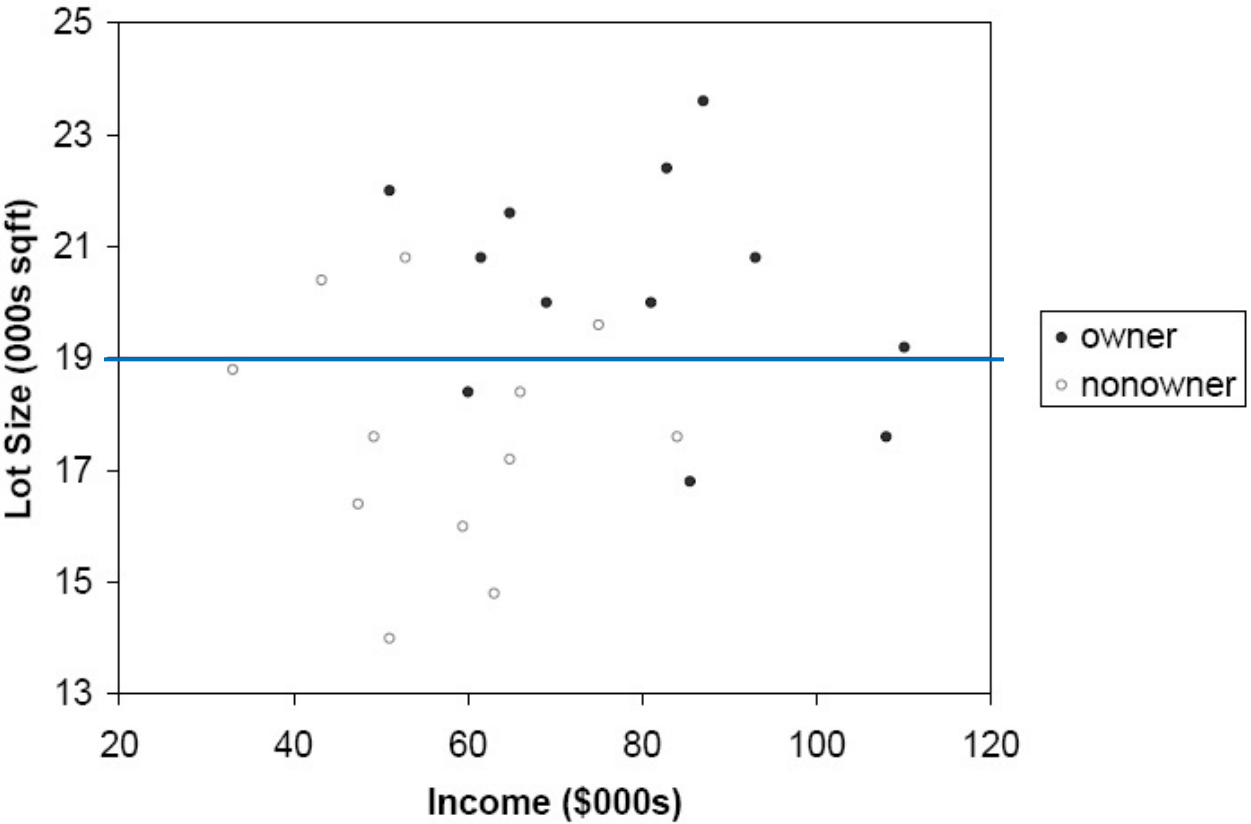
# First Split

Decision tree of depth 1



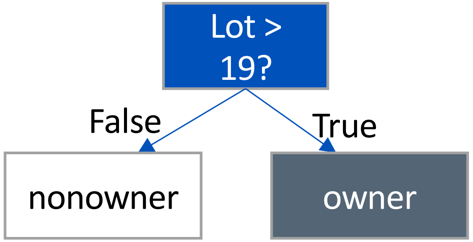

Suppose that the one above is the final tree.

How many classification errors does this tree make on the training set?

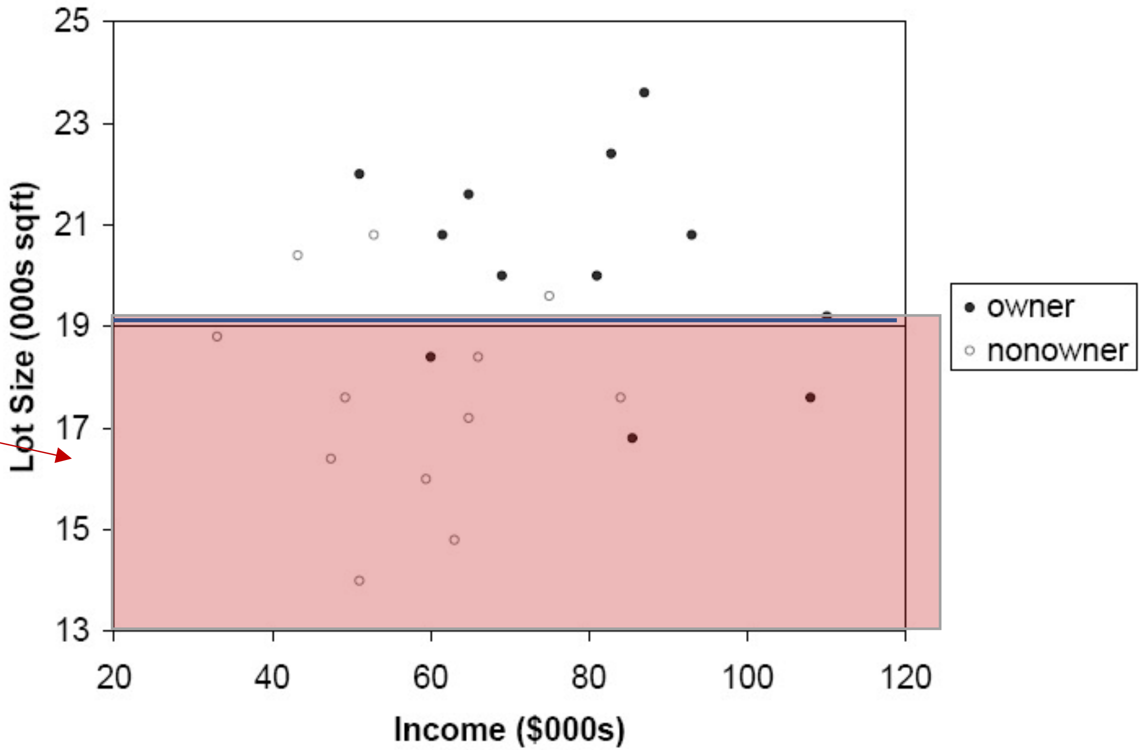


# First Split, Cont'd

Decision tree of depth 1

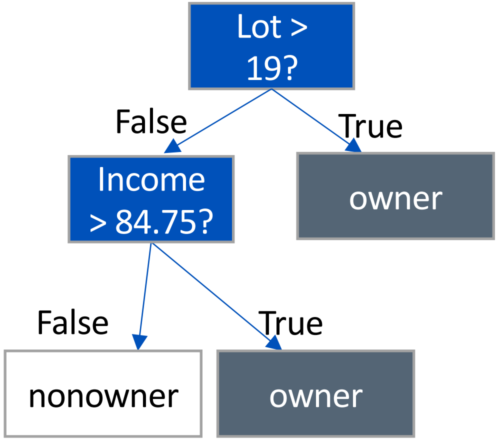



Let's split this node further

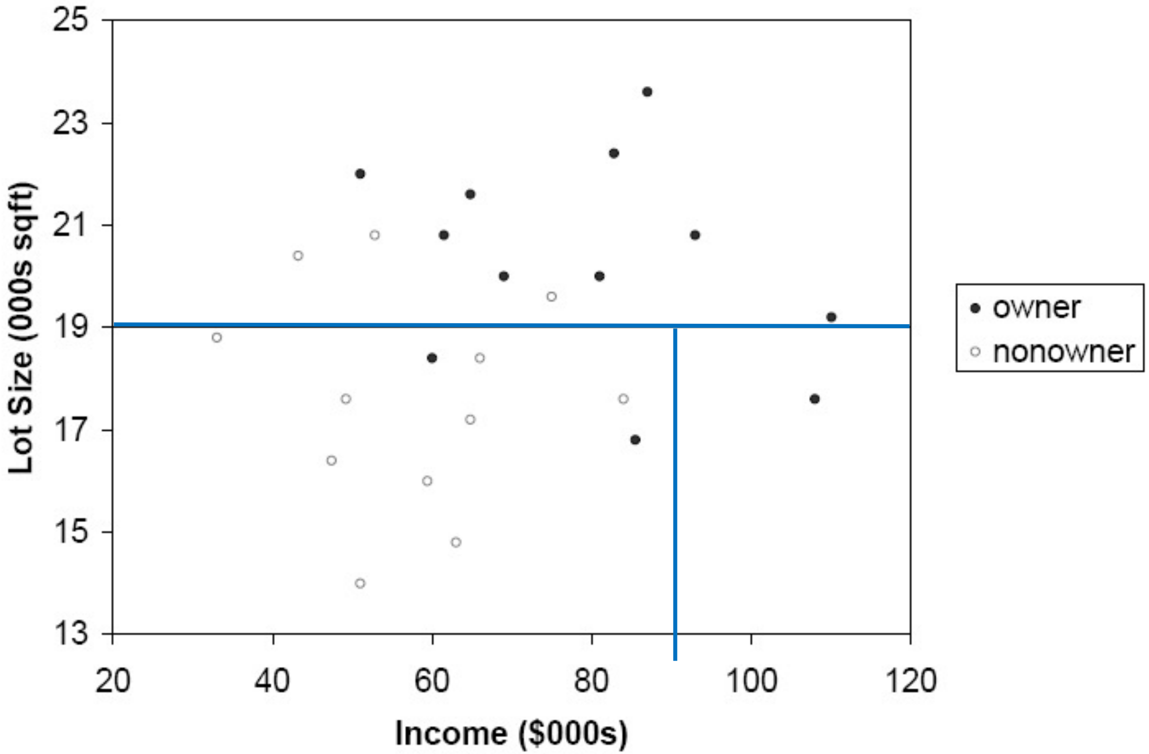


# Second Split: Part I

Decision tree of depth 2

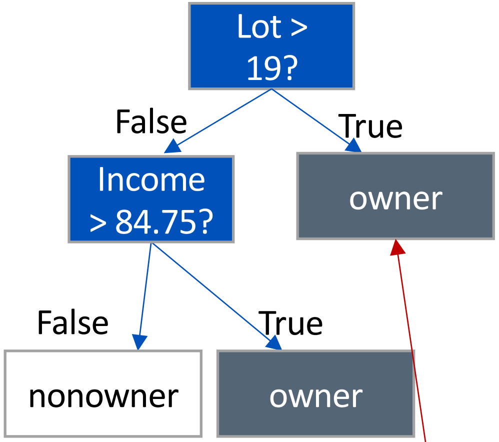



Suppose that the one above is the final tree.
How many classification errors does this tree make on the training set?

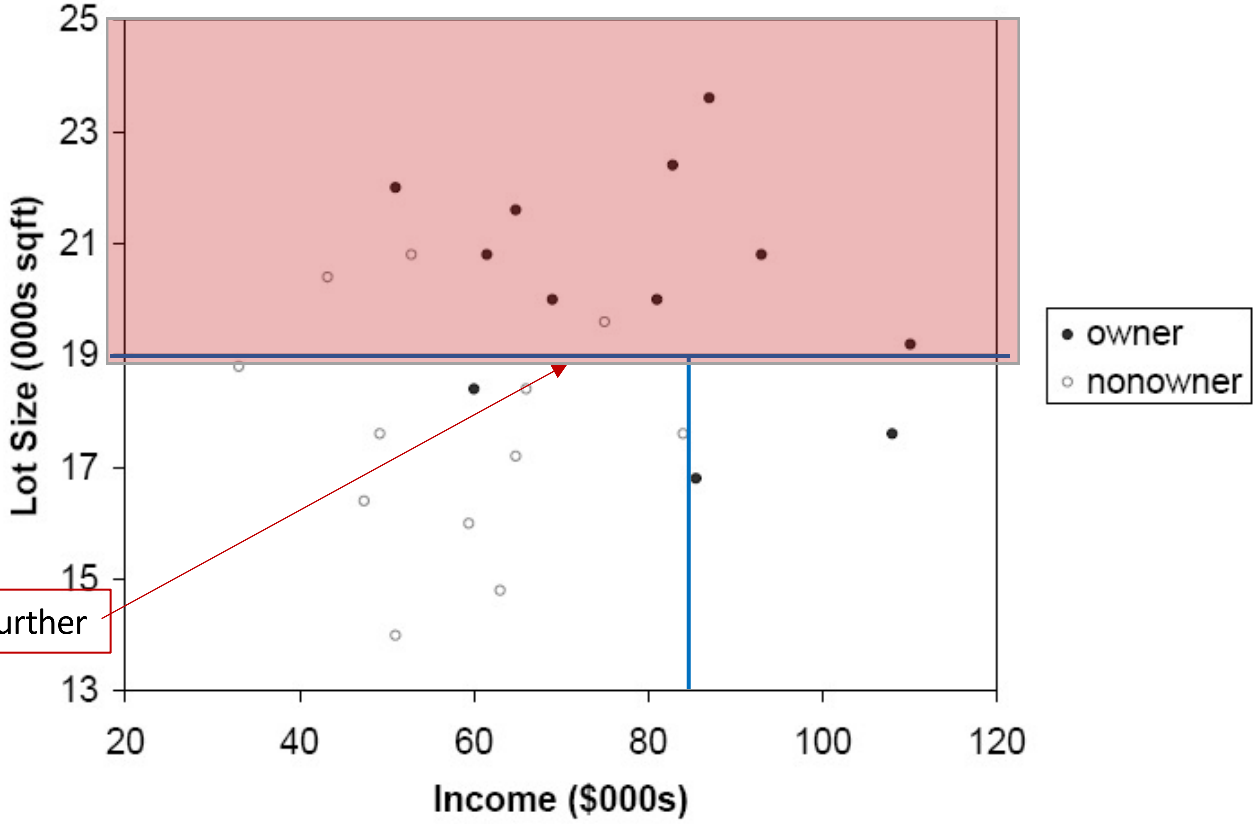


# Second Split: Part II

Decision tree of depth 2

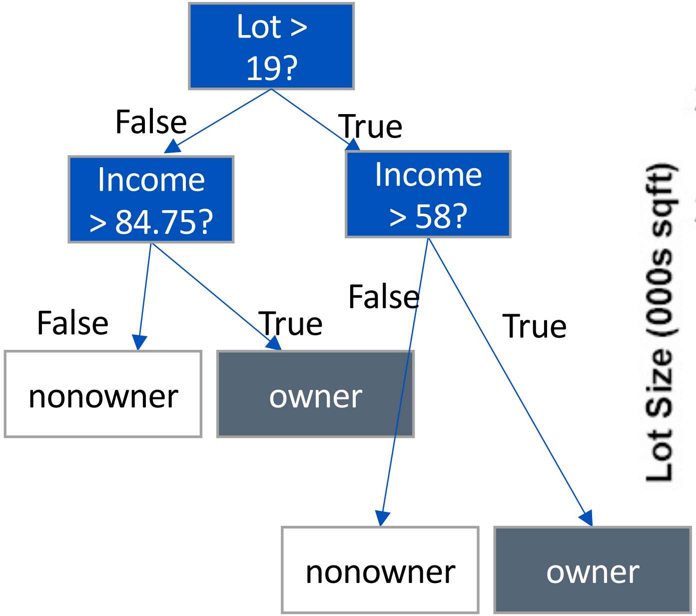


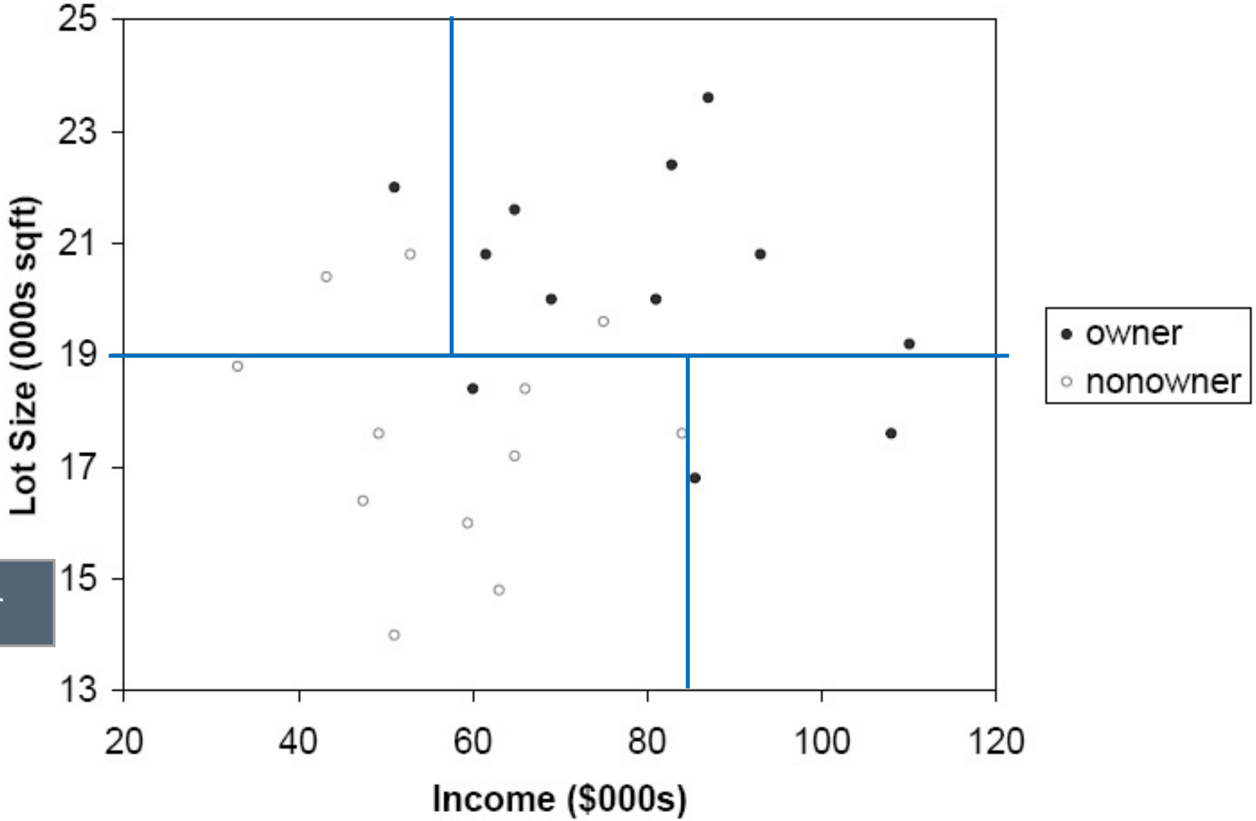

Let’s split this node further



# Second Split: Part III

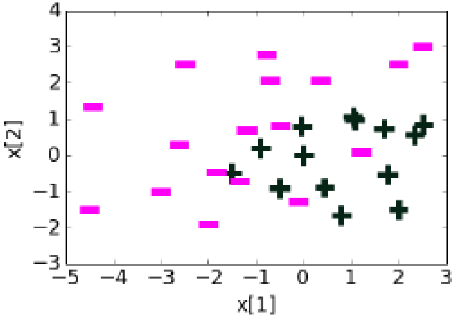
Decision tree of depth 2

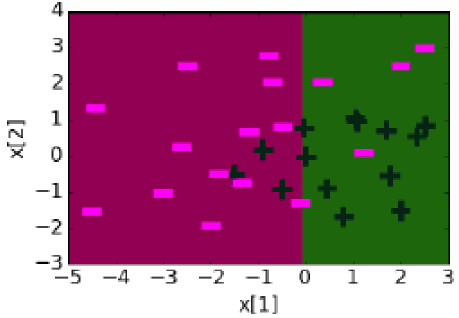


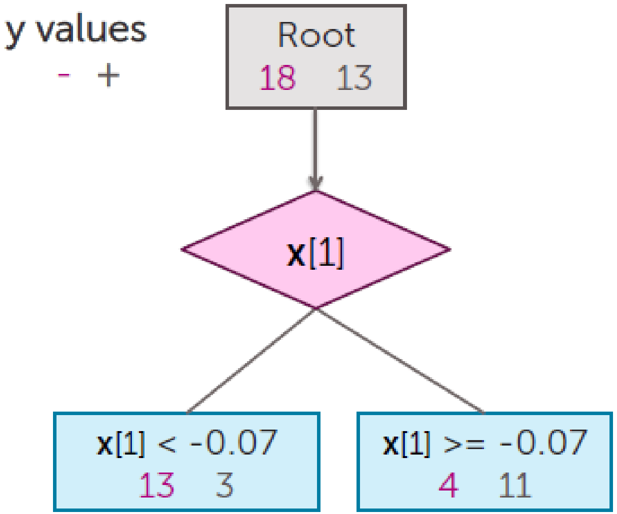




How many classification errors does this tree make on the training set?

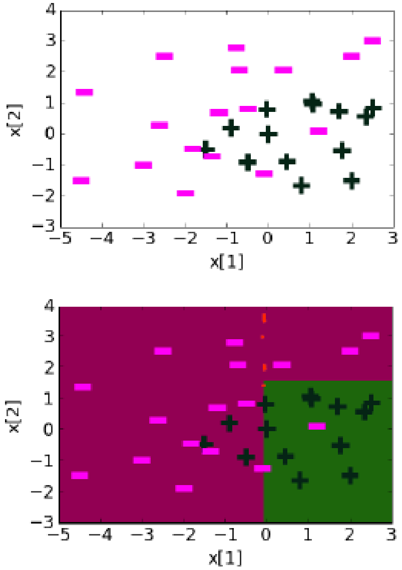
# Decision Boundary With Depth of one (Decision Stump)

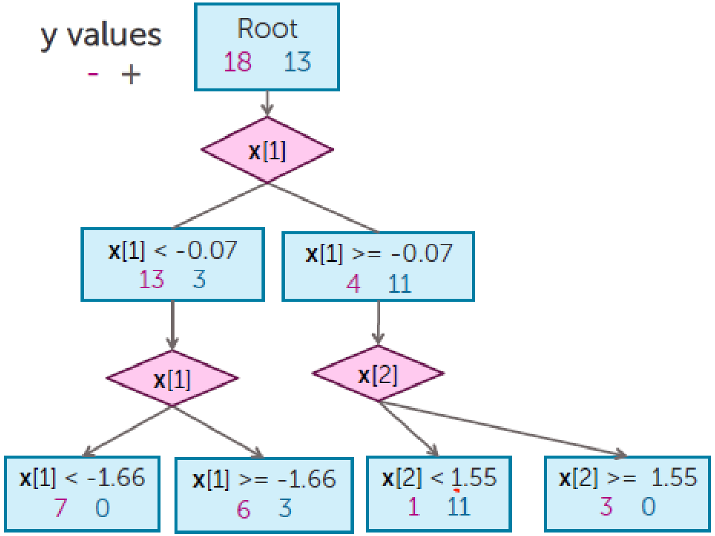


# Decision Boundary with Depth of two

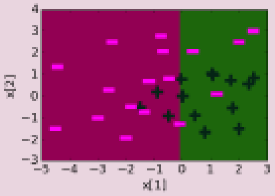
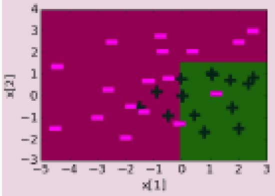
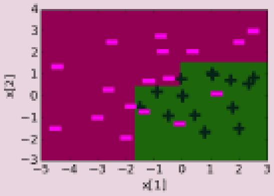
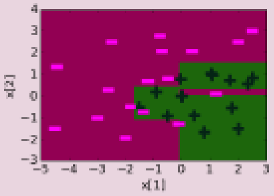
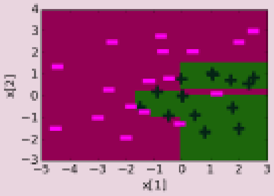


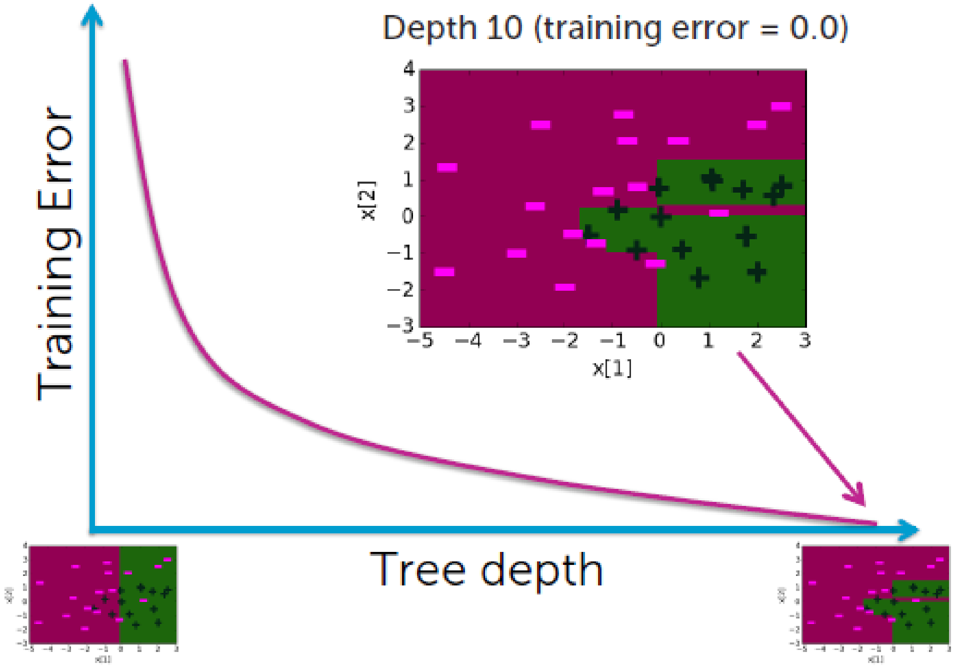
# Decision Boundary Comparison

Why training error reduces with depth?

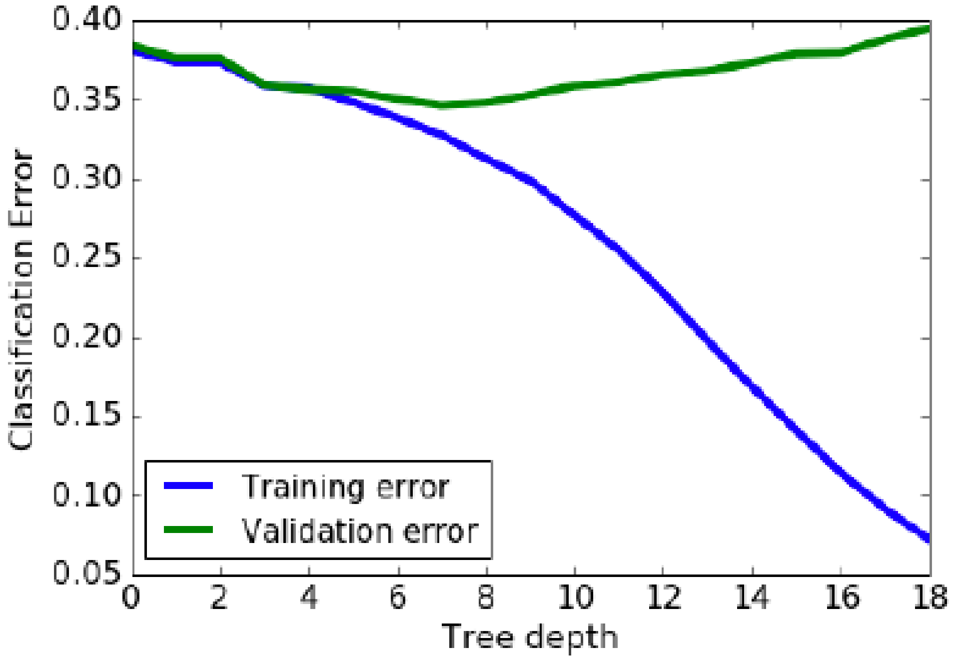
Training error reduces with depth

| Tree depth | depth = 1 | depth = 2 | depth = 3 | depth = 5 | depth = 10 |
|---|---|---|---|---|---|
| Training error | 0.22 | 0.13 | 0.10 | 0.03 | 0.00 |
| Decision boundary |  |  |  |  |  |

# Deeper Trees → Lower Training Error



# Decision Trees Overfitting



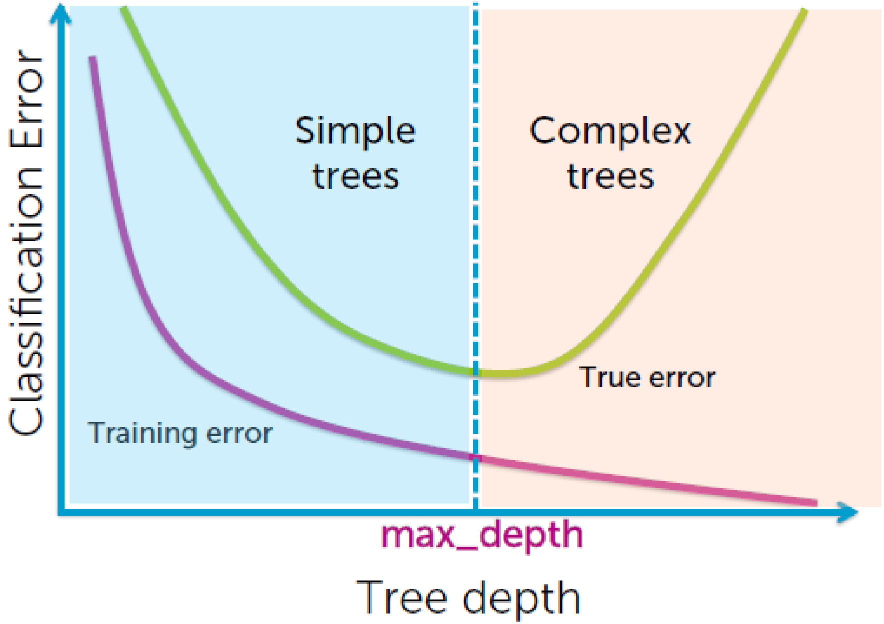
# How do we Pick Simpler Trees?

1. Early Stopping: Stop learning algorithm before tree become too complex (3 conditions)

2. Pruning: Simplify tree after learning algorithm terminates (complements early stopping)

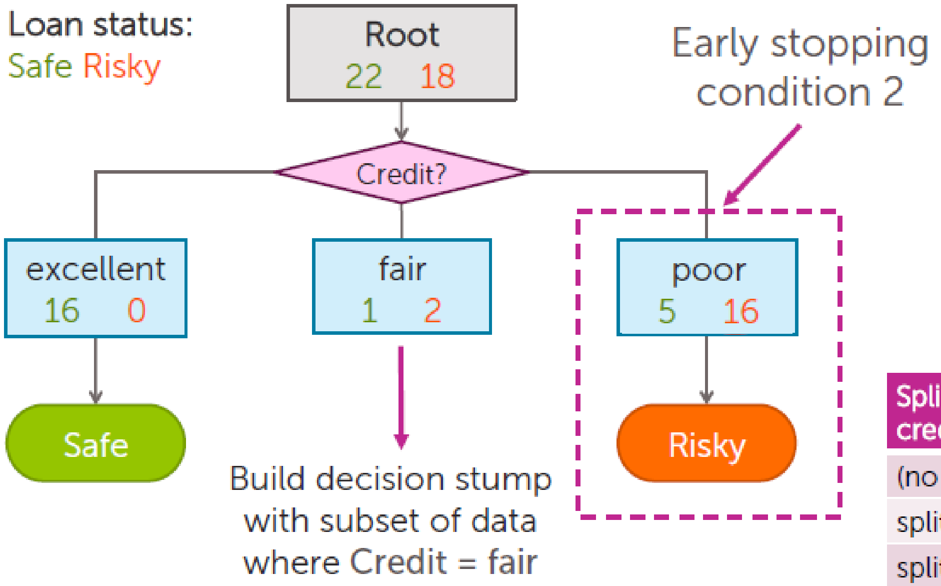
# Early Stopping Condition 1

## Limit the depth of a tree (max_depth)



# Early Stopping Condition 2

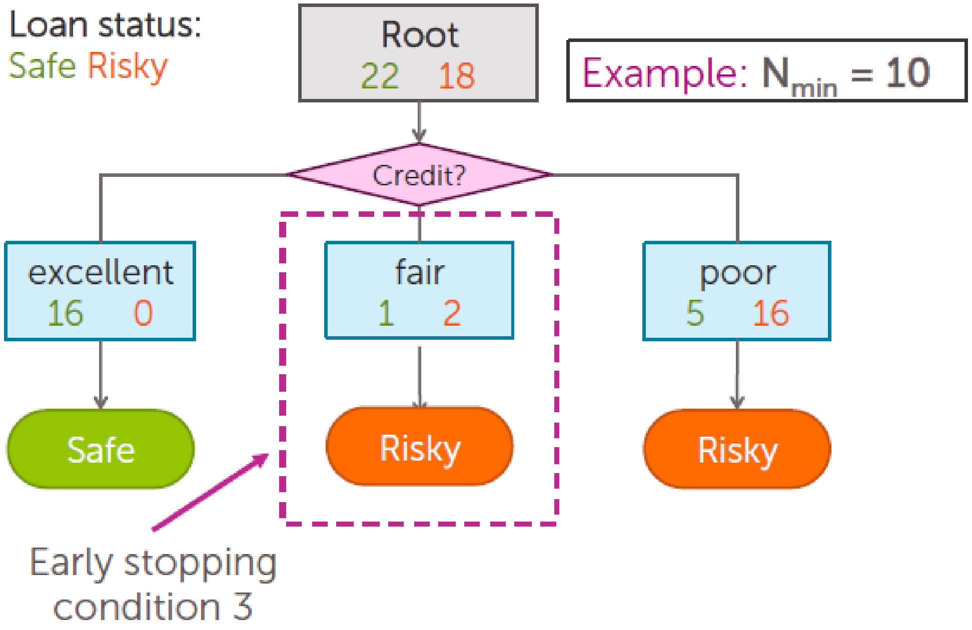
## No split improves classification error (min_impurity_decrease)



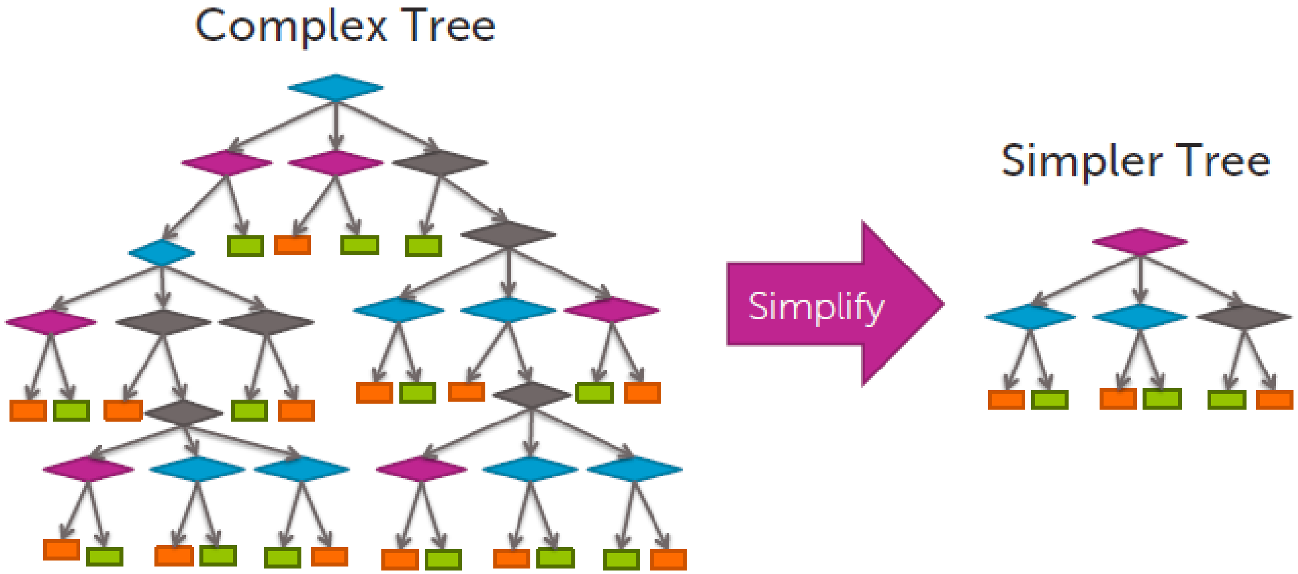

| Splits for credit=poor | Classification error |
|---|---|
| (no split) | 0.24 |
| split on term | 0.24 |
| split on income | 0.24 |

# Early Stopping Condition 3

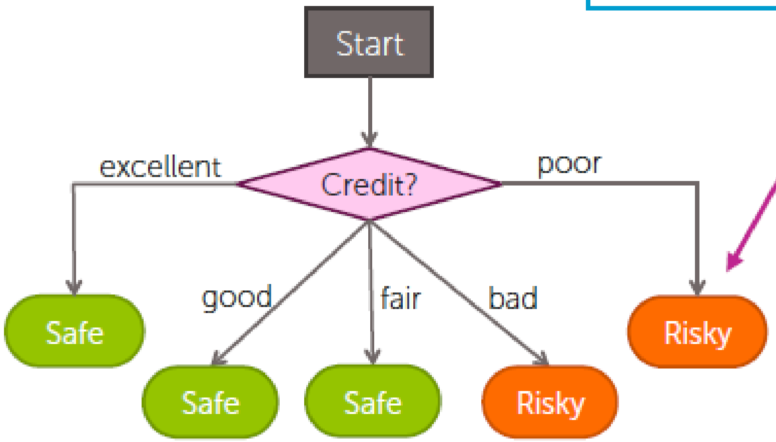
**Stop when data points in a node <= $N_{min}$ (min_samples_split)**



# Pruning: Train a Complex Tree, Simplify Later



# Simple Measure of Complexity of Tree (max_leaf_nodes)




L(T) = # of leaf nodes

# Desired Total Cost Format

Want to balance:

a) How well tree fits data

b) Complexity of tree

Total cost = measure of fit + measure of complexity

Large number indicates bad fit to training data + likely to overfit

# Balancing fit and Complexity (Hyperparameter $\lambda$)

Total cost = measure of fit + measure of complexity

= classification error + number of leaf nodes

= Error(T) + $\lambda$ L(T)

# Decision Trees: Pros and Cons

## Pros

- Easily visualized and interpreted.
- No feature normalization or scaling typically needed.
- Work well with datasets using a mixture of feature types (continuous, categorical, binary).

## Cons

- Even after tuning, decision trees can often still overfit.
- Usually need an ensemble of trees for better generalization performance.

# Optional Materials

# Criteria for Classification

- Gini Index:

$$H_{\text{gini}}(X_m) = \sum_{k \in \mathcal{Y}} p_{mk}(1 - p_{mk})$$

- Cross-Entropy:

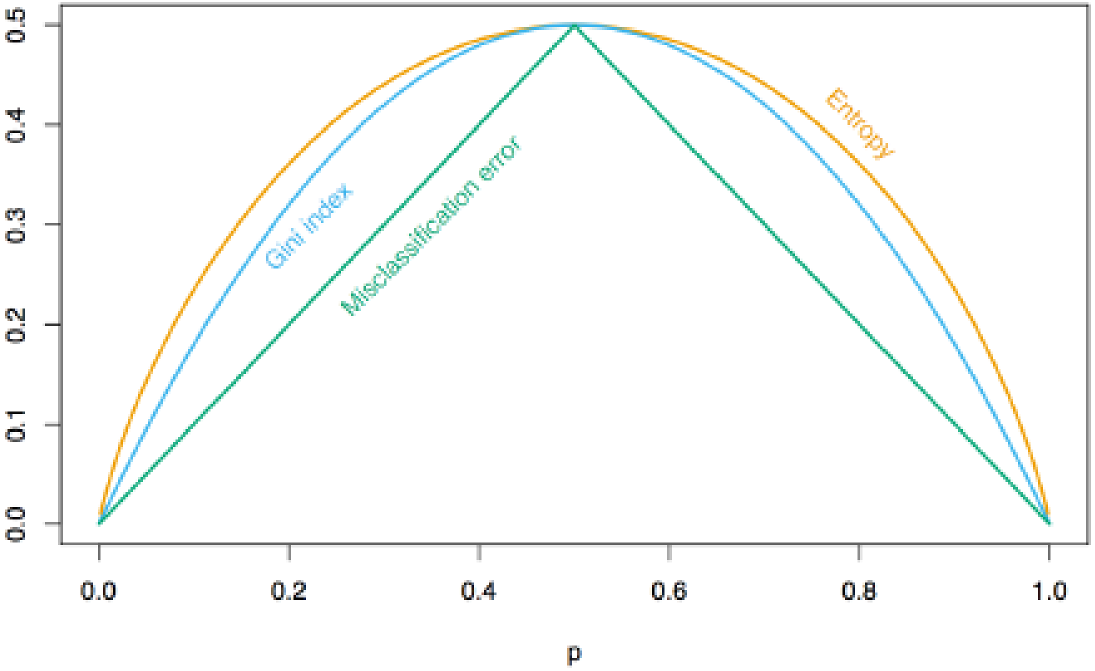
$$H_{\text{CE}}(X_m) = -\sum_{k \in \mathcal{Y}} p_{mk} \log(p_{mk})$$

$X_m$ observations in node m

$\mathcal{Y}$ classes

$p_{m\cdot}$ distribution over classes in node m

# Criteria for Classification, Cont'd



# Criteria for Regression

$$\text{Prediction: } \bar{y}_m = \frac{1}{N_m} \sum_{i \in N_m} y_i$$

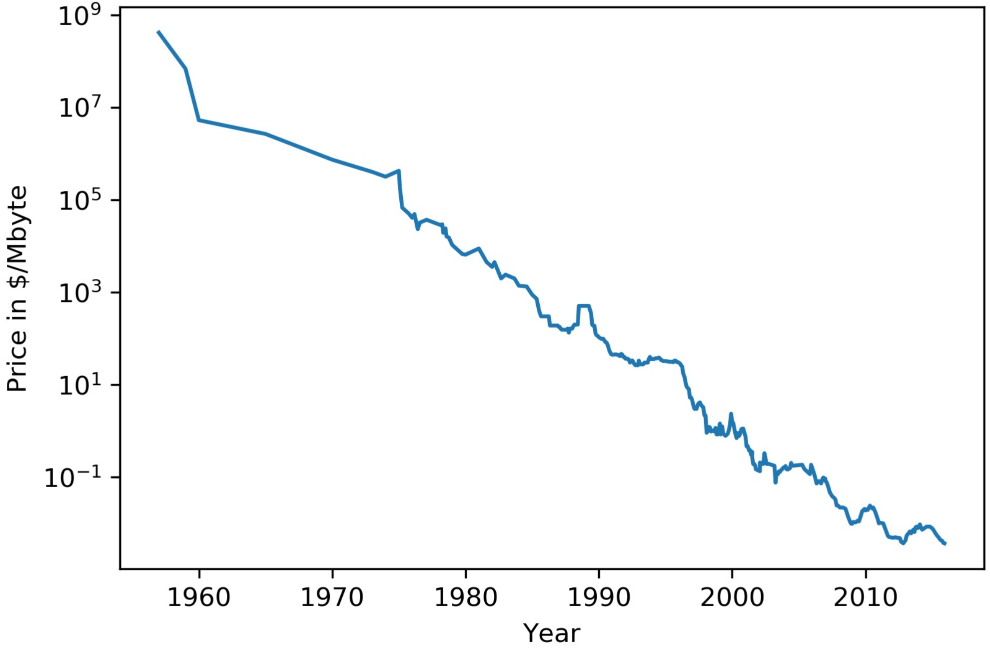
Mean Squared Error:

$$H(X_m) = \frac{1}{N_m} \sum_{i \in N_m} (y_i - \bar{y}_m)^2$$

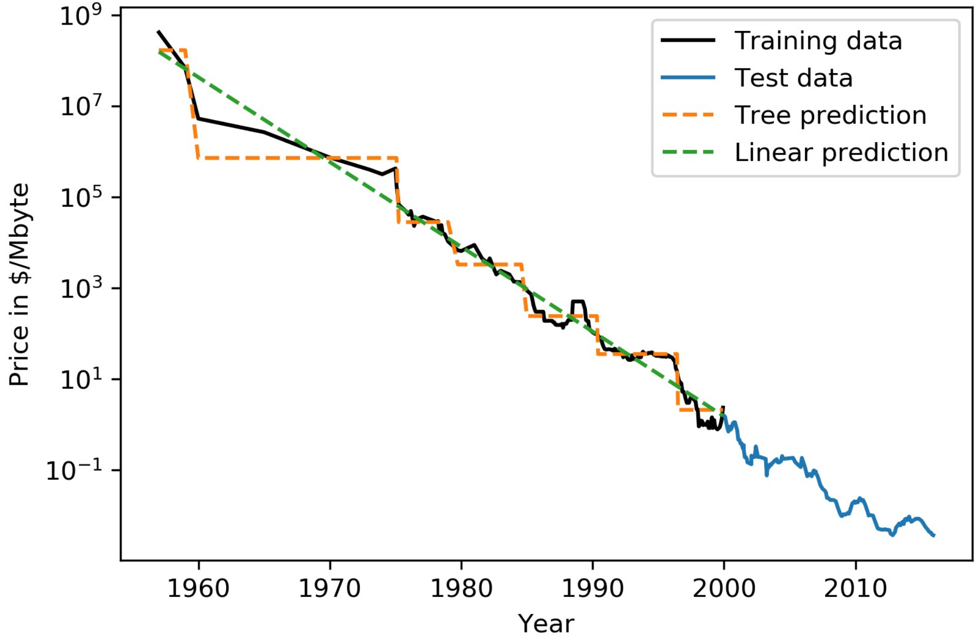
Mean Absolute Error:

$$H(X_m) = \frac{1}{N_m} \sum_{i \in N_m} |y_i - \bar{y}_m|$$

# Extrapolation: Part I



# Extrapolation: Part II



# Extrapolation: Part III