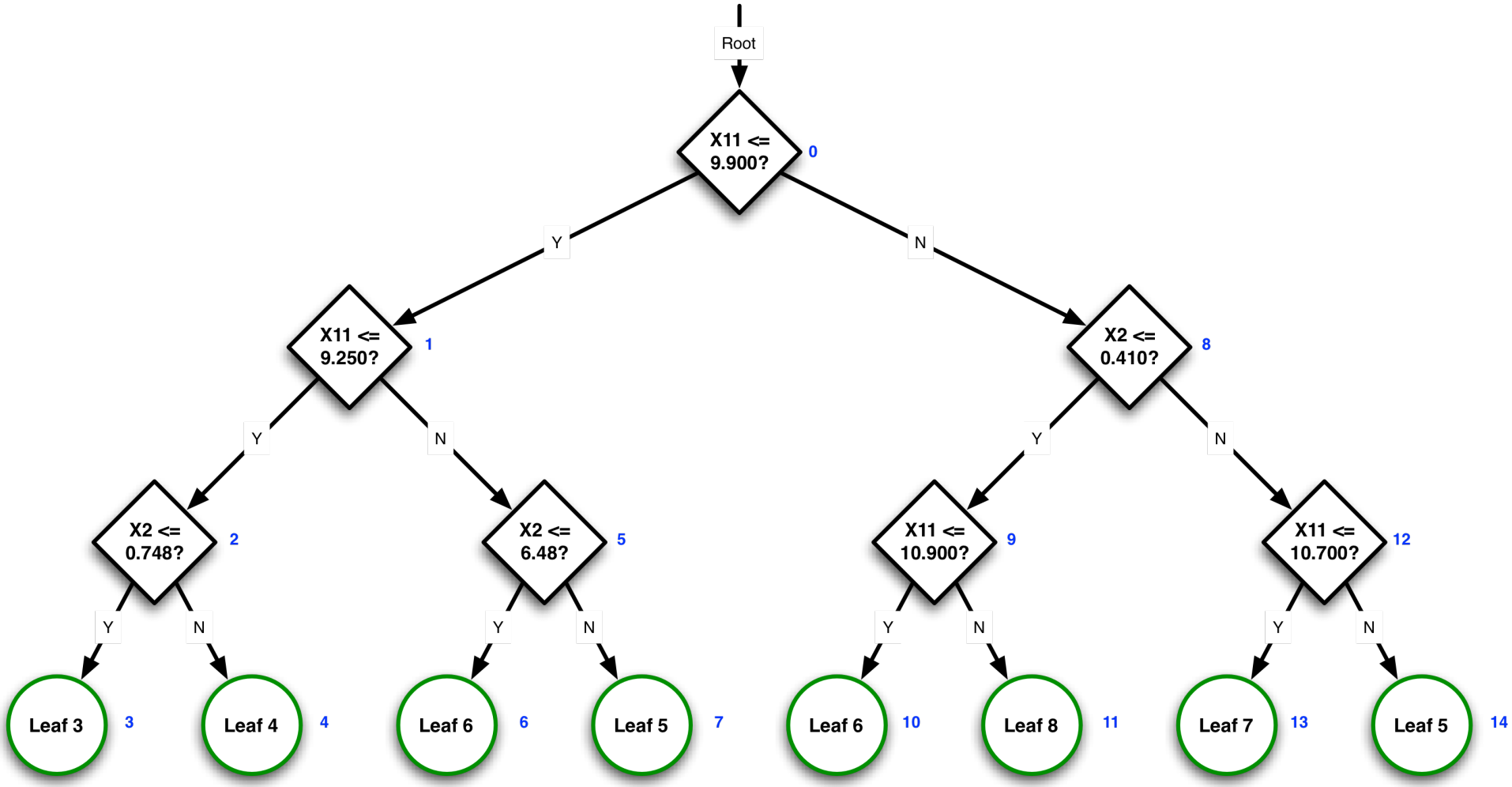
CS 7646: How to build a decision tree from data

Decision Trees

- List of factors or attributes X1… XN
- Labels Y
- Decision nodes
  - Binary decision: Xi <= SplitVal?
- Outgoing edges
- Leaves: Values

- Built from data examples <X1, X2,..., XN, Y>

Decision Tree: Graphical View

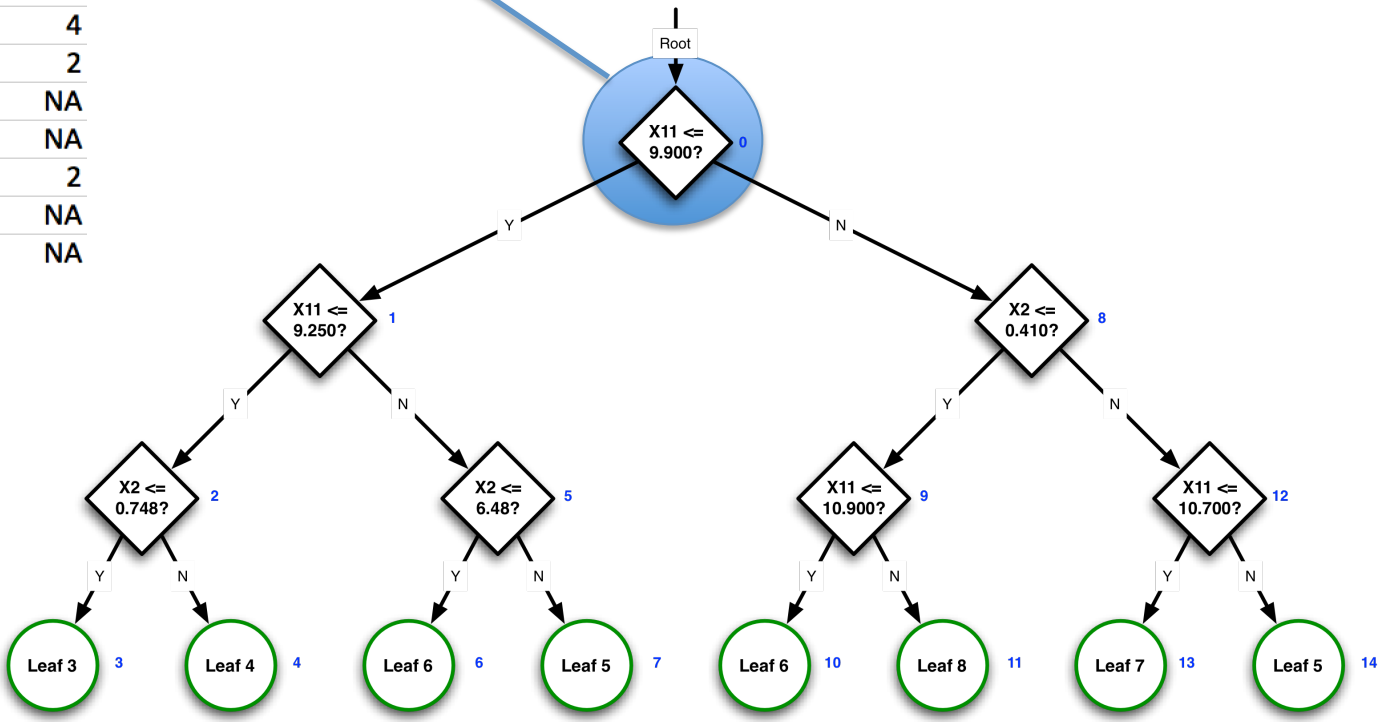


Decision Tree: Tabular View

node	Factor	SplitVal	Left	Right
0	X11	9.900	1	7
1	X11	9.250	1	4
2	X2	0.748	1	2
3	Leaf	3.000	NA	NA
4	Leaf	4.000	NA	NA
5	X2	0.648	1	2
6	Leaf	6.000	NA	NA
7	Leaf	5.000	NA	NA
8	X2	0.410	1	4
9	X11	10.900	1	2
10	Leaf	6.000	NA	NA
11	Leaf	8.000	NA	NA
12	X11	10.700	1	2
13	Leaf	7.000	NA	NA
14	Leaf	5.000	NA	NA

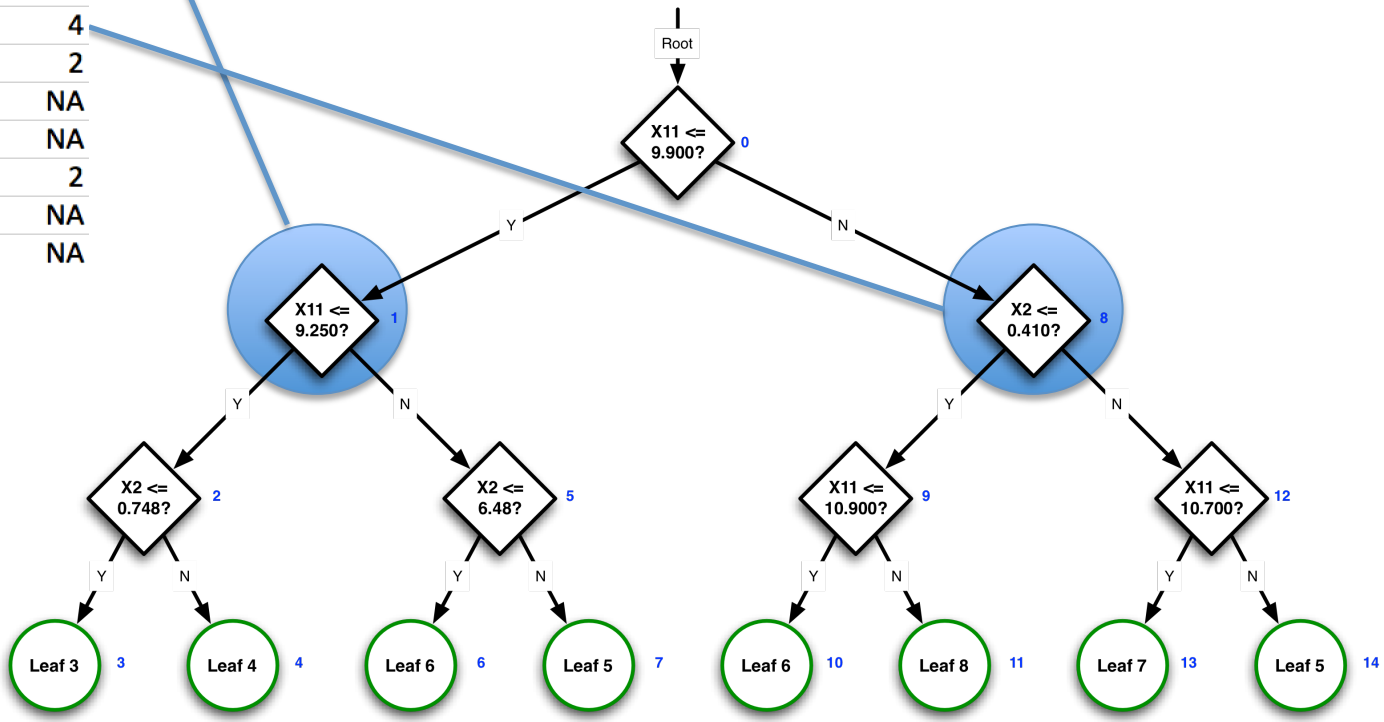
Decision Tree

node	Factor	SplitVal	Left	Right
0	X11	9.900	1	7
1	X11	9.250	1	4
2	X2	0.748	1	2
3	Leaf	3.000	NA	NA
4	Leaf	4.000	NA	NA
5	X2	0.648	1	2
6	Leaf	6.000	NA	NA
7	Leaf	5.000	NA	NA
8	X2	0.410	1	4
9	X11	10.900	1	2
10	Leaf	6.000	NA	NA
11	Leaf	8.000	NA	NA
12	X11	10.700	1	2
13	Leaf	7.000	NA	NA
14	Leaf	5.000	NA	NA



Decision Tree

node	Factor	SplitVal	Left	Right
0	X11	9.900	1	7
1	X11	9.250	1	4
2	X2	0.748	1	2
3	Leaf	3.000	NA	NA
4	Leaf	4.000	NA	NA
5	X2	0.648	1	2
6	Leaf	6.000	NA	NA
7	Leaf	5.000	NA	NA
8	X2	0.410	1	4
9	X11	10.900	1	2
10	Leaf	6.000	NA	NA
11	Leaf	8.000	NA	NA
12	X11	10.700	1	2
13	Leaf	7.000	NA	NA
14	Leaf	5.000	NA	NA



Decision Tree Algorithm (JR Quinlan)

```
build_tree(data)
    if data.shape[0] == 1: return [leaf, data.y, NA, NA]
    if all data.y same: return [leaf, data.y, NA, NA]
    else
        determine best feature i to split on
        SplitVal = data[:,i].median()
        lefttree = build_tree(data[data[:,i]<=SplitVal])
        righttree = build_tree(data[data[:,i]>SplitVal])
        root = [i, SplitVal, 1, lefttree.shape[0] + 1]
        return (append(root, lefttree, righttree))
```

How to determine “best” feature?

Goal: Divide and conquer

Group data into most similar groups.

Approaches:

- Information gain: Entropy
- Information gain: Correlation
- Information gain: Gini Index

Random Tree Algorithm (A Cutler)

```
build_tree(data)
    if data.shape[0] == 1: return [leaf, data.y, NA, NA]
    if all data.y same: return [leaf, data.y, NA, NA]
    else
        determine random feature i to split on
        SplitVal = (data[random,i] + data[random,i]) / 2
        lefttree = build_tree(data[data[:,i]<=SplitVal])
        righttree = build_tree(data[data[:,i]>SplitVal])
        root = [i, SplitVal, 1, lefttree.shape[0] + 1]
        return (append(root, lefttree, righttree))
```

Strengths and weaknesses of decision tree learners

- Cost of learning:
  - Most: Decision Trees
  - Medium: Linear Regression
  - Least: KNN
- Cost of query:
  - Most: KNN
  - Medium: Decision Trees
  - Least: Linear Regression
- Trees: Don’t have to normalize your data and can easily handle missing data