Supervised Learning

- Decision trees
- Artificial neural nets
- K-nearest neighbor
- Support vectors
- Linear regression
- Logistic regression
- ...

Supervised Learning

- \( y = F(x) \): true function (usually not known)
- \( D \): training sample drawn from \( F(x) \)

<table>
<thead>
<tr>
<th>Age</th>
<th>Gender</th>
<th>Height</th>
<th>Weight</th>
<th>Waist</th>
<th>Blood Pressure</th>
<th>Sugar</th>
<th>Heart Rate</th>
<th>Cholesterol</th>
<th>Diabetes</th>
</tr>
</thead>
<tbody>
<tr>
<td>57</td>
<td>M</td>
<td>195</td>
<td>0</td>
<td>125</td>
<td>95</td>
<td>39,25</td>
<td>0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>M</td>
<td>160</td>
<td>1,130</td>
<td>100</td>
<td>37,40</td>
<td>1,0,0,0,0,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>69</td>
<td>F</td>
<td>180</td>
<td>0</td>
<td>115</td>
<td>85</td>
<td>40,22</td>
<td>0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>M</td>
<td>165</td>
<td>0,110</td>
<td>80</td>
<td>41</td>
<td>30,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>F</td>
<td>135</td>
<td>0,115</td>
<td>95</td>
<td>39</td>
<td>35,40</td>
<td>0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>84</td>
<td>F</td>
<td>210</td>
<td>1,135</td>
<td>105</td>
<td>39</td>
<td>24,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>89</td>
<td>F</td>
<td>135</td>
<td>0,120</td>
<td>95</td>
<td>36</td>
<td>36,40</td>
<td>0,0,0,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>M</td>
<td>195</td>
<td>0,115</td>
<td>85</td>
<td>39</td>
<td>32,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>M</td>
<td>205</td>
<td>0,115</td>
<td>90</td>
<td>37</td>
<td>18,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>M</td>
<td>250</td>
<td>1,130</td>
<td>100</td>
<td>38</td>
<td>26,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>F</td>
<td>140</td>
<td>0,125</td>
<td>100</td>
<td>40</td>
<td>30,1,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

...
### Supervised Learning

#### Train Set:

```plaintext
<table>
<thead>
<tr>
<th>ID</th>
<th>Sex</th>
<th>Age</th>
<th>Height</th>
<th>BMI</th>
<th>Waist</th>
<th>Glucose</th>
<th>BloodBP</th>
<th>Fibrinogen</th>
<th>DirectCholesterol</th>
<th>HDLCholesterol</th>
<th>LDLCholesterol</th>
<th>Triglycerides</th>
<th>Insulin</th>
<th>A1c</th>
<th>Albumin</th>
<th>Creatinine</th>
<th>ApolipoproteinA1</th>
<th>ApolipoproteinB</th>
<th>Smoking</th>
<th>Alcohol</th>
<th>HeartDisease</th>
</tr>
</thead>
<tbody>
<tr>
<td>57</td>
<td>M</td>
<td>195</td>
<td>125</td>
<td>95</td>
<td>39</td>
<td>25</td>
<td>0</td>
<td>1</td>
<td>0,00000000000110000000000000000000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>78</td>
<td>M</td>
<td>160</td>
<td>1,130</td>
<td>100</td>
<td>37</td>
<td>40</td>
<td>1</td>
<td>0</td>
<td>0,00000000000110000000000000000000</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>69</td>
<td>F</td>
<td>180</td>
<td>0</td>
<td>115</td>
<td>85</td>
<td>40</td>
<td>22</td>
<td>0</td>
<td>0,00000000000100000000000000000000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>M</td>
<td>165</td>
<td>0</td>
<td>110</td>
<td>80</td>
<td>41</td>
<td>30</td>
<td>0</td>
<td>0,00000000000000000000000000000000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>54</td>
<td>F</td>
<td>135</td>
<td>0</td>
<td>115</td>
<td>95</td>
<td>39</td>
<td>35</td>
<td>1</td>
<td>0,00000000000100000000000000000000</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>84</td>
<td>F</td>
<td>210</td>
<td>1</td>
<td>135</td>
<td>105</td>
<td>39</td>
<td>24</td>
<td>0</td>
<td>0,00000000000000000000000000000000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>89</td>
<td>F</td>
<td>135</td>
<td>0</td>
<td>120</td>
<td>95</td>
<td>36</td>
<td>28</td>
<td>0</td>
<td>0,00000000000000000000000000000000</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>49</td>
<td>M</td>
<td>195</td>
<td>0</td>
<td>115</td>
<td>85</td>
<td>39</td>
<td>32</td>
<td>0</td>
<td>0,00000000000100000000000000000000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>M</td>
<td>205</td>
<td>0</td>
<td>115</td>
<td>90</td>
<td>37</td>
<td>18</td>
<td>0</td>
<td>0,00000000000000000000000000000000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>74</td>
<td>M</td>
<td>250</td>
<td>1</td>
<td>130</td>
<td>100</td>
<td>38</td>
<td>26</td>
<td>1</td>
<td>0,00000000000100000000000000000000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>77</td>
<td>F</td>
<td>140</td>
<td>0</td>
<td>125</td>
<td>100</td>
<td>40</td>
<td>30</td>
<td>1</td>
<td>0,00000000000000000000000000000000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
```

#### Test Set:

```plaintext
<table>
<thead>
<tr>
<th>ID</th>
<th>Sex</th>
<th>Age</th>
<th>Height</th>
<th>BMI</th>
<th>Waist</th>
<th>Glucose</th>
<th>BloodBP</th>
<th>Fibrinogen</th>
<th>DirectCholesterol</th>
<th>HDLCholesterol</th>
<th>LDLCholesterol</th>
<th>Triglycerides</th>
<th>Insulin</th>
<th>A1c</th>
<th>Albumin</th>
<th>Creatinine</th>
<th>ApolipoproteinA1</th>
<th>ApolipoproteinB</th>
<th>Smoking</th>
<th>Alcohol</th>
<th>HeartDisease</th>
</tr>
</thead>
<tbody>
<tr>
<td>71</td>
<td>M</td>
<td>160</td>
<td>1,130</td>
<td>105</td>
<td>38</td>
<td>20</td>
<td>1</td>
<td>0</td>
<td>0,00000000000000000000000000000000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```

…
Supervised Learning

- **F(x):** true function (usually not known)
- **D:** training sample drawn from F(x)

<table>
<thead>
<tr>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
<th>BMI</th>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
<th>BMI</th>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
<th>BMI</th>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
<th>BMI</th>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
<th>BMI</th>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
<th>BMI</th>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
<th>BMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>57</td>
<td>M</td>
<td>195</td>
<td>95</td>
<td>39</td>
<td>25</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>78</td>
<td>M</td>
<td>160</td>
<td>1</td>
<td>130</td>
<td>100</td>
<td>37</td>
<td>40</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>69</td>
<td>F</td>
<td>180</td>
<td>0</td>
<td>115</td>
<td>85</td>
<td>40</td>
<td>22</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>M</td>
<td>165</td>
<td>0</td>
<td>110</td>
<td>80</td>
<td>41</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>54</td>
<td>F</td>
<td>135</td>
<td>0</td>
<td>115</td>
<td>95</td>
<td>39</td>
<td>35</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- **G(x):** model learned from training sample D

<table>
<thead>
<tr>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
<th>BMI</th>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
<th>BMI</th>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
<th>BMI</th>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
<th>BMI</th>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
<th>BMI</th>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
<th>BMI</th>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
<th>BMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>71</td>
<td>M</td>
<td>160</td>
<td>1</td>
<td>130</td>
<td>105</td>
<td>38</td>
<td>20</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Goal:** \( E<(F(x)-G(x))^2> \) is small (near zero) for future test samples drawn from F(x)
Decision Trees
A Simple Decision Tree

Outlook

- Sunny
  - Humidity
    - High
      - No
    - Normal
      - Yes

- Overcast
  - Yes

- Rain
  - Wind
    - Strong
      - No
    - Weak
      - Yes
Representation

internal node = attribute test
branch = attribute value
leaf node = classification

A Real Decision Tree
A Real Decision Tree

Decision Tree Trained on 1000 Patients:

+833+167 (tree) 0.8327 0.1673 0
  fetal_presentation = 1: +822+116 (tree) 0.8759 0.1241 0
  | previous_csection = 0: +767+81 (tree) 0.904 0.096 0
  | | primiparous = 0: +399+13 (tree) 0.9673 0.03269 0
  | | primiparous = 1: +368+68 (tree) 0.8432 0.1568 0
  | | | fetal_distress = 0: +334+47 (tree) 0.8757 0.1243 0
  | | | birth_weight < 3349: +201+10.555 (tree) 0.9482 0.05176 0
  | | | birth_weight >= 3349: +133+36.445 (tree) 0.783 0.217 0
  | | | fetal_distress = 1: +34+21 (tree) 0.6161 0.3839 0
  | previous_csection = 1: +55+35 (tree) 0.6099 0.3901 0
fetal_presentation = 2: +3+29 (tree) 0.1061 0.8939 1
fetal_presentation = 3: +8+22 (tree) 0.2742 0.7258 1
Real Data: C-Section Prediction

Demo summary:

- Fast
- Reasonably intelligible
- Larger training sample $\Rightarrow$ larger tree
- Different training sample $\Rightarrow$ different tree

collaboration with Magee Hospital, Siemens Research, Tom Mitchell
Search Space

- all possible sequences of all possible tests
- very large search space, e.g., if N binary attributes:
  - 1 null tree
  - N trees with 1 (root) test
  - N*(N-1) trees with 2 tests
  - N*(N-1)*(N-1) trees with 3 tests
  - ≈ N^4 trees with 4 tests
  - maximum depth is N
- size of search space is exponential in number of attributes
  - too big to search exhaustively
  - exhaustive search probably would overfit data (too many models)
  - so what do we do instead?
Top-Down Induction of Decision Trees

• TDIDT
• a.k.a. Recursive Partitioning
  – find “best” attribute test to install at root
  – split data on root test
  – find “best” attribute tests to install at each new node
  – split data on new tests
  – repeat until:
    • all nodes are pure
    • all nodes contain fewer than k cases
    • distributions at nodes indistinguishable from chance
    • tree reaches predetermined max depth
    • no more attributes to test
Find “Best” Split?

Attribute_1?

0

50+,75-

1

40+,15-

left

10+,60-

right

Attribute_2?

0

50+,75-

1

25+,15-

left

25+,60-

right

0.6234

0.4412
Splitting Rules

- Information Gain = reduction in entropy due to splitting on an attribute
- Entropy = expected number of bits needed to encode the class of a randomly drawn + or – example using the optimal info-theory coding

\[ \text{Entropy} = p_+ \log_2 p_+ + p_- \log_2 p_- \]

\[ \text{Gain}(S, A) = \text{Entropy}(S) - \frac{|S_v|}{|S|} \text{Entropy}(S_v) \]
Entropy
Splitting Rules

- Problem with Node Purity and Information Gain:
  - prefer attributes with many values
  - extreme cases:
    - Social Security Numbers
    - patient ID’s
    - integer/nominal attributes with many values (JulianDay)
Splitting Rules

GainRatio\((S, A)\) = \[
\frac{\text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)}{\sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \log_2 \frac{|S_v|}{|S|}}
\]

\text{Entropy}(S) = \sum_{i} p_i \log_2 \frac{1}{p_i}

\text{Entropy}(S_v) = \sum_{i} p_i \log_2 \frac{1}{p_i}

S_v = \{ x \in S \mid a(x) = v \}

p_i = \frac{\text{count}(x_i)}{|S_v|}

|S| = \text{size of } S

|S_v| = \text{size of } S_v
Gain Ratio Correction Factor

Gain Ratio for Equal Sized n-Way Splits

Number of Splits

Correction Factor
Splitting Rules

- GINI Index
  - Measure of node impurity

\[
\text{GINI}_{node}(\text{Node}) = 1 - \sum_{c \text{ classes}} [p_c]^2
\]

\[
\text{GINI}_{split}(A) = \sum_{v \text{ Values(A)}} \frac{|S_v|}{|S|} \text{GINI}(N_v)
\]
Experiment

- Randomly select # of cases: 2-1000
- Randomly select fraction of +'s and -'s
- Randomly select attribute arity: 2-1000
- Randomly assign cases to branches!!!!!!
- Compute IG, GR, GINI

741 cases: 309+, 432-
Info_Gain

Good Splits

Poor Splits
Gain_Ratio

Good Splits

Poor Splits

Attribute Arity (Number of Attribute Values)
GINI Score

Good Splits

Poor Splits

"ns.ig.gr.gi" using 1:4

Attribute Arity (Number of Attribute Values)
Info_Gain vs. Gain_Ratio

```
"ns.ig.gr.gl" using 312
```

![Graph showing the relationship between Info_Gain and Gain_Ratio](image)
GINI Score vs. Gain_Ratio
Overfitting
Pre-Pruning (Early Stopping)

- Evaluate splits before installing them:
  - don’t install splits that don’t look worthwhile
  - when no worthwhile splits to install, done

- Seems right, but:
  - hard to properly evaluate split without seeing what splits would follow it (use lookahead?)
  - some attributes useful only in combination with other attributes
  - suppose no single split looks good at root node?
Post-Pruning

- Grow decision tree to full depth (no pre-pruning)
- Prune-back full tree by eliminating splits that do not appear to be warranted statistically
- Use train set, or an independent prune/test set, to evaluate splits
- Stop pruning when remaining splits all appear to be warranted
- Alternate approach: convert to rules, then prune rules
Greedy vs. Optimal

- **Optimal**
  - Maximum expected accuracy (test set)
  - Minimum size tree
  - Minimum depth tree
  - Fewest attributes tested
  - Easiest to understand

- Test order not always important for accuracy
- Sometimes random splits perform well
Decision Tree Predictions

- Classification
- Simple probability
- Smoothed probability
- Probability with threshold(s)
Performance Measures

• Accuracy
  – High accuracy doesn’t mean good performance
  – Accuracy can be misleading
  – What threshold to use for accuracy?
• Root-Mean-Squared-Error
  \[ \text{RMSE} = \sqrt{\frac{1}{\#\text{test}} \sum_{i=1}^{\#\text{test}} (1 - \text{Pred}_i\text{Prob}_i(\text{True}_i\text{Class}_i))^2} \]
• Other measures: ROC, Precision/Recall, …
Attribute Types

• Boolean
• Nominal
• Ordinal
• Integer
• Continuous
  – Sort by value, then find best threshold for binary split
  – Cluster into n intervals and do n-way split
Missing Attribute Values

• Some data sets have many missing values
Regression Trees vs. Classification

- Split criterion: minimize RMSE at node
- Tree yields discrete set of predictions

\[ \text{RMSE} = \sum_{i=1}^{\# test} (\text{True}_i - \text{Pred}_i)^2 \]
Converting Decision Trees to Rules

- each path from root to a leaf is a separate rule:

  fetal_presentation = 1: +822+116 (tree) 0.8759 0.1241 0
  |   previous_csection = 0: +767+81 (tree) 0.904 0.096 0
  |   |   primiparous = 1: +368+68 (tree) 0.8432 0.1568 0
  |   |   |   fetal_distress = 0: +334+47 (tree) 0.8757 0.1243 0
  |   |   |   birth_weight < 3349: +201+10.555 (tree) 0.9482 0.05176 0
  fetal_presentation = 2: +3+29 (tree) 0.1061 0.8939 1
  fetal_presentation = 3: +8+22 (tree) 0.2742 0.7258 1

  if (fp=1 & \neg pc & primip & \neg fd & bw<3349) \Rightarrow 0,
  if (fp=2) \Rightarrow 1,
  if (fp=3) \Rightarrow 1.
Advantages of Decision Trees

- TDIDT is relatively fast, even with large data sets ($10^6$) and many attributes ($10^3$)
  - advantage of recursive partitioning: only process all cases at root
- Small-medium size trees usually intelligible
- Can be converted to rules
- TDIDT does feature selection
- TDIDT often yields compact models (Occam’s Razor)
- Decision tree representation is understandable
Decision Trees are Intelligible
Not *ALL* Decision Trees Are Intelligible

Part of Best Performing C-Section Decision Tree
Predicting Probabilities with Trees

• Small Tree
  – few leafs
  – few discrete probabilities

• Large Tree
  – many leafs
  – few cases per leaf
  – few discrete probabilities
  – probability estimates based on small/noisy samples

• What to do?
A Simple Two-Class Problem

From Provost, Domingos pet-mlj 2002
Classification vs. Predicting Probs

From Provost, Domingos pet-mlj 2002
A Harder Two-Class Problem

From Provost, Domingos pet-mlj 2002
Classification vs. Prob Prediction

From Provost, Domingos pet-mlj 2002
PET: Probability Estimation Trees

- **Smooth large trees**
  - correct estimates from small samples at leafs

- **Average many trees**
  - average of many things each with a few discrete values is more continuous
  - averages improve quality of estimates

- **Both**
Laplacian Smoothing

- Small leaf count: 4+, 1–
- Maximum Likelihood Estimate: k/N
  \[ P(+) = \frac{4}{5} = 0.8; \quad P(–) = \frac{1}{5} = 0.2 \]
- Could easily be 3+, 2– or even 2+, 3–, or worse
- Laplacian Correction: \( \frac{k+1}{N+C} \)
  \[ P(+) = \frac{4+1}{5+2} = \frac{5}{7} = 0.7143 \]
  \[ P(–) = \frac{1+1}{5+2} = \frac{2}{7} = 0.2857 \]
  \[ \text{If } N=0, \quad P(+) = P(–) = \frac{1}{2} \]
  \[ \text{Bias towards } P(\text{class}) = \frac{1}{C} \]
Bagging (Model Averaging)

- Train many trees with different random samples
- Average prediction from each tree
Results

Table II. Summary of experimental results: AUC comparisons.

<table>
<thead>
<tr>
<th>Systems</th>
<th>Wins-Ties-Losses</th>
<th>Avg. diff. (%)</th>
<th>Sign test</th>
<th>Wilcoxon test</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4.4 vs. C4.5</td>
<td>18 - 1 - 6</td>
<td>2.0</td>
<td>1.0</td>
<td>0.3</td>
</tr>
<tr>
<td>C4.4 vs. C4.5-L</td>
<td>13 - 3 - 9</td>
<td>0.2</td>
<td>30.0</td>
<td>30.0</td>
</tr>
<tr>
<td>C4.5-L vs. C4.5</td>
<td>21 - 2 - 2</td>
<td>1.7</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>C4.5-B vs. C4.5</td>
<td>24 - 1 - 0</td>
<td>7.3</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>C4.4-B vs. C4.4</td>
<td>23 - 2 - 0</td>
<td>5.3</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>C4.4-B vs. C4.5-B</td>
<td>11 - 5 - 9</td>
<td>-0.1</td>
<td>45.0</td>
<td>50.0</td>
</tr>
</tbody>
</table>

C4.4: no pruning or collapsing
“L”: Laplacian Smoothing
“B”: bagging

From Provost, Domingos pet-mlj 2002
Weaknesses of Decision Trees

- Large or complex trees can be just as unintelligible as other models
- Trees don’t easily represent some basic concepts such as M-of-N, parity, non-axis-aligned classes…
- Don’t handle real-valued parameters as well as Booleans
- If model depends on summing contribution of many different attributes, DTs probably won’t do well
- DTs that look very different can be same/similar
- Usually poor for predicting continuous values (regression)
- Propositional (as opposed to 1st order)
- Recursive partitioning: run out of data fast as descend tree
Popular Decision Tree Packages

- **ID3 (ID4, ID5, …) [Quinlan]**
  - research code with many variations introduced to test new ideas
- **CART: Classification and Regression Trees [Breiman]**
  - best known package to people outside machine learning
  - 1st chapter of CART book is a good introduction to basic issues
- **C4.5 (C5.0) [Quinlan]**
  - most popular package in machine learning community
  - both decision trees and rules
- **IND (INDuce) [Buntine]**
  - decision trees for Bayesians (good at generating probabilities)
  - available from NASA Ames for use in U.S.
When to Use Decision Trees

- Regression doesn’t work
- Model intelligibility is important
- Problem does not depend on many features
  - Modest subset of features contains relevant info
  - not vision
- Speed of learning is important
- Linear combinations of features not critical
- Medium to large training sets
Current Research

- Increasing representational power to include M-of-N splits, non-axis-parallel splits, perceptron-like splits, …
- Handling real-valued attributes better
- Using DTs to explain other models such as neural nets
- Incorporating background knowledge
- TDIDT on really large datasets
  - \( \gg 10^6 \) training cases
  - \( \gg 10^3 \) attributes
- Better feature selection
- Unequal attribute costs
- Decision trees optimized for metrics other than accuracy