A Classification Problem

Is a loan to a person who is 45 years old, divorced, renting an apartment, with two kids and annual income of 100K high risk or low risk?

<table>
<thead>
<tr>
<th>Age</th>
<th>Home Owner</th>
<th>Marital Status</th>
<th># of Kids</th>
<th>Annual Income</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>45</td>
<td>No</td>
<td>Divorced</td>
<td>2</td>
<td>100K</td>
<td>?</td>
</tr>
</tbody>
</table>

Terminology and Concepts ...

Record (or tuple)
- Attributes
  - E.g. age, marital status, # of kids, owns home or not, credit score ...
- Class label
  - E.g. high risk, low risk ...
- Classification: predict the class label with given attribute values

... Terminology and Concepts

Step 1: 
- Training set → Classifier

Step 2: 
- Attribute values → Classifier → Class label

Classifier (or model)
- Training set: records with known class labels that are used to construct (i.e. train) the classifier

Classification vs. Regression

Classification predicts categorical attribute values
Regression predicts continuous numerical attribute values

<table>
<thead>
<tr>
<th>SID</th>
<th>HW1</th>
<th>HW2</th>
<th>HW3</th>
<th>Final</th>
<th>Pass/Fail</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>40</td>
<td>60</td>
<td>70</td>
<td>95</td>
<td>Passed</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>15</td>
<td>11</td>
<td>65</td>
<td>Failed</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>45</td>
<td>40</td>
<td>75</td>
<td>Passed</td>
</tr>
<tr>
<td>4</td>
<td>35</td>
<td>50</td>
<td>35</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

A Training Set

<table>
<thead>
<tr>
<th>TID</th>
<th>Home Owner</th>
<th>Marital Status</th>
<th>Annual Income</th>
<th>Defaulted Borrower</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Single</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Married</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Single</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Married</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Divorced</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Married</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Divorced</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Single</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Married</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Single</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>
A Decision Tree

Decision Tree Induction

Terminating Conditions

Split on Binary Attributes

Split on Nominal Attributes

Split on Ordinal Attributes
Split on Numerical Attributes

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Humidity</th>
<th>Windy</th>
<th>Temperature</th>
<th>Play Golf</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>85</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>90</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>70</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>86</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>88</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>96</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>80</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>65</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>95</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>70</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>80</td>
<td>Yes</td>
</tr>
</tbody>
</table>

... Split on Numerical Attributes

- Binary split
- Pre-discretization
  - Unsupervised
    - Equi-width, equi-depth
  - Supervised
    - Entropy-based with MDL stopping Criterion

Binary Split

- Sort the values
- May split between any two adjacent values
  - May exclude splits that separate two adjacent values with the same class label
- Numerical attributes can be used to split multiple times (unlike categorical attributes)

Splitting Attribute Selection

- After a split, ideally each subset would “pure”, i.e. contains only one class of records

<table>
<thead>
<tr>
<th>Gender</th>
<th>Age</th>
<th>Preferred color</th>
</tr>
</thead>
<tbody>
<tr>
<td>female</td>
<td>20</td>
<td>pink</td>
</tr>
<tr>
<td>male</td>
<td>20</td>
<td>black</td>
</tr>
<tr>
<td>female</td>
<td>15</td>
<td>pink</td>
</tr>
<tr>
<td>male</td>
<td>15</td>
<td>black</td>
</tr>
</tbody>
</table>

Attribute Selection Measures

- Entropy (Information Gain)
- Gini index
- Gain Ratio

Entropy

\[
\text{Entropy}(D) = - \sum_{i=1}^{m} p_i \log_2(p_i)
\]

- \(p_i\) is the fraction of records in \(D\) that belongs to class \(C_i\)
- \(m\) is the number of classes in \(D\)
Entropy Example

- Preferred color
  - 2 black and 2 pink
  - 3 black and 1 pink
  - 4 black

Information Gain

- Suppose \( D \) is split into \( \mathcal{V} \) subsets on attribute \( A \)

\[
Gain(A) = Entropy(D) - \sum_{i=1}^{\mathcal{V}} \frac{|D_i|}{|D|} \times Entropy(D_i)
\]

Information Gain Example

- Preferred color
  - Gain(Gender)
  - Gain(Age)

Split Information

- Information gain favors attributes with lots of distinct values
- Split information can be used to “normalized” information gain

\[
SplitInfo(A) = -\sum_{j=1}^{m} \frac{|D_j|}{|D|} \times \log_2 \left( \frac{|D_j|}{|D|} \right)
\]

Gain Ratio

\[
GainRatio(A) = \frac{Gain(A)}{SplitInfo(A)}
\]

Gini Index

\[
Gini(D) = 1 - \sum_{i=1}^{m} p_i^2
\]

- Used in the CART algorithm for binary split
Gini Index Example

- Preferred color
  - 2 black and 2 pink??
  - 3 black and 1 pink??
  - 4 black??
  - Split on gender??
  - Split on age??

Decision Tree Induction Example

<table>
<thead>
<tr>
<th></th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>L</td>
<td>20</td>
<td>C1</td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>S</td>
<td>9</td>
<td>C2</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>S</td>
<td>11</td>
<td>C2</td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>M</td>
<td>14</td>
<td>C1</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>L</td>
<td>14</td>
<td>C1</td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>S</td>
<td>15</td>
<td>C1</td>
<td></td>
</tr>
</tbody>
</table>

How do we make the first split??

Training Error and Testing Error

- Training error
  - Misclassification of training records
- Testing (Generalization) error
  - Misclassification of testing records

Model Overfitting and Underfitting

Overfitting Due to Outliers/Noise ...

... Overfitting Due to Outliers/Noise
Occam’s Razor

- A.K.A. Principle of Parsimony
- Given two models with the same generalization errors, the simpler model is preferred over the more complex model

Tree Pruning

- Replace a subtree with a leaf node
- The class label of the leaf is the majority class label of the records associated with the subtree

Prepruning

- Prune during decision tree construction
  - Number of records < threshold
  - "Purity gain" < threshold
- Performs poorly in practice

Postpruning

- Bottom-up pruning of a fully constructed tree
  - Replace a subtree with a leaf node if it reduces testing error
    - How do we know whether it reduces testing error or not??
  - Pruning based on Minimum Description Length (MDL)

Estimate Testing Errors ...

- Use a pruning/validation set
  - Usually 1/3 of the original training set
  - Less records for training

... Estimate Testing Errors

- Optimistic error estimation
  - The training set is a good representation of the overall data (optimistic!), so the training error is the testing error
- Pessimistic error estimation
  - Training error + penalty term for model complexity
**Pessimistic Error Estimation**

\[ e_e(T) = \frac{\sum [e(t_i) + \Omega(t_i)]}{\sum n(t_i)} = \frac{e(T) + \Omega(T)}{N} \]

- T – A decision tree
- \( n(t_i) \) – # of training records at leaf node \( t_i \)
- \( e(t_i) \) – # of misclassified training records at \( t_i \)
- \( \Omega(t_i) \) – Penalty term associated with \( t_i \)

**Example of Pessimistic Error Estimation**

\[ e_e(T) \text{ with } \Omega(t_i) = 0.5?? \Omega(t_i) = 1?? \]

**Minimum Description Length (MDL)**

- \( n \) records
- \( m \) binary attributes
- \( k \) classes
- \( \text{Cost(Internal Node)} = \log_m m \)
- \( \text{Cost(Leaf node)} = \log_k k \)
- \( \text{Cost(Error)} = \log_2 n \)
- \( \text{Cost} = \text{Cost(All Nodes)} + \text{Cost(All Errors)} \)

**The best model is the one that minimizes the number of bits to encode both the model and the exceptions to the model**

**MDL Example …**

**… MDL Example**

- 128 records, 8 binary attributes and 3 classes
- Left tree has 7 errors and right tree has 4 errors

**Pruning with Estimated Testing Error**

- Pessimistic error estimation with \( \Omega(t_i) = 1 ?? \)
- MDL ??
About Decision Tree Classification ... 

- Inexpensive to construct
- Extremely fast at classifying unknown records
- Easy to interpret for small-sized trees
- Accuracy is comparable to other classification techniques for many simple data sets

... About Decision Tree Classification

- Data fragmentation
- Tree replication
- Finding an optimal decision tree is NP-hard
- Limitation on expressiveness

Readings

- Textbook Chapter 8.1 and 8.2