CS 3265 and CS 5265
Vanderbilt University

Lecture on Data Mining

• Difference between FD mining and association rule mining
• Unsupervised learning and supervised learning
• Decision trees
A comment on difference between association rules
(https://my.vanderbilt.edu/cs265/data-mining-tasks-1/)
and functional dependencies
(https://my.vanderbilt.edu/cs265/data-mining-programming-assignment/)
Association rules, as presented

\{onions, potatoes\} \rightarrow \{burger\}, \ (e.g., \text{with support 0.15})
where ‘onions’, ‘potatoes’, ‘burger’ are values of binary-valued attributes
(aka variables) that also have values of \sim onions, \sim potatoes, \sim burger

Attribute Burger has possible values burger or \sim burger (or present or absent)
Attribute Onion has possible values onion or \sim onion (or present or absent)
Attribute Potato has possible values potato or \sim potato (or present or absent)

More generally, we can have many-valued attributes, such as
  Color (with values red, green, blue, yellow, \ldots),
  Size (with values xxl, xl, l, m, s, xs)
  Shape (with values triangle, square, circle, squashed-ellipse, \ldots)

Color=red, Shape=circle \rightarrow Size=xl \ (or \text{just red, circle } \rightarrow \text{xl})
Color=blue, Shape=circle \rightarrow Size = s
Size=m \rightarrow Shape=Shape=triangle

\ldots
Functional Dependencies

Example with many-valued attributes, such as
- Color (with values red, green, blue, yellow) 4 values
- Size (with values xxl, xl, l, m, s, xs) 6 values
- Shape (with values triangle, square, circle) 3 values

Color, Shape $\rightarrow$ Size (e.g., with support 0.96)

- Color=red, Shape=circle $\rightarrow$ Size=xl
- Color=blue, Shape=circle $\rightarrow$ Size=s
- Color=green, Shape=circle $\rightarrow$ Size=s
- ... (for all (4*3) x, y pairs)
- Color=green, Shape=triangle $\rightarrow$ Size=m

One could think of FD discovery as including association rule discovery as a subroutine, but the uses of association rule discovery (requiring only very modest Support) and FD discovery (requiring high support) are substantially different.
Association rules are a form of unsupervised data mining (or machine learning)

If a data set is described by attributes A, B, C, … then the result of association rule learning can (potentially help predict the value of any attribute X, from the values of one or more of the other attributes (i.e., pattern completion)

So, given

\[
\text{Knowledge Base}
\]

\[
\begin{array}{c}
\text{blue} \quad ? \quad \text{circle} \\
\downarrow \\
\text{blue small circle}
\end{array}
\]

... Color=blue, Shape=circle $\rightarrow$ Size=s
... Size=m $\rightarrow$ Shape=Triangle
...
Other forms of unsupervised learning:
- Clustering
- Belief network learning

<table>
<thead>
<tr>
<th>Amount of training</th>
<th>Test data accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0.01</td>
</tr>
<tr>
<td>0.01 – 0.12</td>
<td>– 0.45</td>
</tr>
</tbody>
</table>

Incomplete Data

0.5 ? 0.01 -0.12 .... ?

Completed data

0.5 0.75 0.01 –0.12 .... –0.45
Two perspectives of Machine Learning:
Machine Learning for advanced *data analysis*
Machine Learning for robust artificial *agents*

- Performance “quality” (e.g., accuracy)
  - Training or time or …

- Performance “quality” (e.g., error, response time)
  - Training or time or …

Pessimism (be cautious) and optimism (jump to conclusions)
Environment

Labeled data

Learning Component

Classifier

Unlabeled datum such as $v_{11}, v_{21}, v_{32}, \ldots, v_{m2}, C_1$

Performance Component

Supervised learning

Douglas H. Fisher
Each internal node represents a test of a variable, and each leaf represents a decision based on the conditions (variable values) along the path to that leaf.
Decision tree classifiers

[ SciFi = -1, Suspense = 1, Romance = -1, Ebert = 1, Siskel = 1, …, Rent-it???, ...

\[ \text{SciFi} \]

\[ \text{Ebert} \]

\[ \text{Siskel} \]

\[ \text{Rent-it} \]

\[ \text{BigStar} \]

\[ \text{B&W} \]

\[ \text{Romance} \]

\[ \text{Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]

\[ \text{~Rent-it} \]
Decision tree classifiers

\[ \text{SciFi} = -1, \text{Suspense} = 1, \text{Romance} = -1, \text{Ebert} = 1, \text{Siskel} = 1, \ldots, \text{Rent-it} \]
Decision tree classifiers

[SciFi = -1, Suspense = 1, Romance = -1, Ebert = 1, Siskel = 1, ..., Rent-it??]

SciFi

-1

~Rent-it

-1

Rent-it

-1

Ebert

1

~Rent-it

-1

Rent-it

1

BigStar

1

B&W

1

Rent-it

~Rent-it

~Rent-it

Rent-it

Rent-it

Rent-it

Rent-it

B&W

~Rent-it

Rent-it

~Rent-it

Rent-it

Rent-it

Rent-it
Decision tree classifiers

[SciFi = -1, Suspense = 1, Romance = -1, Ebert = 1, Siskel = 1, ..., Rent-it???

SciFi

-1

~Rent-it

(1)

BigStar

Rent-it

(1)

Rent-it

~Rent-it

Rent-it

~Rent-it

Rent-it

Rent-it

Rent-it

Rent-it

B&W

~Rent-it

Rent-it

~Rent-it

Rent-it

~Rent-it

Rent-it

~Rent-it
Decision tree classifiers

[ SciFi = -1, Suspense = 1, Romance = -1, Ebert = 1, Siskel = 1, ..., Rent-it?? ]

SciFi
-1

~Rent-it (-1)

~Rent-it

Rent-it (1)

BigStar

-1

Rent-it

~Rent-it

Suspense
-1

~Rent-it

Rent-it

Rent-it

Romance
-1

Rent-it

B&W

-1

~Rent-it

Rent-it

Rent-it
Consider a completely new test datum, with a different value for Romance (and Suspense); I have also shown the value for B&W.

\[
\begin{bmatrix}
  \text{SciFi} = -1, \\
  \text{Suspense} = -1, \\
  \text{Romance} = -1, \\
  \text{Ebert} = 1, \\
  \text{Siskel} = 1, \\
  \text{B&W} = -1, \\
  \ldots, \\
  \text{Rent-it???}
\end{bmatrix}
\]
The values for Romance and B&W of this new datum would lead to a different classification than the previous datum.

\[
\begin{align*}
\text{SciFi} &= -1, \\
\text{Suspense} &= -1, \\
\text{Romance} &= 1, \\
\text{Ebert} &= 1, \\
\text{Siskel} &= 1, \\
\text{B&W} &= -1, \\
\ldots, \\
\text{Rent-it}???
\end{align*}
\]
What decision would be made for the following datum, Rent-it or ~Rent-it?

[SciFi = 1, Suspense = 1, Romance = -1, Ebert = -1, Siskel = 1, BigStar = 1, ..., Rent-it??]

Douglas H. Fisher
\[
[ \text{SciFi} = 1, \text{Suspense} = 1, \text{Romance} = -1, \text{Ebert} = -1, \text{Siskel} = 1, \text{BigStar} = 1, \ldots, \text{Rent-it?} ]
\]
Environment

Learning Component

Labeled data

NEXT UP: LEARNING Decision Trees

 Classifier

Unlabeled datum
such as

$\mathbf{v}_{11}$ $\mathbf{v}_{21}$ $\mathbf{v}_{32}$ ... $\mathbf{v}_{m2}$ $C1$

Labeled datum

$\mathbf{v}_{11}$ $\mathbf{v}_{21}$ $\mathbf{v}_{32}$ ... $\mathbf{v}_{m2}$ $C$?
The standard greedy (hill-climbing) approach
(Top-Down Induction of Decision Trees)

Node TDIDT (Set Data,
  int (* TerminateFn) (Set, Set, Set),
  Variable (* SelectFn) (Set, Set, Set)) {

  IF ((* TerminateFn) (Data)) RETURN ClassNode(Data);

  BestVariable = (* SelectFn)(Data);

  RETURN ( TestNode(BestVariable) )

  TDIDT({d | d in Data and
          Value(BestAttribute, d)
          = v_1})
  TDIDT({d | d in Data and
          Value(BestAttribute, d)
          = v_2})

This is not the only way to learn a decision tree !!
### Data:

<table>
<thead>
<tr>
<th>V1</th>
<th>V2</th>
<th>V3</th>
<th>V4</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>c1</td>
</tr>
<tr>
<td>-1</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>c1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>c1</td>
</tr>
<tr>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>c1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>-1</td>
<td>c2</td>
</tr>
<tr>
<td>-1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>c2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>c2</td>
</tr>
</tbody>
</table>

A datum \( d \rightarrow \text{Vector} \)

### Best-attribute: V4

Assume left branch always corresponds to -1

Assume right branch always corresponds to 1

\[
\text{TDIDT}([-1-1-1c1, -111-1c2, -11-1-1c2, 111-1c2])
\]

\[
\text{TDIDT}([1-111c1, -11-11c1, -1-11c1, -1-111c2])
\]
Best Attribute: V2

TDIDT([-1-11-1c1, -111-1c2, -11-1-1c2, 111-1c2])

Best Attribute: V2

TDIDT([-1-11-1c1])

TDIDT([-111-1c2, -11-1-1c2, 111-1c2])

TDIDT([-11-1-1c1, -1-11-1c1, -1-111c1, -1-111c2])
Number of data at leaf in C1 (right entry) and not in C1 (left entry)
TDIDT([-1-11-1c1])

TDIDT([-111-1c2, -11-1-1c2, 111-1c2])

TDIDT([-1-11-1c1])

TDIDT([1-111c1, -11-11c1, -1-1-11c1, -1-111c2])
BestAttribute: V3

TDIDT([1-111c1, -11-11c1, -1-1-11c1, -1-111c2])
BestAttribute: V3

\[ \text{TDIDT}([-1-11c1, -1-11c1]) \]

\[ \text{TDIDT}([-1-11c1, -1-11c2]) \]
BestAttribute: V1
In general, it might appear that one integer field of a leaf will always be 0, but some termination functions allow “non-pure” leaves (e.g., no split changes the class distribution significantly).

Douglas H. Fisher
Selecting the best divisive attribute (SelectFN):

Attribute $V_i$ that minimizes:

$$\sum_j P(V_i = v_{ij}) \sum_k P(C_k | V_i = v_{ij}) \cdot \log P(C_k | V_i = v_{ij})$$

Expected number of bits necessary to encode $C$ membership conditioned on $V_i = v_{ij}$

Expected number of bits necessary to encode $C$ conditioned on knowledge of $V_i$ value

treat $0 \cdot \log 0$ as 0, else a runtime error will be generated ($\log 0$ is undefined)

Douglas H. Fisher
Selecting the best divisive attribute (SelectFN):

Attribute $V_i$ that minimizes:

$$\sum_j P(V_i = v_{ij}) \sum_k P(C_k | V_i = v_{ij}) \log P(C_k | V_i = v_{ij})$$

- $V_{i1}$
- $V_{i2}$

- $V_1$
- $V_2$
- $V_3$
- $V_4$
- $V_5$

treat $0 \times \log 0$ as 0, else a runtime error will be generated (log 0 is undefined)

$$0.5 \times \left[ (0.5 \times 1) + (0.5 \times 1) \right] +
0.5 \times \left[ (0.5 \times 1) + (0.5 \times 1) \right] = 1$$

$$0.5 \times \left[ (0.75 \times 0.42) + (0.25 \times 2) \right] +
0.5 \times \left[ (0.5 \times 1) + (0.5 \times 1) \right] = 0.9075$$

$$0.5 \times \left[ (0.25 \times 2) + (0.75 \times 0.42) \right] +
0.5 \times \left[ (0.5 \times 1) + (0.5 \times 1) \right] = 0.815$$

$$0.8 \times \left[ (0.9 \times 0.152) + (0.1 \times 3.32) \right] +
0.2 \times \left[ (0.3 \times 1.74) + (0.7 \times 0.52) \right] = 0.5522$$

$$0.5 \times \left[ (1.0 \times 0.0) + (0.0 \times \text{undefined}) \right] +
0.5 \times \left[ (0.0 \times \text{undefined}) + (1.0 \times 0.0) \right] = 0$$
Selecting the best divisive attribute (alternate):

Attribute that maximizes:

$$\sum_{j} P(V_i = v_{ij}) \sum_{k} P(C_k | V_i = v_{ij})^2$$

The big picture on attribute selection:

- if $V_i$ and $C$ are statistically independent, value $V_i$ least
- if each value of $V_i$ associated with exactly one $C$, value $V_i$ most
- most cases somewhere in between
Assume that a decision tree has been constructed from training data, and it includes a node that tests on V at the frontier of the tree, with its left child yielding a prediction of class C1 (because the only training datum there is C1), and the right child predicting C2 (because the only training data there are C2). The situation is illustrated here:

Suppose that during subsequent use, it is found that
i) a large number of items (N > 1000) are classified to the node (with the test on V to the right)
ii) 50% of these have V = -1 and 50% of these have V = 1
iii) post-classification analysis shows that of the N items reaching the node during usage, 25% were C1 and 75% were C2
iv) of the 0.5 * N items that went to the left leaf during usage, 25% were C1 and 75% were C2
v) of the 0.5 * N items that went to the right leaf during usage, 25% were also C1 and 75% were C2

What was the error rate on the sample of N items that went to the sub-tree shown above?

\[ 0.5(0.75) + 0.5(0.25) = 0.5 \]

What would the error rate on the same sample of N items have been if the sub-tree on the previous page (and reproduced here) had been pruned to not include the final test on V, but to rather be a leaf that predicted C2?

\[ 0.25 \]

Issue: C and V are statistically independent in this context (that is, conditionally independent)
Mitigate overfitting by statistical testing for likely dependence?

From data. Consider congressional voting records. Suppose that we have data on House votes (and political party). Suppose variables are ordered Party, Immigration, StarWars, ....

\[ \text{P(Republican)} = 0.52 \quad (226/435 \text{ Republicans} \quad 209/435 \text{ Democrats}) \]

To determine relationship between Party and Immigration, we count

<table>
<thead>
<tr>
<th>Immigration</th>
<th>Actual Counts</th>
<th>Predicted Counts (if Immigration and Party independent)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Republican</td>
<td>17</td>
<td>209</td>
</tr>
<tr>
<td>Democrat</td>
<td>160</td>
<td>49</td>
</tr>
</tbody>
</table>

Very different distributions – conclude dependent

\[ P(\text{Rep}) \cdot P(\text{Yes}) \cdot \frac{435}{(17+160)/435 \cdot 435} = 0.52 \cdot (17+160)/435 \cdot 435 \]
Decomposition and search are important principles in machine learning.
Decomposition and search are important principles in machine learning.
Decomposition and search are important principles in machine learning.
Issues, variations, optimizations, etc:

• continuous attributes
  hard versus soft splits
• other node types (e.g., perceptron trees)
• continuous classes (regression trees)
• **termination conditions (pruning)**
• selection measures (see problem DT1)
• missing values
  during training
  during classification (see expansion)
• noise in data
• irrelevant attributes
• **less greedy variants (e.g., lookahead, search)**
• incremental construction
• applications (e.g., [Banding](#))
• cognitive modeling (e.g., Hunt)
• DT based approaches to nearest neighbor search, object recognition
• background **knowledge** to augment feature space
• **ensembles (forests of decision trees)**
The top-down greedy method is essentially a “hill climb” (section 4.7.1) in what could be a much more extensive search.

The top-down greedy method tends to result in “small” and accurate trees, but a systematic search could do better.
What would be a heuristic for this search?
Ensembles of classifiers
Decision Forests

“Bagging” is one (of several) methods for building a forest. Assume that there are N training data D

Embed greedy DT induction into a loop

For i = 1 to desired size of forest {

    Training Set, TrS = Randomly sample N times from D, with replacement

    Run greedy DT induction on TrS

    Output resulting tree to forest

}

To use the forest classifier, run a test datum through each tree of the forest and take a vote on its classification
More on decision tree classifiers

A decision tree defines disjunctive concepts
(in DNF)
Each path of a decision tree represents a conjunction of values.

A decision tree defines disjunctive concepts.

\[
\text{Rent-it} = \begin{cases} 
\text{Rent-it} & \text{(Ebert = -1 and SciFi = 1 and BigStar = 1)} \\
\text{Rent-it} & \text{(Ebert = 1 and Siskel = -1 and Suspense = 1)} \\
\text{Rent-it} & \text{(Ebert = 1 and Siskel = 1 and Romance = -1)} \\
\text{Rent-it} & \text{(Ebert = 1 and Siskel = 1 and Romance = 1 and B&W = 1)} \\
\text{~Rent-it} & \text{(Ebert = 1 and Siskel = 1 and Romance = 1 and B&W = 0)} \\
\text{~Rent-it} & \text{(Ebert = 1 and Siskel = 1 and Romance = 0 and B&W = 1)} \\
\text{~Rent-it} & \text{(Ebert = 1 and Siskel = 1 and Romance = 0 and B&W = 0)} \\
\end{cases}
\]
What is the DNF representation of \( \sim \text{Rent-it} \)?

\[ \sim \text{Rent-it} = ? \]
~Rent-it definition: each path to a leaf labeled by ~Rent-it is a disjunct in the DNF expression

\[
\neg\text{Rent-it} = \left[ \begin{array}{c}
(Ebert = -1 \text{ and } SciFi = -1) \text{ or }
   (Ebert = -1 \text{ and } SciFi = 1 \text{ and } BigStar = -1) \text{ or }
   (Ebert = 1 \text{ and } Siskel = -1 \text{ and } Suspense = -1) \text{ or }
   (Ebert = 1 \text{ and } Siskel = 1 \text{ and } Romance = 1 \text{ and } B&W = -1)
\end{array} \right]
\]

Douglas H. Fisher
Rent-it = [ (Ebert = -1 and SciFi = 1 and BigStar = 1)  
    or (Ebert = 1 and Siskel = -1 and Suspense = 1)  
    or (Ebert = 1 and Siskel = 1 and Romance = -1)  
    or (Ebert = 1 and Siskel = 1 and Romance = 1 and B&W = 1) ]

In propositional form, write $x=1$ as $x$ and $x=-1$ as $\neg x$,  
‘and’ as $\land$ and ‘or’ as $\lor$

Rent-it = [ ($\neg$ebert $\land$ scifi $\land$ bigstar)  
    $\lor$ (ebert $\land$ $\neg$siskel $\land$ suspense)  
    $\lor$ (ebert $\land$ siskel $\land$ $\neg$romance)  
    $\lor$ (ebert $\land$ siskel $\land$ romance $\land$ $\neg$b&w) ]

$\neg$Rent-it = [ ($\neg$ebert $\land$ $\neg$scifi)  
    $\lor$ ($\neg$ebert $\land$ sciFi $\land$ $\neg$bigstar)  
    $\lor$ (ebert $\land$ $\neg$siskel $\land$ $\neg$suspense)  
    $\lor$ (ebert $\land$ siskel $\land$ romance $\land$ $\neg$b&w) ]
Equivalent Logic Program

\[
\begin{align*}
\text{skips} & \leftarrow \text{long}. \\
\text{reads} & \leftarrow \text{short} \land \text{new}. \\
\text{reads} & \leftarrow \text{short} \land \text{follow\_up} \land \text{known}. \\
\text{skips} & \leftarrow \text{short} \land \text{follow\_up} \land \text{unknown}.
\end{align*}
\]

or with negation as failure:

\[
\begin{align*}
\text{reads} & \leftarrow \text{short} \land \text{new}. \\
\text{reads} & \leftarrow \text{short} \land \sim \text{new} \land \text{known}.
\end{align*}
\]
A decision tree covers all possible data defined over the tree’s variables:

Show that

\[
\neg [ (~e\text{bert} \land scifi \land big\text{star})
\lor (e\text{bert} \land \neg s\text{iskel} \land susp\text{ense})
\lor (e\text{bert} \land s\text{iskel} \land \neg roman\text{ce})
\lor (e\text{bert} \land s\text{iskel} \land roman\text{ce} \land b\&w)]
\]

= 

\[
[ (~e\text{bert} \land \neg scifi)
\lor (~e\text{bert} \land scifi \land \neg big\text{star})
\lor (e\text{bert} \land \neg s\text{iskel} \land \neg susp\text{ense})
\lor (e\text{bert} \land s\text{iskel} \land roman\text{ce} \land \neg b\&w)]
\]

Douglas H. Fisher
Decision trees explicitly encode context

and differences in variables that are most important in differing contexts
Different kinds of variables (though all appear the same to the learning system)

Low level descriptive variables, such as “black-and-white?” or even continuous variables (e.g., runtime < 90 min or >= 90min )

Variables with values that are values of well-defined functions over “basic” variables (e.g., logical equivalence of two binary variables; the square of a more basic continuous variable)

Variables with values that are complex (and UNKNOWN) functions of other variables:

Genre (human consensus)

Human recommendations (experts, friends, etc)

Other recommender systems (or AIs generally) like those of Netflix, iTunes, etc

Douglas H. Fisher