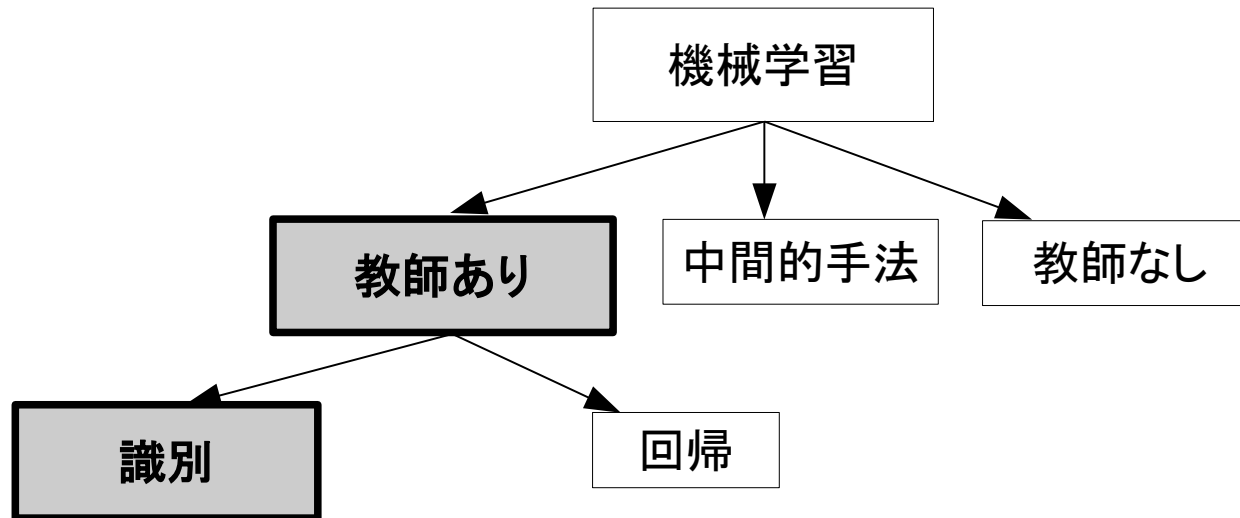


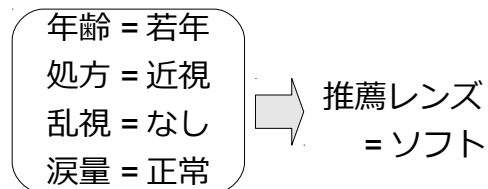
3. 識別 ―概念学習と決定木―

- 問題設定

- 教師あり学習
- ラベル入力 → ラベル出力



- ラベル特徴



- 数値特徴

contact-lenses データ

年齢・処方・
乱視・涙量

Relation: contact-lenses					
	1: age Nominal	2: spectacle-prescrip Nominal	3: astigmatism Nominal	4: tear-prod-rate Nominal	5: contact-lenses Nominal
1	young	myope	no	reduced	none
2	young	myope	no	normal	soft
3	young	myope	yes	reduced	none
4	young	myope	yes	normal	hard
5	young	hypermetrope	no	reduced	none
6	young	hypermetrope	no	normal	soft
7	young	hypermetrope	yes	reduced	none
8	young	hypermetrope	yes	normal	hard
9	pre-presbyopic	myope	no	reduced	none
10	pre-presbyopic	myope	no	normal	soft
11	pre-presbyopic	myope	yes	reduced	none
12	pre-presbyopic	myope	yes	normal	hard
13	pre-presbyopic	hypermetrope	no	reduced	none
14	pre-presbyopic	hypermetrope	no	normal	soft
15	pre-presbyopic	hypermetrope	yes	reduced	none
16	pre-presbyopic	hypermetrope	yes	normal	none
17	presbyopic	myope	no	reduced	none
18	presbyopic	myope	no	normal	none
19	presbyopic	myope	yes	reduced	none
20	presbyopic	myope	yes	normal	hard
21	presbyopic	hypermetrope	no	reduced	none
22	presbyopic	hypermetrope	no	normal	soft
23	presbyopic	hypermetrope	yes	reduced	none
24	presbyopic	hypermetrope	yes	normal	none

Undo OK Cancel

推薦コンタクトレンズ
none, soft, hard


3.2 概念学習とは

- 概念学習とは

- 正解の概念を説明する特徴ベクトルの性質（論理式）を求めること
- 論理式の例

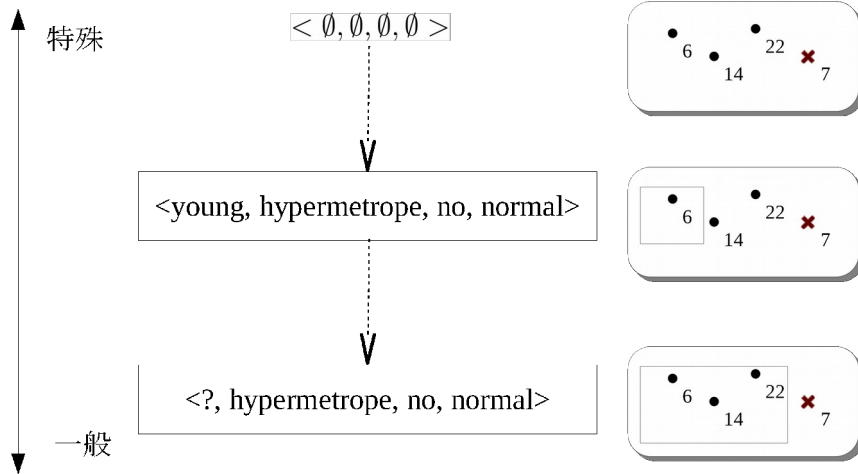
（乱視 = あり） \wedge （ドライアイ = なし） \Rightarrow soft

- 学習の方法

- 可能な論理式が少数
 - 正解概念の候補を絞り込んでゆく（候補削除アルゴリズム）
- 可能な論理式が多数
 - バイアス（偏見）をかけて探索する  決定木

3.3 初歩的な概念学習アルゴリズム

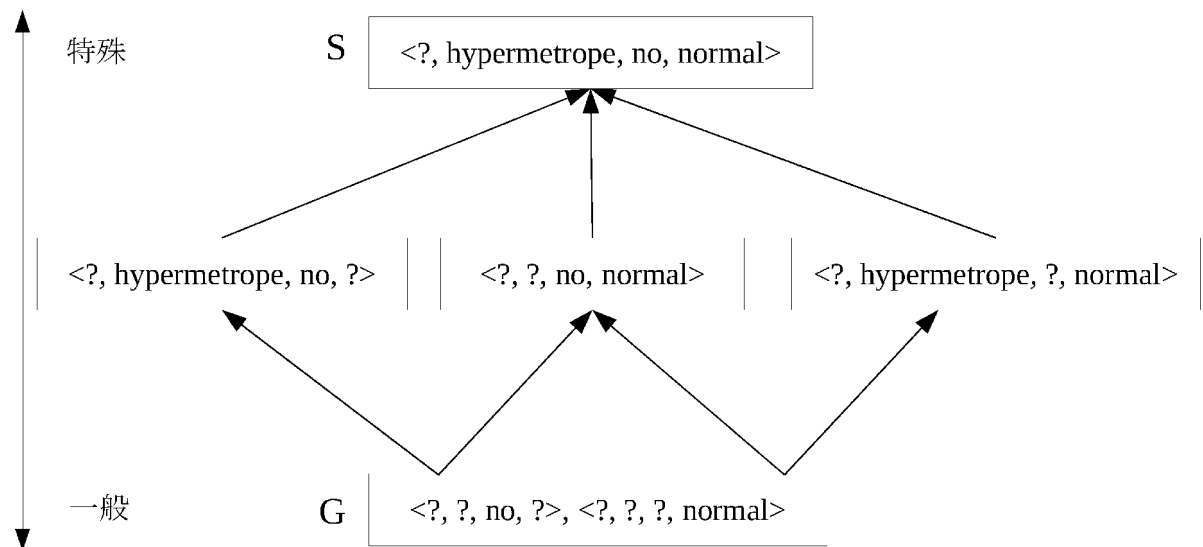
FIND-S アルゴリズム



対象概念をカバーするように
仮説を段階的に一般化

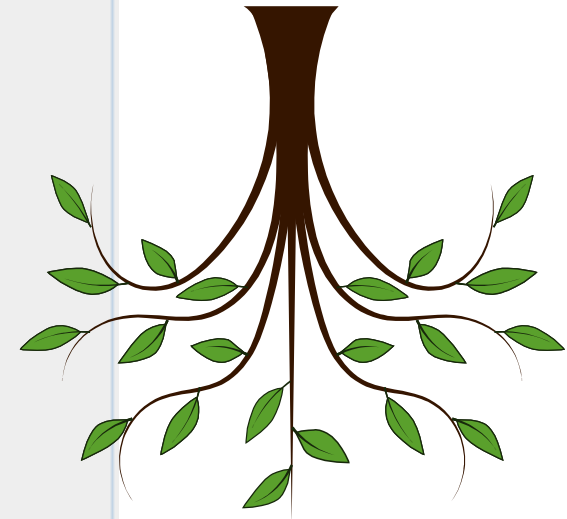
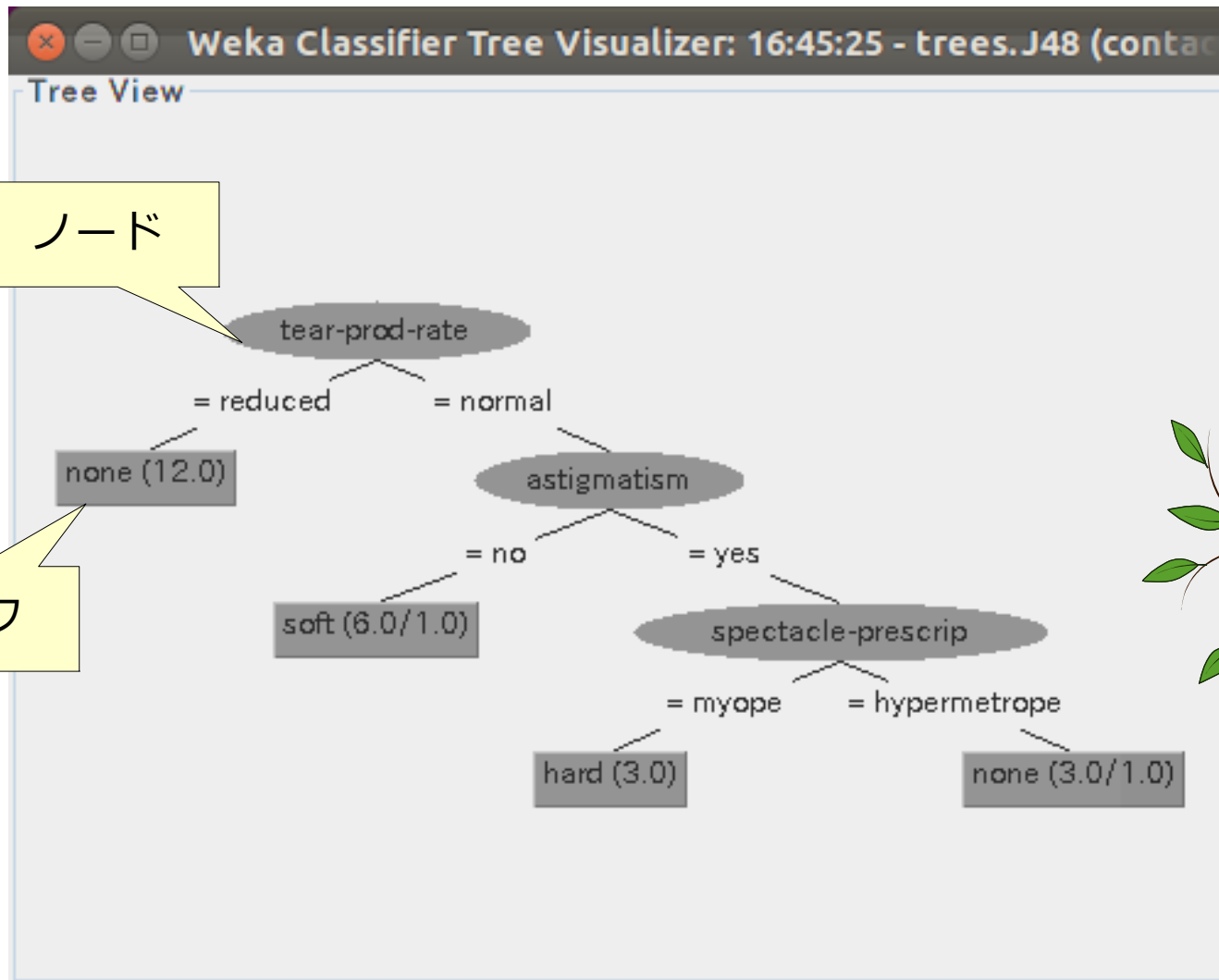
候補削除アルゴリズム

正例：カバーするように一般化
負例：カバーしないように特殊化



3.4 決定木の学習

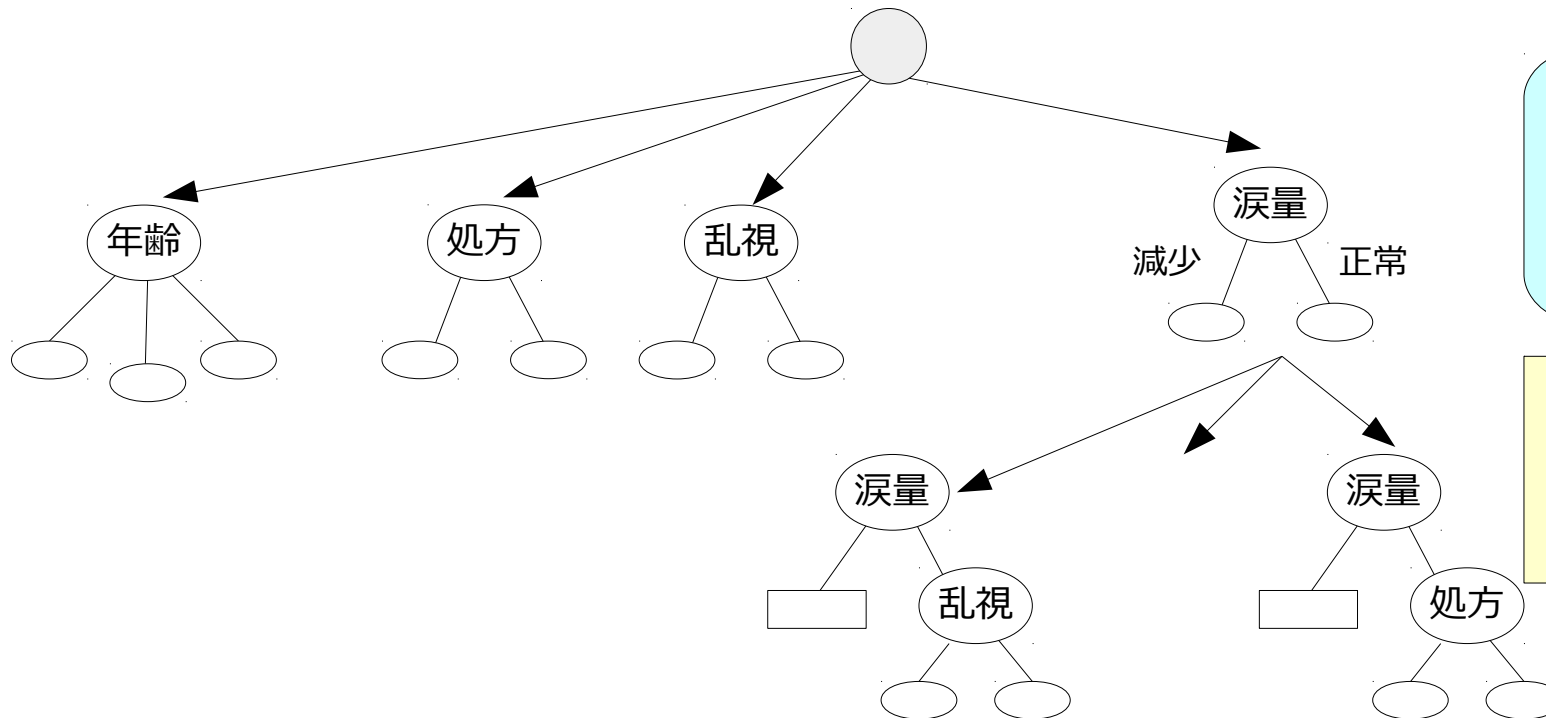
- 学習した決定木の例



3.4 決定木の学習

- 決定木学習の考え方

- ノードは、データを分割する条件を持つ
 - できるだけ同一クラスのデータがリーフに偏るように
- 分割後のデータ集合に対して、同様の操作を行う
- 全てのリーフが単一クラスの集合になれば終了



この手順に従うと、
一般には小さな木
ができる

バイアス

複雑な説明よりも
単純な説明の方が
汎用性が高い

Algorithm 3.3 ID-3 アルゴリズム

入力: 正解付学習データ D , クラス特徴 y , 特徴集合 A

出力: 決定木 T

 root ノードを作成

if D が全て正例 **then**

return ラベル Yes

else if D が全て負例 **then**

return ラベル No

else if 特徴集合 $A == \emptyset$ **then**

return データ中の最頻値のラベル

else

$a \leftarrow A$ 中で最も分類能力の高い特徴

 root ノードの決定特徴 $\leftarrow a$

for all a の取りうる値 v **do**

$a = v$ に対応する枝を作成

 データの中から値 v を取る部分集合 D_v を作成

if $D_v == \emptyset$ **then**

return データ中の最頻値のラベル

else

 ID3(部分集合 D_v , クラス特徴 y , 特徴集合 $A - a$)

end if

end for

end if

return root ノード

属性の分類能力 (1/2)

- 分類能力の高い属性を決定する方法
 - その属性を使った分類を行うことによって、なるべくきれいにクラスが分かれるように
- エントロピー
 - データ集合 S の乱雑さを表現
 - 正例の割合 : p^+ , 負例の割合 : p^-
 - エントロピーの定義

$$Entropy(S) = -p^+ \log p^+ - p^- \log p^-$$

属性の分類能力 (2/2)

- 情報獲得量

- 属性 A を用いた分類後のエントロピーの減少量
- 値 v を取る訓練例の集合 : S_v
- S_v の要素数 : $|S_v|$
- 情報獲得量の定義

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Weka での決定木学習

決定木は J48

学習データを
評価に使う

右クリック→
Visualize tree
で木を表示

The screenshot shows the Weka Explorer window with the 'Classify' tab selected. The 'Classifier' dropdown is set to 'J48 -C 0.25 -M 2'. Under 'Test options', 'Use training set' is selected. The 'Result list' shows '12:44:23 - trees.J48' selected. The 'Classifier output' pane displays the following information:

```
=== Run information ===

Scheme:      weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:    contact-lenses
Instances:   24
Attributes:  5
              age
              spectacle-prescrip
              astigmatism
              tear-prod-rate
              contact-lenses

Test mode:   evaluate on training data

=== Classifier model (full training set) ===

J48 pruned tree
-----

tear-prod-rate = reduced: none (12.0)
tear-prod-rate = normal
|   astigmatism = no: soft (6.0/1.0)
|   astigmatism = yes
|   |   spectacle-prescrip = myope: hard (3.0)
|   |   spectacle-prescrip = hypermetrope: none (3.0/1.0)

Number of Leaves :    4
Size of the tree :    7

Time taken to build model: 0 seconds
```

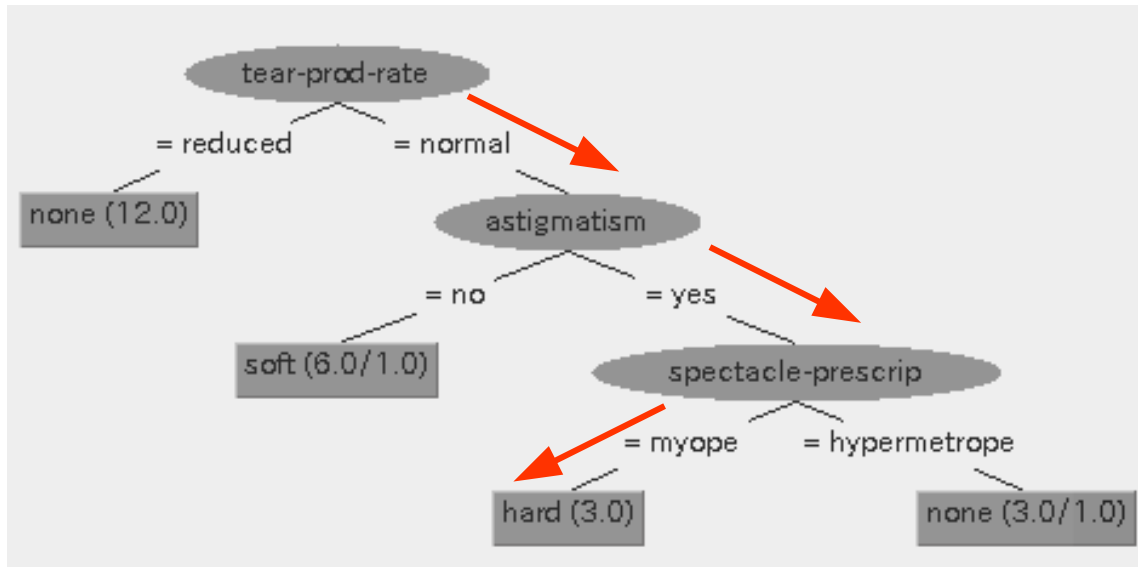
At the bottom, the status is 'OK' and there is a 'Log' button.

木のテキスト
表示

実行例

- 入力

年齢 処方 乱視 涙量
(young, myope, yes, normal)



- 出力

hard

バイアスの検討

なぜ単純な木の方がよいか

- オッカムの剃刀

「データに適合する最も単純な仮説を選べ」

- 複雑な仮説

- 表現能力が高い

- 偶然にデータを説明できるかもしれない

- 単純な仮説

- 表現能力が低い

- 偶然にデータを説明できる確率は低い

- でも説明できた！

- **必然**

連続値属性の扱い

- 連続値 A を持つ属性から真偽値 ($A < c?$) を値とするノードを作成

→ c をどうやって決めるか

気温	40	48	60	72	80	90
playTennis	No	No	Yes	Yes	Yes	No

$$c = (48 + 60) / 2 \\ = 54$$

$$c = (80 + 90) / 2 \\ = 85$$

情報獲得量の多い方

連続値属性の扱い

Weka Explorer

Preprocess | **Classify** | Cluster | Associate | Select attributes | Visualize

Classifier
Choose J48 -G 0.25 -M 2

Test options
☐ Use training set
☐ Supplied test set Set...
☒ Cross-validation Folds 10
☐ Percentage split % 66
More options...

(Nom) class

Start Stop

Result list (right-click for options)
16:44:11 - lazy.IBk
16:45:25 - trees.J48

Classifier output

```
Test mode: 10-fold cross-validation  
=== Classifier model (full training set) ===  
J48 pruned tree  
-----  
petalwidth <= 0.6: Iris-setosa (50.0)  
petalwidth > 0.6  
| petalwidth <= 1.7  
| | petallength <= 4.9: Iris-versicolor (48.0/1.0)  
| | petallength > 4.9  
| | | petalwidth <= 1.5: Iris-virginica (3.0)  
| | | petalwidth > 1.5: Iris-versicolor (3.0/1.0)  
| petalwidth > 1.7: Iris-virginica (46.0/1.0)  
leaves : 5  
tree : 9  
to build model: 0.01 seconds  
ried cross-validation ===  
===  
classified Instances 144 96 %  
Classified Instances 6 4 %  
stic 0.94  
te error 0.035  
quared error 0.1586  
olute error 7.8705 %
```

Weka Classifier Tree Visualizer: 16:46:52 - trees.J48 (iris)

Tree View

```
graph TD
    A(petalwidth) -- "<= 0.6" --> B[Iris-setosa (50.0)]
    A -- "> 0.6" --> C(petalwidth)
    C -- "<= 1.7" --> D(petallength)
    C -- "> 1.7" --> E[Iris-virginica (46.0/1.0)]
    D -- "<= 4.9" --> F[Iris-versicolor (48.0/1.0)]
    D -- "> 4.9" --> G(petalwidth)
    G -- "<= 1.5" --> H[Iris-virginica (3.0)]
    G -- "> 1.5" --> I[Iris-versicolor (3.0/1.0)]
```

Log x 0