# Indukcja drzew

## Część 2


JERZY STEFANOWSKI
Instytut Informatyki
Politechnika Poznańska

Uwagi do wykładu dla Ucz. Maszynowe
Aktualizacja 2016 i 2019

# Ograniczenia w uczeniu się drzew decyzyjnych

Pytania i problemy, np.:

- Jak uwzględniać atrybuty ze zbyt dużą liczbą wartości w stosunku do dziedzin pozostałych atrybutów?
- Jak uwzględniać atrybuty ilościowe?
- Jak uwzględniać atrybuty z nieznanymi wartościami?
- Jak uwzględniać dane "zaszumione"?
- Kiedy należy zaprzestać rozbudowywać drzewa?
  - aby zapobiec” przespecjalizowaniu” opisu
  - duże drzewa są trudne do analizy i zrozumienia

# Silnie wielowartościowe atrybuty

- Trudności z atrybutami o b. licznych dziedzinach (extreme case: ID code), zwłaszcza jeśli inne atrybuty mają mniej liczne dziedziny.
  - Patrz, np., UCI dane crx
- Pewne „obciążenie” użycia entropii (dążenie do tzw. czystości informacji)

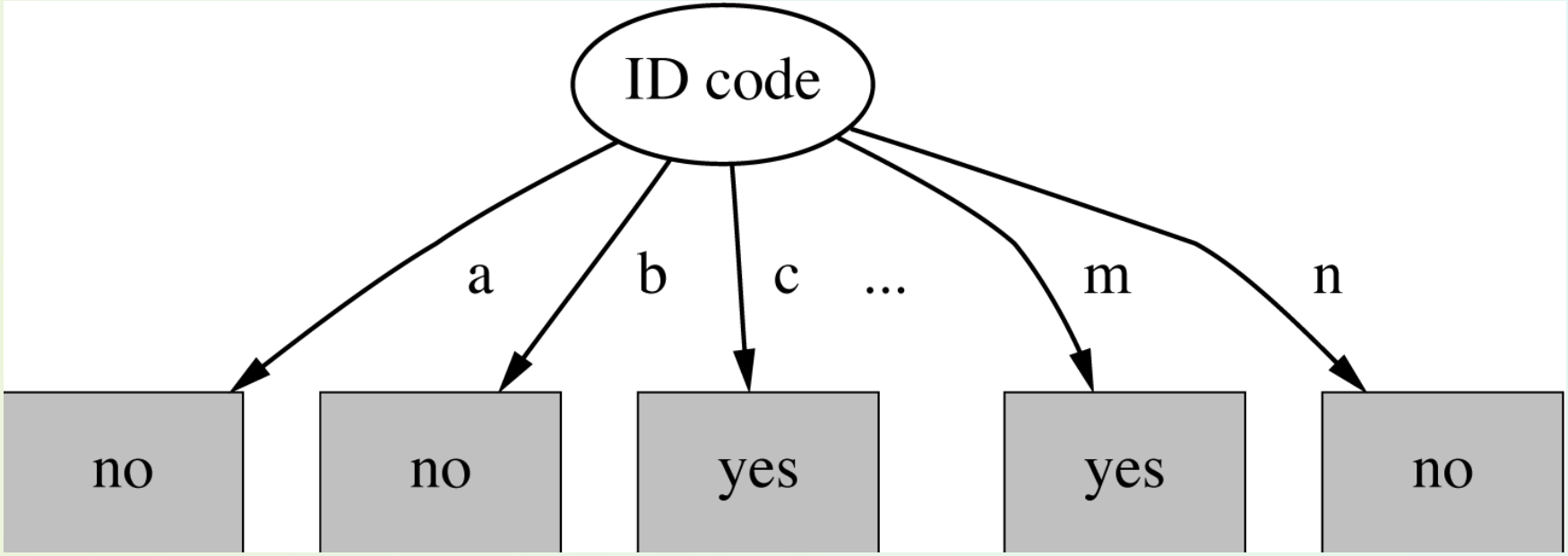
  ⇒ zysk informacyjny preferuje “czyste” podziały, nawet jeśli wspierane przez mało przykładów

  ⇒Tendencja do tzw. *overfitting* (wybór atuybutów o niskiej przydatności do predekcji nowych obiektów)

# Weather Data with ID code

| ID | Outlook | Temperature | Humidity | Windy | Play? |
|---|---|---|---|---|---|
| a | sunny | hot | high | false | No |
| b | sunny | hot | high | true | No |
| c | overcast | hot | high | false | Yes |
| d | rain | mild | high | false | Yes |
| e | rain | cool | normal | false | Yes |
| f | rain | cool | normal | true | No |
| g | overcast | cool | normal | true | Yes |
| h | sunny | mild | high | false | No |
| i | sunny | cool | normal | false | Yes |
| j | rain | mild | normal | false | Yes |
| k | sunny | mild | normal | true | Yes |
| l | overcast | mild | high | true | Yes |
| m | overcast | hot | normal | false | Yes |
| n | rain | mild | high | true | No |

# Podział w drzewie dla ID Code Attribute




Entropy of split = 0 (since each leaf node is “pure”, having only one case).

Information gain najkorzystniejszy dla atr. ID code
Lecz co z możliwością ugólnienia (ang. Generalization error)

# Gain ratio – normalizacja entropii

- Propozycje: ***Gain Ratio*** [Quinlan 93]
- ***Intuicja***: karać zbyt szerokie dziedziny atrybutów; po to aby dążyć do **znormalizowanych** rozkładów wartości
- "Gain ratio takes number and size of branches into account when choosing an attribute.
  - It corrects the information gain by taking the *intrinsic information* of a split into account (i.e. how much info do we need to tell which branch an instance belongs to)".

$$GainRatio(S,A) = \frac{Gain(S,A)}{IntrinsicInfo(S,A)}.$$

$$IntrinsicInfo(S,A) \equiv -\sum \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}.$$

# Binary Tree – budowa drzew binarnych

- Drzewa binarne mogę być skuteczniejsze w klasyfikacji nowych faktów
- Podział binarny w węźle drzewa:
  - Atrybuty liczbowe A, reprezentacja w postaci value(A)<x gdzie x jest wartością z dziedziny A.
  - Atrybuty nieliczbowe A, warunek w postaci value(A)$\in$X gdzie X$\subset$domain(A)

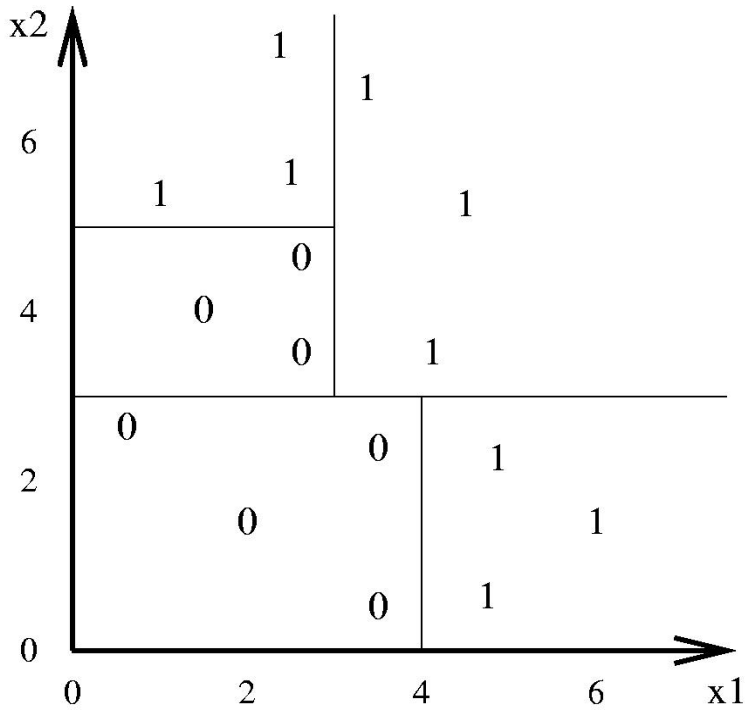
# Binary tree (Quinlan's C4.5 output)

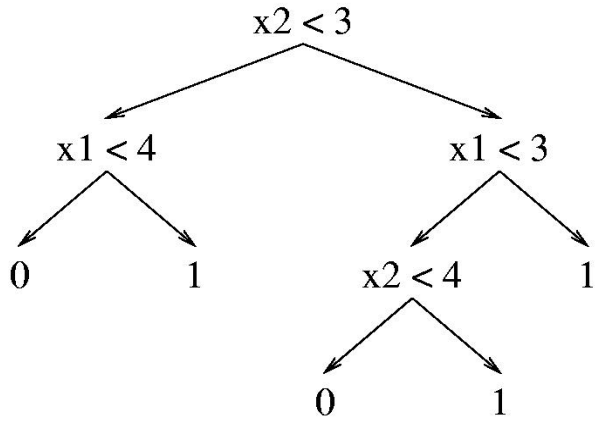
```
Pruned decision tree:

A9 = t:
|   A15 > 228 : + (106.0/3.8)
|   A15 <= 228 :
|   |   A14 <= 102 :
|   |   |   A4 in {l,t}: + (0.0)
|   |   |   A4 = u:
|   |   |   |   A6 in {c,d,cc,i,k,m,q,w,x,e,aa}: + (46.4/3.1)
|   |   |   |   A6 in {j,ff}: - (2.0/1.0)
|   |   |   |   A6 = r: + (0.0)
|   |   |   A4 = y:
|   |   |   |   A6 in {c,i,aa,ff}: - (7.0/3.4)
|   |   |   |   A6 in {d,j,w,x}: + (4.0/1.2)
|   |   |   |   A6 in {cc,k,m,r,q,e}: + (0.0)
|   |   A14 > 102 :
|   |   |   A6 in {j,r}: + (0.0)
|   |   |   A6 in {c,d,k,m,e,aa,ff}:
|   |   |   |   A14 <= 132 : - (4.1/1.2)
|   |   |   |   A14 > 132 :
|   |   |   |   |   A3 <= 1.625 :
|   |   |   |   |   |   A14 <= 292 : - (13.0/1.3)
|   |   |   |   |   |   A14 > 292 :
|   |   |   |   |   |   |   A13 = g: + (2.0/1.0)
|   |   |   |   |   |   |   A13 = s: - (6.0/2.3)
|   |   |   |   |   |   |   A13 = p: - (0.0)
|   |   |   |   |   A3 > 1.625 :
|   |   |   |   |   |   A6 in {k,m}: + (5.0/1.2)
|   |   |   |   |   |   A6 = ff: + (0.0)
|   |   |   |   |   |   A6 in {c,d,e,aa}:
|   |   |   |   |   |   |   A2 <= 32.08 : + (9.5/4.1)
|   |   |   |   |   |   |   A2 > 32.08 : - (8.0/3.5)
|   |   |   A6 in {cc,i,q,w,x}:
|   |   |   |   A8 <= 10.75 : + (36.0/9.3)
|   |   |   |   A8 > 10.75 : - (2.0/1.0)
A9 = f:
|   A4 in {u,y}: - (237.0/17.3)
|   A4 = l: + (2.0/1.0)
|   A4 = t: - (0.0)
```

- Crx (Credit Data) UCI ML Repository

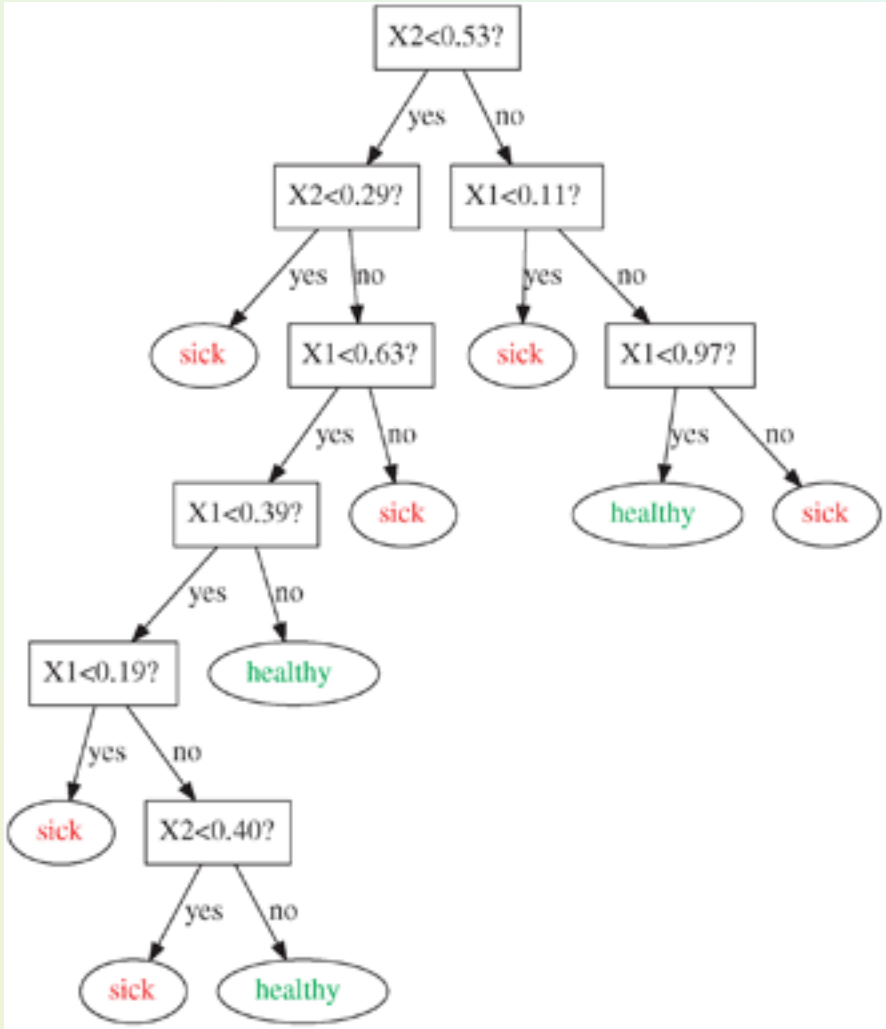
# Decision Tree Decision Boundaries

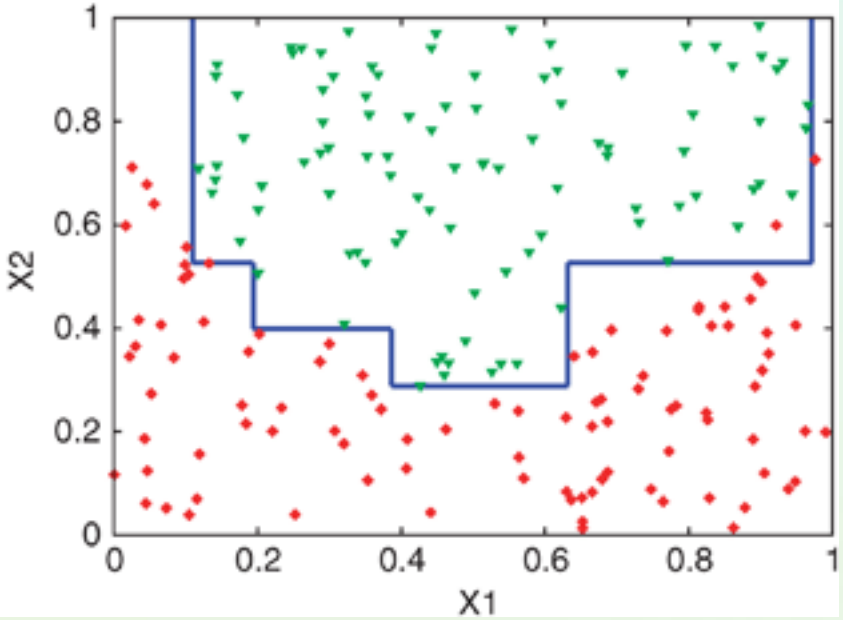
Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the $K$ classes.



# Przykład medyczny wizualizacja drzewa



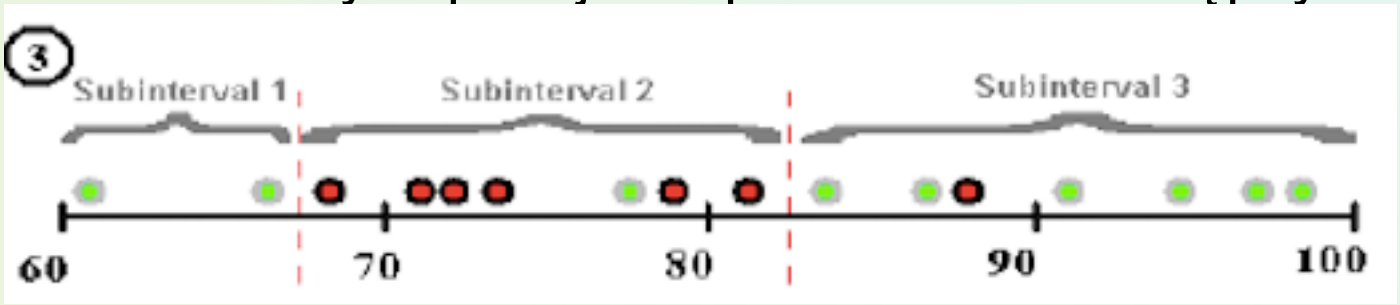
# Weather data – zmiana atr. na skale liczbowe

| Outlook | Temperature | Humidity | Windy | Play |
|---|---|---|---|---|
| Sunny | 85 | 85 | False | No |
| Sunny | 80 | 90 | True | No |
| Overcast | 83 | 86 | False | Yes |
| Rainy | 75 | 80 | False | Yes |
| … | … | … | … | … |

```
If outlook = sunny and humidity > 83 then play = no
If outlook = rainy and windy = true then play = no
If outlook = overcast then play = yes
If humidity < 85 then play = yes
If none of the above then play = yes
```

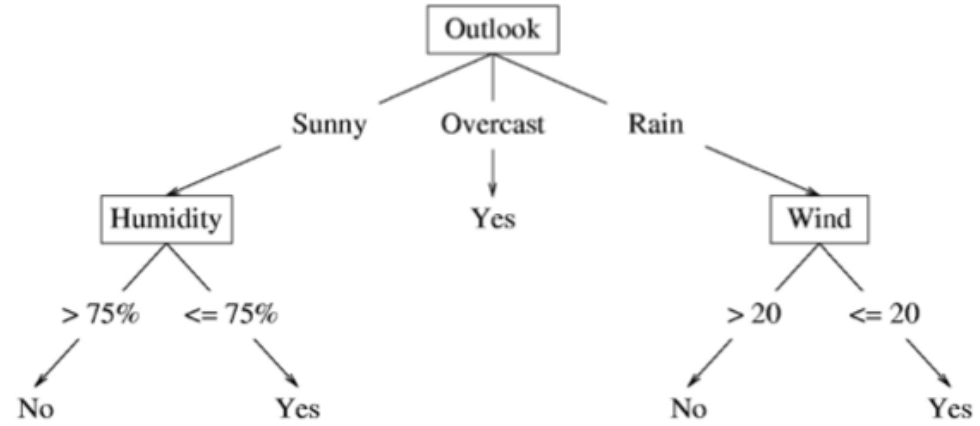
# Atrybuty liczbowe: Dyskretyzacja

- Niektóre metody wymagają danych dyskretnych, np. Naïve Bayes, zbiory przybliżone, reguły asocjacyjne, wzorce sekwencji.
- Dyskretyzacja jest:
  - Procesem zamiany atrybutów liczbowych na atrybuty symboliczne typu porządkowego - Podział oryginalnej dziedziny atrybutu liczbowego na pewną liczbę przedziałów i przypisaniu tym przedziałom kodów symbolicznych.
  - Wiele różnorodnych podejść w przetwarzaniu wstępnym




- Czy dotyczy to także drzew?
- W przypadku drzew – rodzaj wewnętrznej dyskretyzacji lokalnej w trakcie binaryzacji

# Poszukuj najlepszego podziału w węźle




- Change to binary splits by choosing a threshold
- One method:
  - Sort instances by value, identify adjacencies with different classes

| Temperature: | 40 | 48 | 60 | 72 | 80 | 90 |
|---|---|---|---|---|---|---|
| PlayTennis: | No | No | Yes | Yes | Yes | No |

candidate splits

Zastosuj metodę oceny wg. info-gain ale dla podziału binarnego na S1 I S2

# Binaryzacja atrybutu ilościowego

- Punkt podziału - Split dla atr. temperature :

| 64 | 65 | 68 | 69 | 70 | 71 | 72 | 72 | 75 | 75 | 80 | 81 | 83 | 85 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Yes | No | Yes | Yes | Yes | No | No | Yes | Yes | Yes | No | Yes | Yes | No |

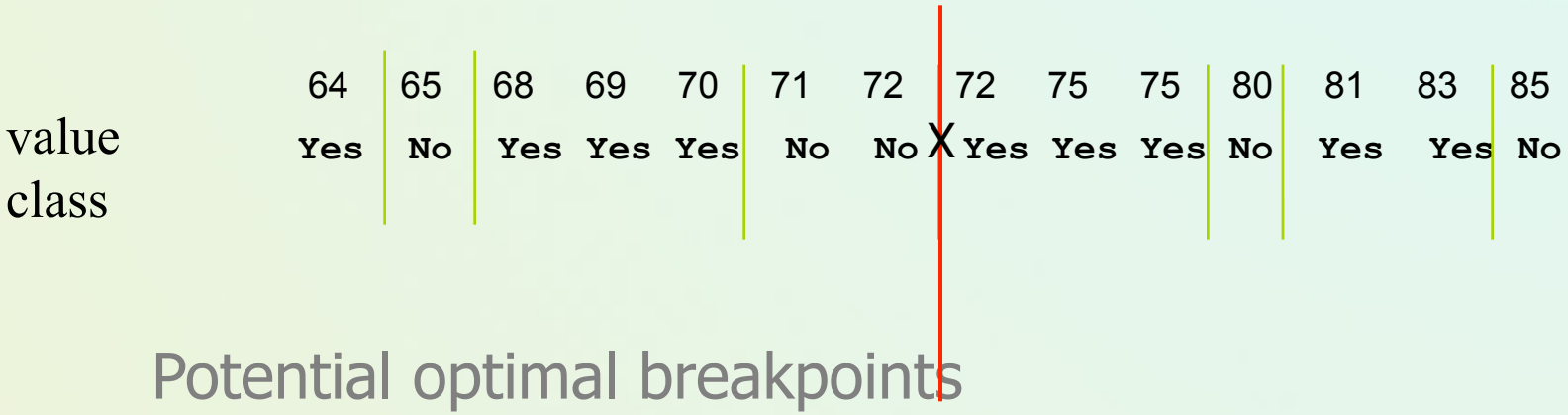
  - E.g. temperature < 71.5: yes/4, no/2
    temperature ≥ 71.5: yes/5, no/3

  - Info([4,2],[5,3])
    = 6/14 info([4,2]) + 8/14 info([5,3])
    = 0.939

- Wstaw próg między istniejące przykłady
- Efektywne obliczeniowo

# Szybsze obliczenia

- Entropy only needs to be evaluated between points of different classes (Fayyad & Irani, 1992)




Potential optimal breakpoints

Breakpoints between values of the same class cannot be optimal

# Inne wyzwanie …

| Outlook | Temperature | Humidity | Windy | Play? |
|---|---|---|---|---|
| sunny | hot | high | false | No |
| sunny | hot | high | true | No |
| overcast | hot | high | false | Yes |
| rain | mild | high | false | Yes |
| rain | cool | normal | false | Yes |
| rain | cool | normal | true | No |
| overcast | cool | normal | true | Yes |
| sunny | mild | high | false | No |
| **sunny** | **cool** | **normal** | **false** | **Yes** |
| rain | mild | normal | false | Yes |
| sunny | mild | normal | true | Yes |
| overcast | mild | high | true | Yes |
| overcast | hot | normal | false | Yes |
| rain | mild | high | true | No |

*Note:*
*All examples are consistent*

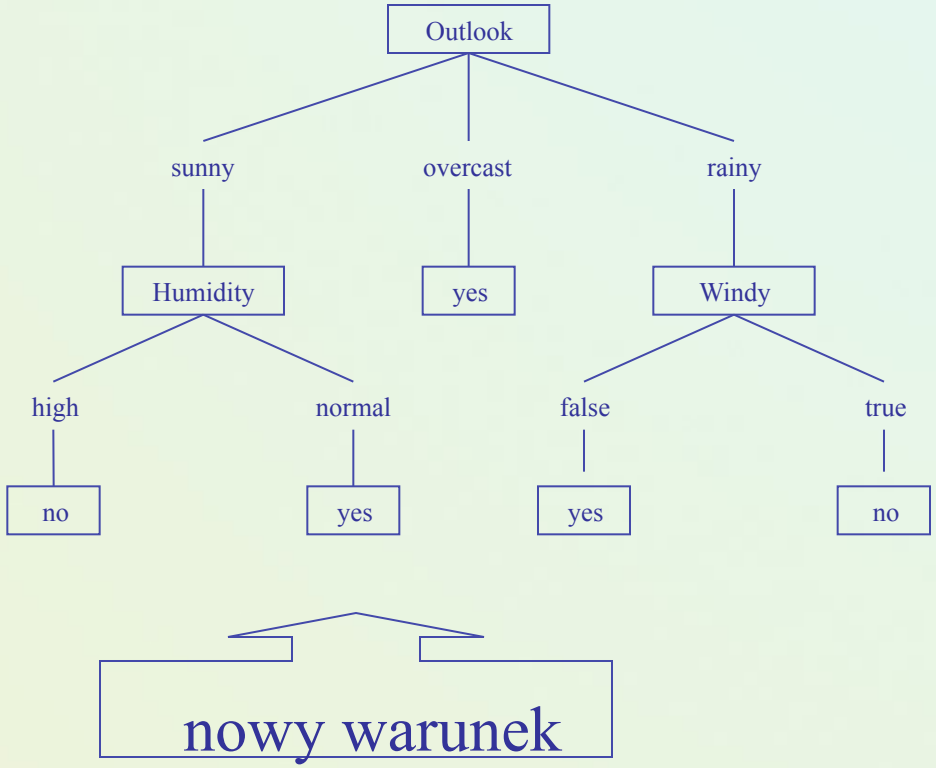
# Sprzeczne przykłady (Inconsistency)

- Opis przynajmniej dwóch przykładów tymi samymi wartościami atrybutów, lecz inne etykiety klas
- Przyczyny
  - Brak dodatkowych informacji (np. medycyna)
  - Niespójność ocen wielu ekspertów
  - Błąd pomiaru, przekłamanie w rejestracji (ang. **noise** – szum informacyjny, tzw. zaszumiony przykład); Także błąd etykietowania (label noise vs. attribute noise)
- Jakie są konsekwencje dla  indukcji drzew oraz ich zdolności predykcji?

# Sprzeczne opisy przykładów

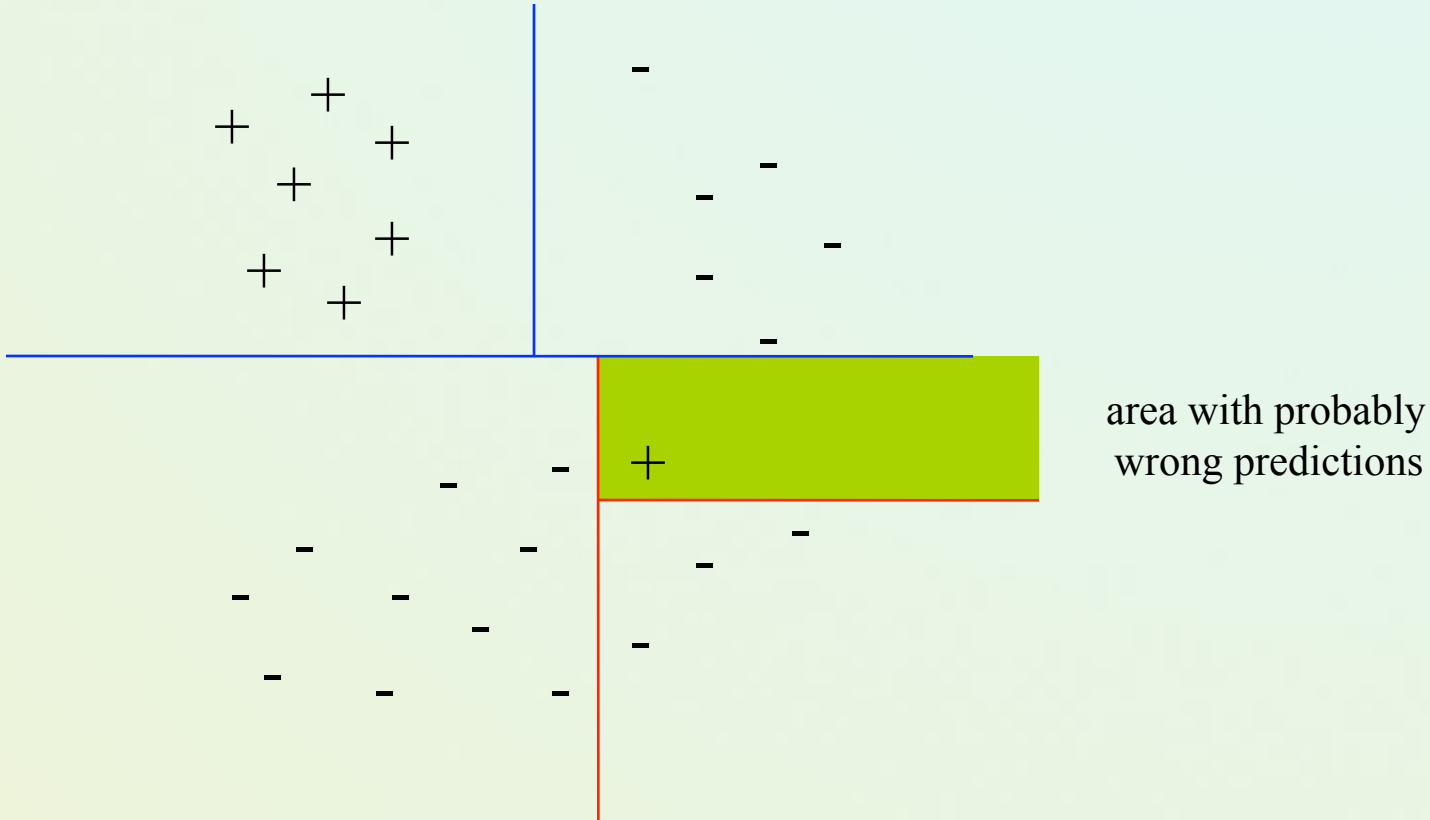
- ***Incosistent training instances***:

Outlook = *Sunny;* Temperature = *Cool;* Humidity = *Normal;* Wind = *False;* PlayTennis = *No*



# Inne przyczyny przeuczenia drzewa

- ***Small number of instances are associated with leaf nodes.*** In this case it is possible that for coincidental regularities to occur that are unrelated to the actual target concept.



# Przeuczenie klasyfikatora (Overfitting)

- Dobry klasyfikator (drzewo) musi nie tylko wystarczająco spójnie odwzorowywać dane uczące, lecz także trafnie klasyfikować nowe dane (niewidziane w trakcie procesu uczenia się).
- Innymi słowami – klasyfikator musi mieć niski błąd uczący, lecz przede wszystkim niski błąd uogólnienia na nowe dane (testowe)
  - Training error vs. generalization error (testing error)
- Nadmiernie rozbudowane drzewo, dopasowane do trudnych przykładów uczących traci zdolności uogólniania.

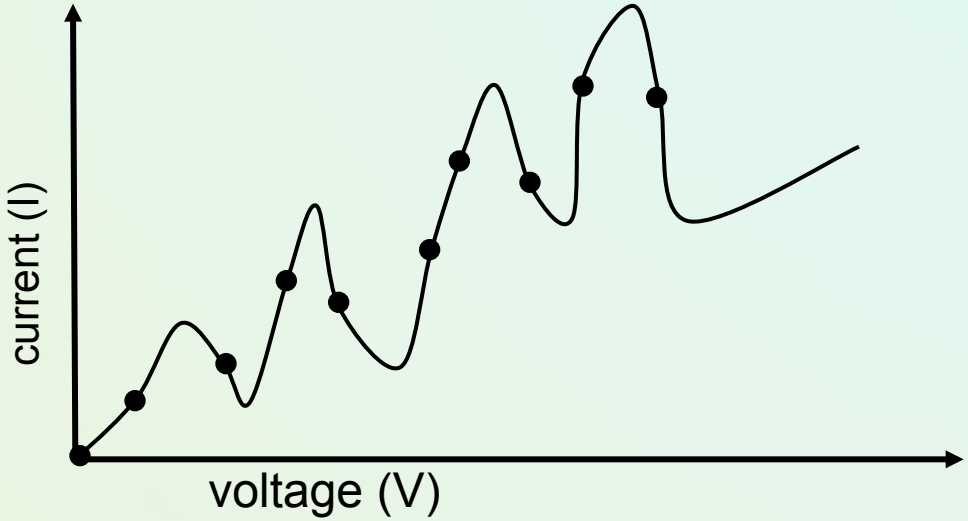
# Overfitting - ilustracja przeuczenia

**Testing Ohms Law: V = IR (I = (1/R)V)**

Experimentally measure 10 points

Fit a curve to the

Resulting data.

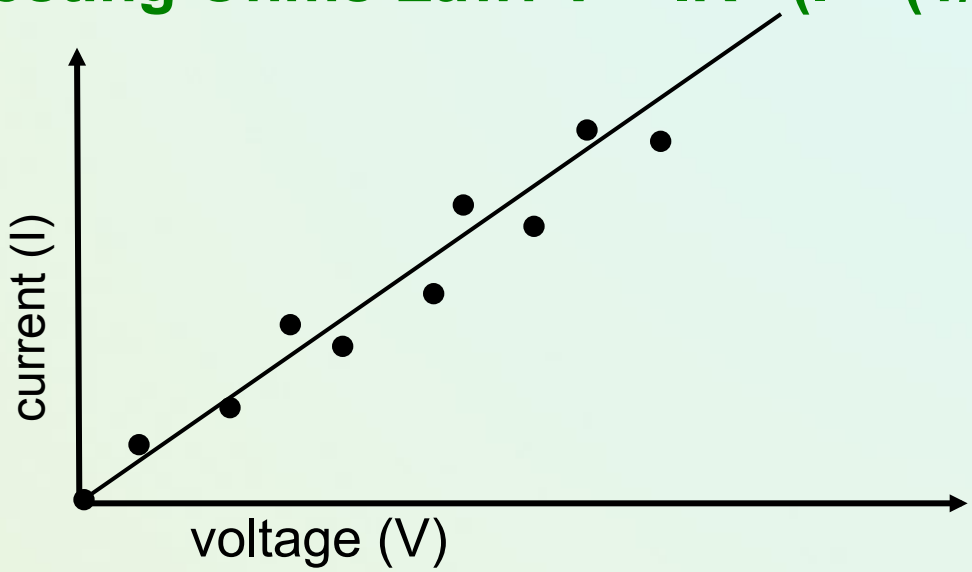



Perfect fit to training data with an 9th degree polynomial
(can fit *n* points exactly with an *n*-1 degree polynomial)

**Ohm was wrong, we have found a more accurate function!**

# Unikajmy nadmiernego dopasowania

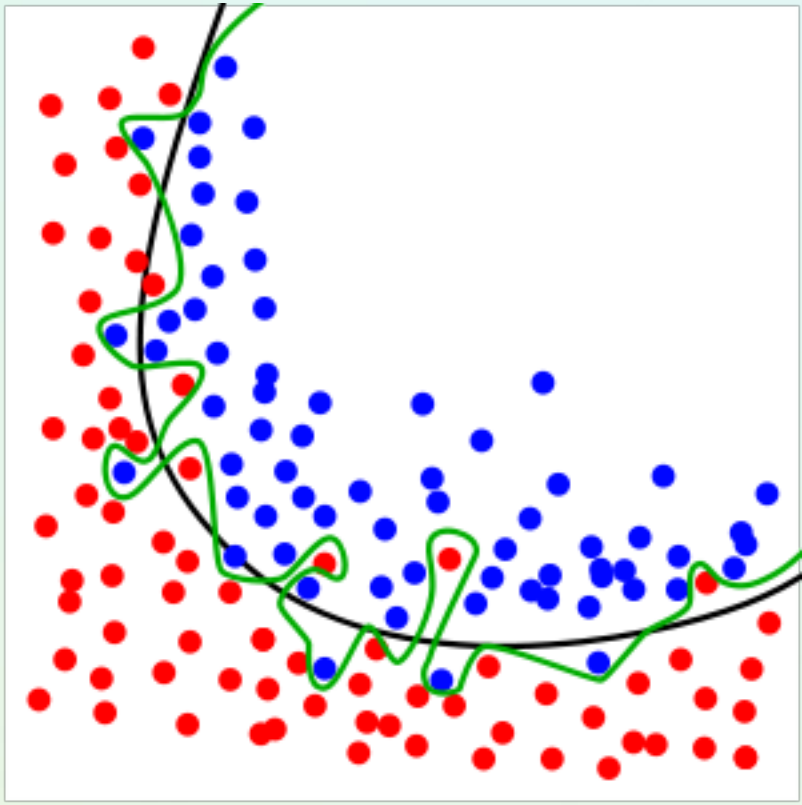
**Testing Ohms Law: V = IR (I = (1/R)V)**



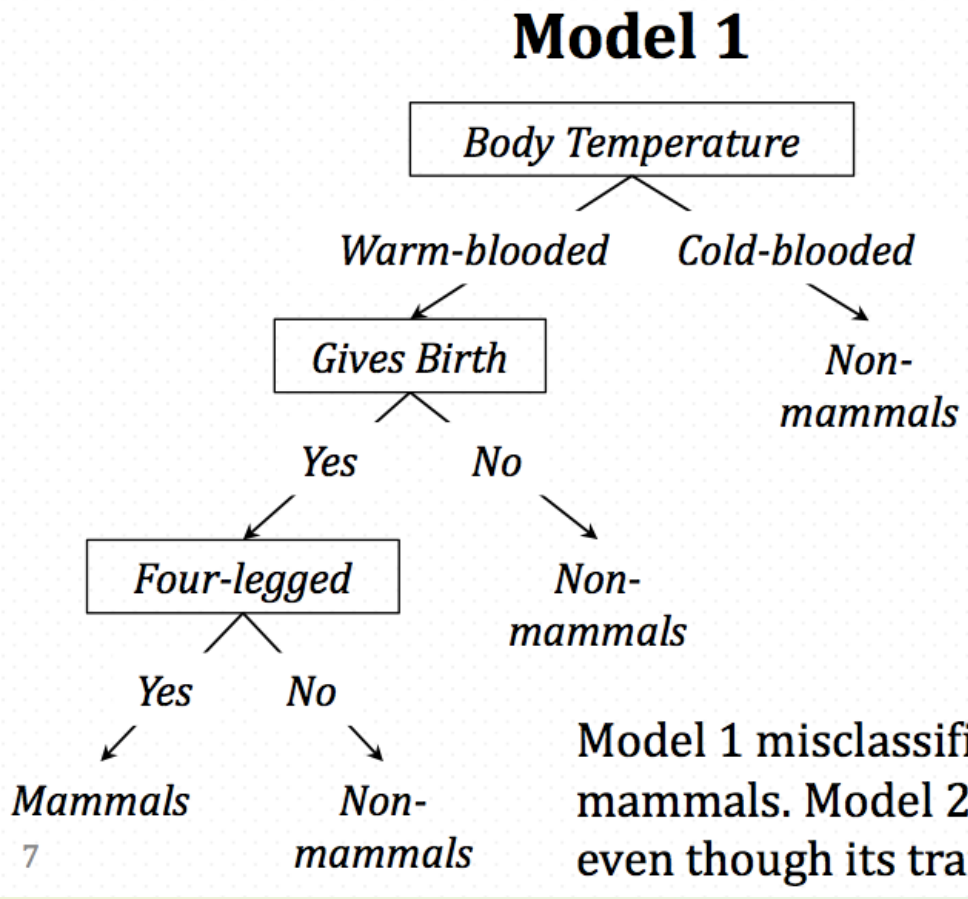

Better generalization with a linear function that fits training data less accurately.

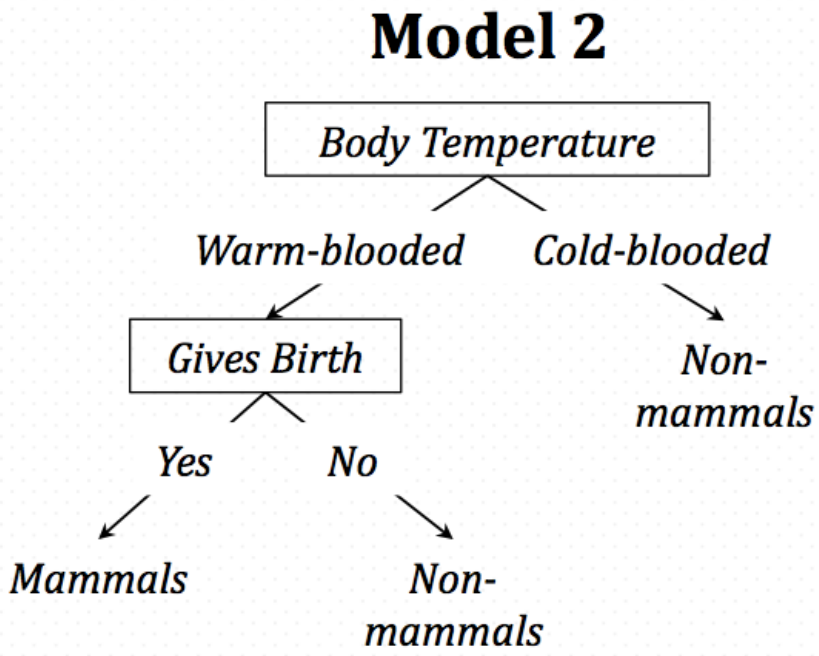
# Przyczyny przeuczenia

- Obecność sprzecznych albo zaszumionych przykładów
- Brak w zbiorze uczącym wystarczającej liczby reprezentatywnych przykładów
- Nadmierne przeszukiwanie zbyt dużej przestrzeni możliwych hipotez (ang. Multiple comparision procedures and spurious fitting) – zwłaszcza dla wysoce wielowymiarowych przestrzeni + gdy model ma zbyt dużo parametrów w stosunku do rozmiaru danych uczących.



# Przykład błędnych predykcji dla przeuczonego drzewa




Model 1 misclassifies humans and dolphins as non-mammals. Model 2 has a lower test error rate (10%) even though its training error rate is higher (20%).



UCI zoo data opisujące cechy różnych stworzeń

# Inspiracja – Brzytwa Ockhama

- “Everything should be made as simple as possible, but no simpler.”

- All other things being equal, simple theories are preferable to complex ones.

# Brzytwa Ockhama – proste hipotezy

Dlaczego preferować prostsze drzewa?

1. Jest dużo mniej prostych hipotez niż złożonych, więc mała szansa, że przypadkiem pasują do danych.
2. Proste drzewa nie powinny zbytnio dopasować się do danych.
3. Przetrenowanie modelu dla zbyt złożonych drzew, zła generalizacja.

Ale:

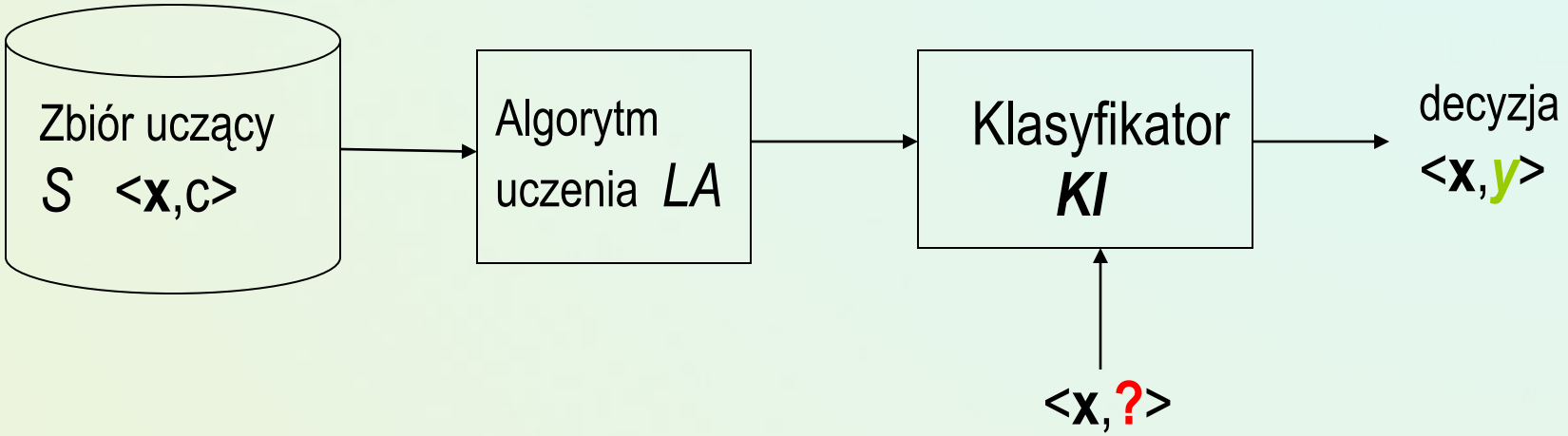
1. Dla małych zbiorów o wielu atrybutach można tworzyć wiele prostych opisów danych.

# Ze źródeł lit.…

- Brzytwa Ockhama – wprowadzona przez teologa franciszkańskiego Williama Ockhama (ok. 1285-1349) zasada:
  istnień nie należy mnożyć ponad potrzebę (łac. Non sunt multiplicanda entia sine necessitate), tłumaczona także tradycyjnie jako:
  „Bytów nie mnożyć, fikcyj nie tworzyć, tłumaczyć fakty jak najprościej.”
- W praktyce tłumaczy się to jako: proste rozwiązanie jest najlepsze albo nie wymyślaj nowych czynników jeżeli nie istnieje taka potrzeba, a jeżeli już, to udowodnij najpierw ich istnienie.

# Predykcja nowych faktów - klasyfikatory

- Poszukiwanie reprezentacji wiedzy o przydziale obiektów do klas na podstawie opisu obiektów za pomocą wartości atrybutów (zbiór uczący).
- Predykcja klasyfikacji nowych obiektów (zbiór testowy)




Przykłady $S = \{\langle \mathbf{x}_1, c_1 \rangle, \langle \mathbf{x}_2, c_2 \rangle, \cdots, \langle \mathbf{x}_n, c_n \rangle\}$

$\mathbf{x}_i = \langle x_{i1}, x_{i2}, \ldots, x_{im} \rangle$ opisywane przez *m* atrybutów

Atrybuty różnego typu

$c_i$ – etykieta jednej z klas $\{C_1, \ldots, C_K\}$

Miara oceny, np:

***trafność klasyfikowania***

$$\eta = \frac{N_c}{N_t}$$

Eksperymentalna ocena

→ *Cross validation*

# Trafność klasyfikowania

- Użyj przykładów testowych nie wykorzystanych w fazie indukcji klasyfikatora:
  - $N_t$ – liczba przykładów testowych
  - $N_c$ – liczba poprawnie sklasyfikowanych przykładów testowych
- Trafność klasyfikowania (ang. classification accuracy):

$$\eta = \frac{N_c}{N_t} \qquad \varepsilon = \frac{N_t - N_c}{N_t}$$

- Alternatywnie błąd klasyfikowania.

# Przetrenowanie / Przeuczenie klasyfikatora

Model *H* jest zbytnio dopasowany do danych (overfits) gdy:

Istnieje model *H'* taki, że:

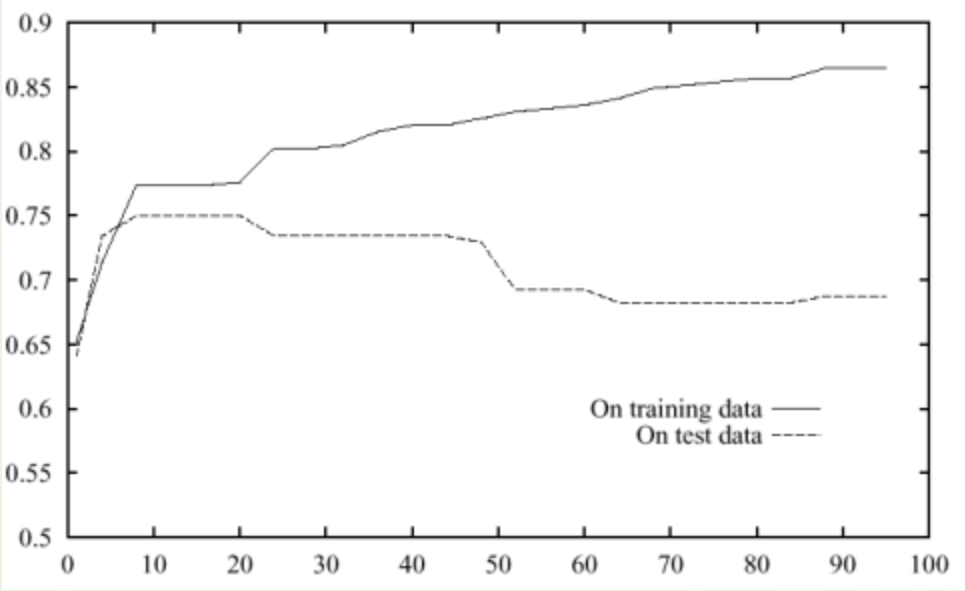
Błąd-treningowy(*H*) < Błąd-treningowy(*H'*)

Błąd-testowy(*H*) > Błąd-testowy(*H'*)

Zbyt szczegółowe wnioski przy dla danej populacji przypadków treningowych.

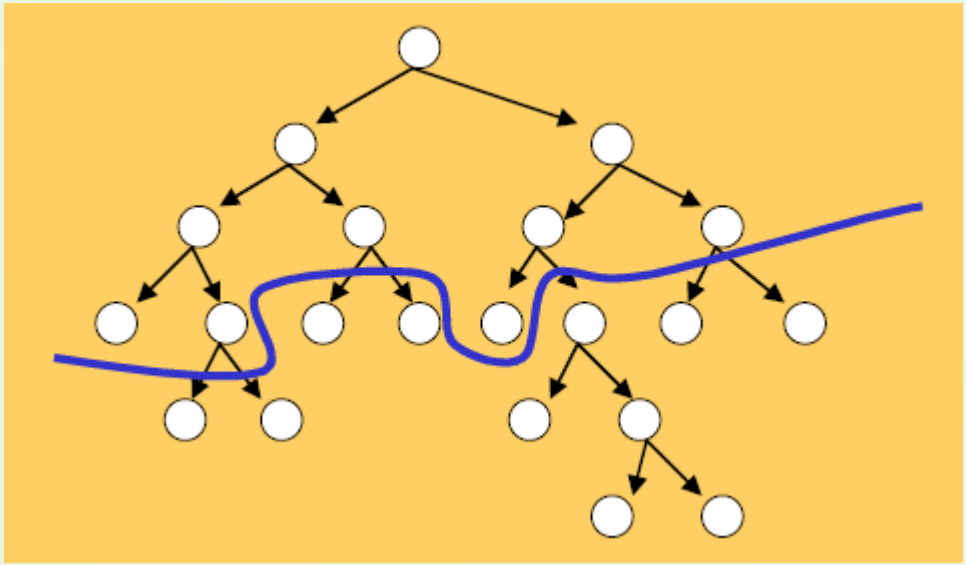
Dokładność jako funkcja liczby węzłów drzewa.

Wyniki mogą być gorsze niż dla klasyfikatora większościowego!



# Tree pruning – upraszczanie drzew

- Unikanie przetrenowanie / nadmiernej specjalizacji
- Po upraszczaniu wzrost trafności klasyfikowania!



# Avoid Overfitting in Classification - Pruning

- Two approaches to avoid overfitting:
  - (Stop earlier / Forward pruning): Stop growing the tree earlier – extra stopping conditions, e.g.
    1. Stop splitting the nodes if the number of samples is too small to make reliable decisions.
    2. Stop if the proportion of samples from a single class (node purity) is larger than a given threshold - forward pruning
  - (Post-pruning): Allow overfit and then post-prune the tree.
    - Estimation of errors and tree size to decide which sub-tree should be pruned.

# Unikanie przetrenowania

Jak uniknąć przetrenowania i radzić sobie z szumem?

1. Zakończ rozwijanie węzła jeśli jest zbyt mało danych by wiarygodnie dokonać podziału.
2. Zakończ jeśli czystość węzłów (dominacja jednej klasy) jest większa od zadanego progu – pre- pruning
   DT => drzewo prawd. klas.
3. Utwórz drzewo a potem je przytnij (post - pruning)

1. Przycinaj korzystając z wyników dla k-cv (CART) lub **dla zbioru walidacyjnego.**
2. Korzystaj z MDL (Minimum Description Length):
   Min Rozmiar(Drzewa) + Rozmiar(Drzewa(Błędów))
3. Oceniaj podziały zaglądając poziom (lub więcej) w głąb.

# Kryteria dla pre-pruning

- The number of cases in the node is less than the given threshold.
- The probability of predicting the strongest class in the node is sufficiently high.
- The best splitting criterion is not greater than a certain threshold.
- The change of probability distribution is not significant.
  - Stop growing the tree when there is no *statistically significant* association between any attribute and the class at a particular node
  - Most popular test: *chi-squared test*
  - Only statistically significant attributes were allowed to be selected by information gain procedure
- …

# Uwagi nt. pre-pruning (2)

It seems to be right but remember about …

- hard to properly evaluate node split without seeing what splits would follow it (use a lookahead technique?)
- some attributes useful only in combination with other attributes

On the other hand

It is less computationally expensive than post-pruning!

# Reduced Error Pruning

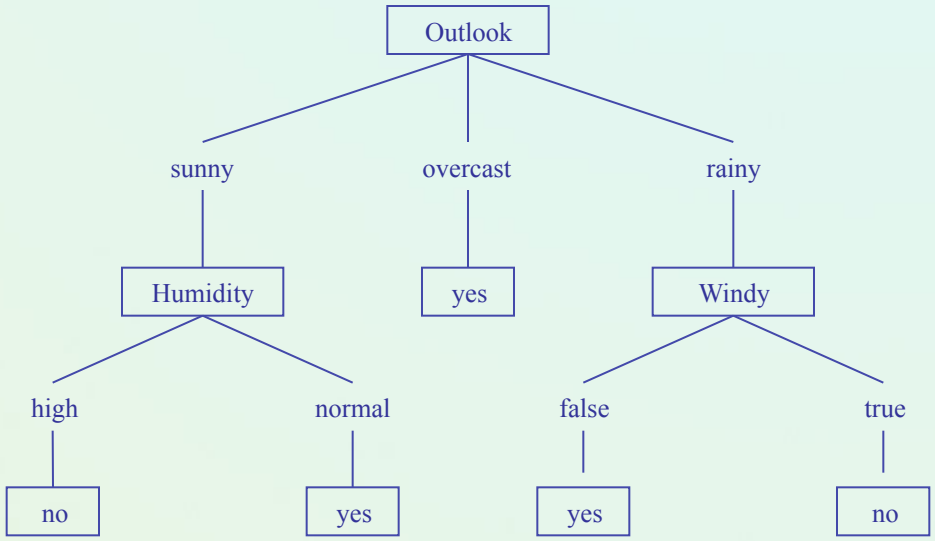
**Split data into training and validation sets.**

**Pruning a decision node *d* consists of:**

1. removing the subtree rooted at *d*.
2. making *d* a leaf node.
3. assigning *d* the most common classification of the training instances associated with *d*.

**Do until further pruning is harmful:**

1. Evaluate impact on validation set of pruning each possible node (plus those below it).
2. Greedily remove the one that most improves validation set accuracy.



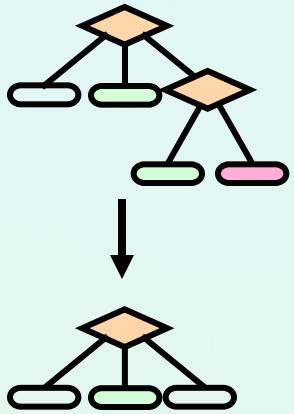
# Cost-complexity approach

- Consider both size and error estimate of the tree.

$$\alpha \cdot R(t) + e(t)$$

- where $t$ – the current tree, $R(t)$ – size of the tree, $e(t)$ – estimation of the error while classifying objects, $\alpha$ - technical coefficient.
- Try to minimize it!
- A strategy to look through a family of reduced trees.
- MDL principle - …

# Reduced-Error Pruning



- Post-Pruning, Cross-Validation Approach
- Split Data into Training and Validation Sets
- Function *Prune*(*T, node*)
  - Remove the subtree rooted at *node*
  - Make *node* a leaf (with majority label of associated examples)
- Algorithm *Reduced-Error-Pruning* (*D*)
  - Partition *D* into $D_{train}$ (training / "growing"), $D_{validation}$ (validation / "pruning")
  - Build complete tree *T* using *ID3* on $D_{train}$
  - UNTIL accuracy on $D_{validation}$ decreases DO

    FOR each non-leaf node *candidate* in *T*

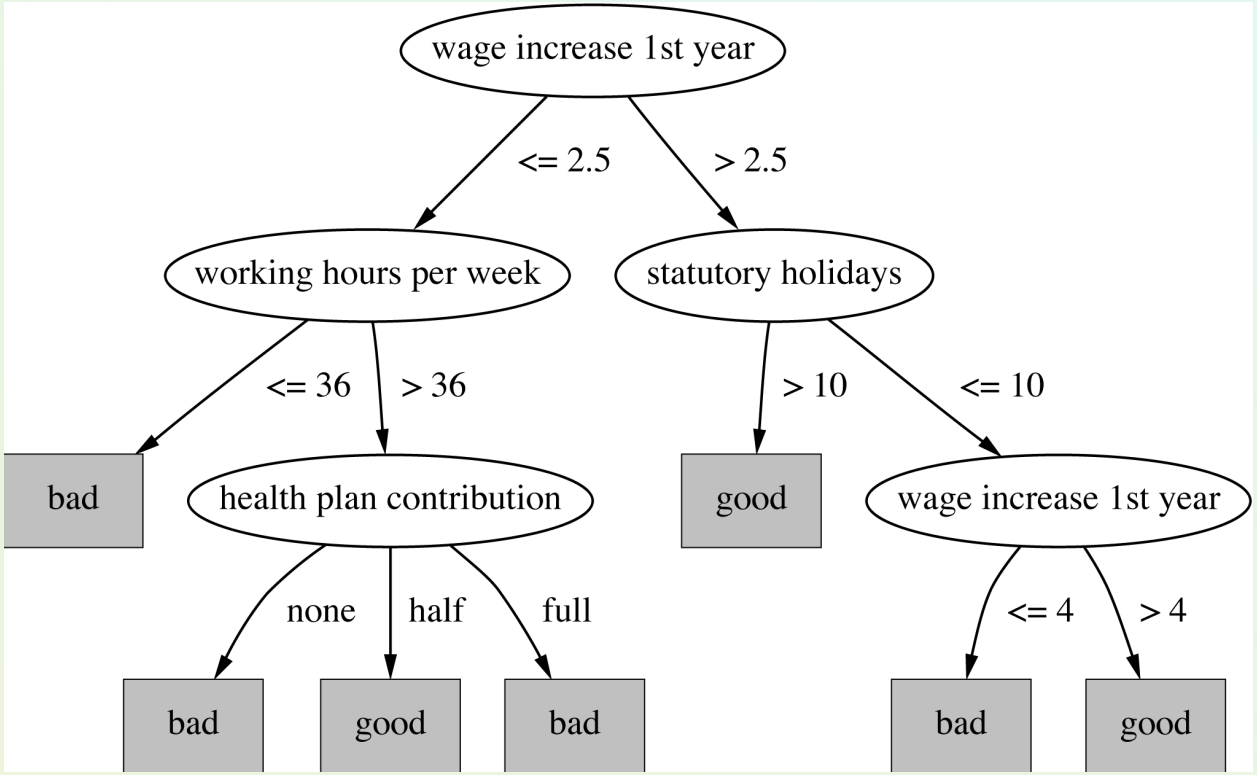
    $\textbf{\textit{Temp}[candidate]} \leftarrow \textbf{\textit{Prune}}\ (\textbf{\textit{T, candidate}})$

    $\textbf{\textit{Accuracy}[candidate]} \leftarrow \textbf{\textit{Test}}\ (\textbf{\textit{Temp}[candidate]}, D_{validation})$

    $T \leftarrow T' \in Temp$ with best value of *Accuracy* (best increase; *greedy*)
  - RETURN (pruned) *T*

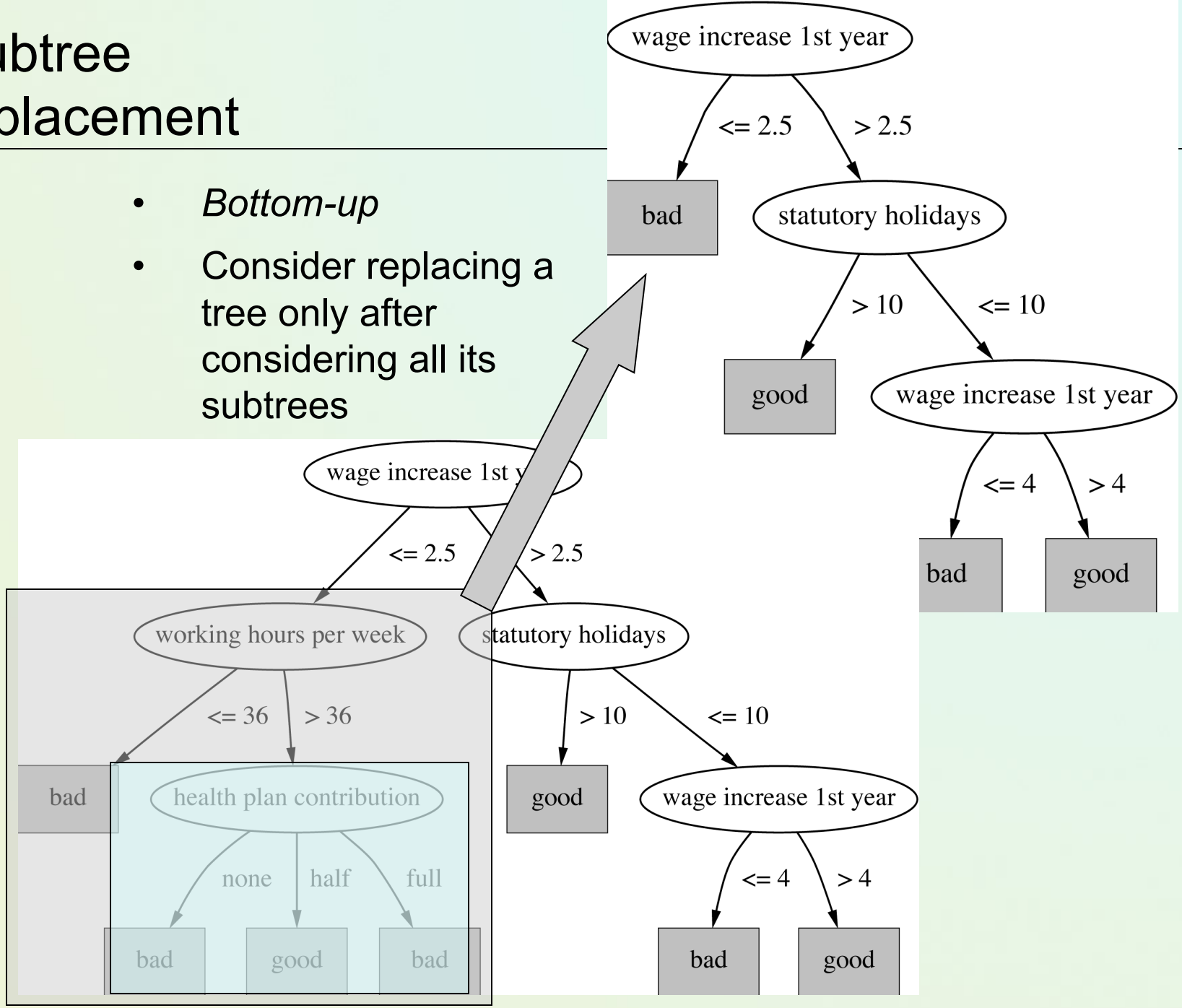
# Post-pruning

- *Bottom-up*
- Consider replacing a tree only after considering all its subtrees
- Ex: labor negotiations



# Subtree replacement

- *Bottom-up*
- Consider replacing a tree only after considering all its subtrees

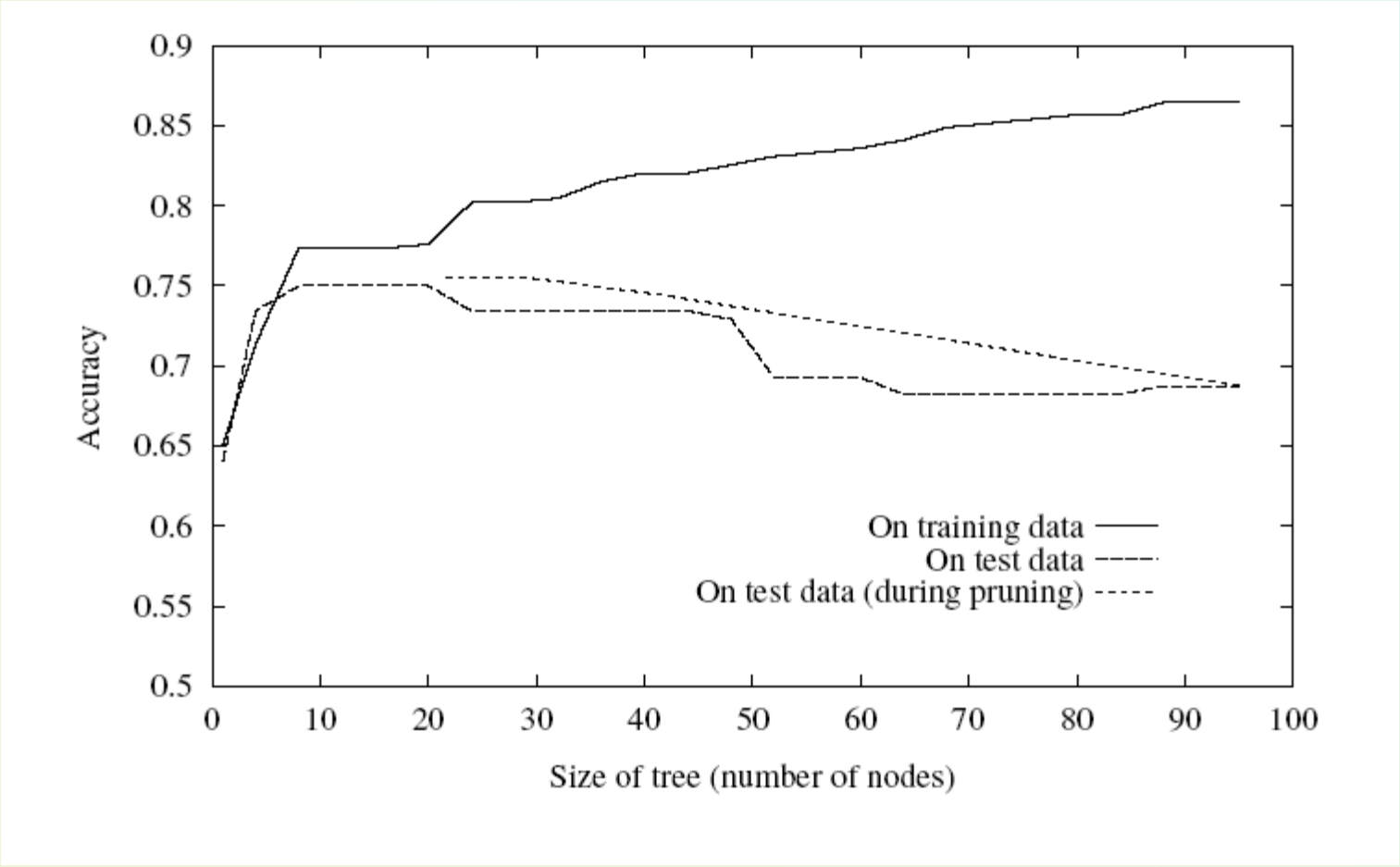


# Remarks to post-pruning

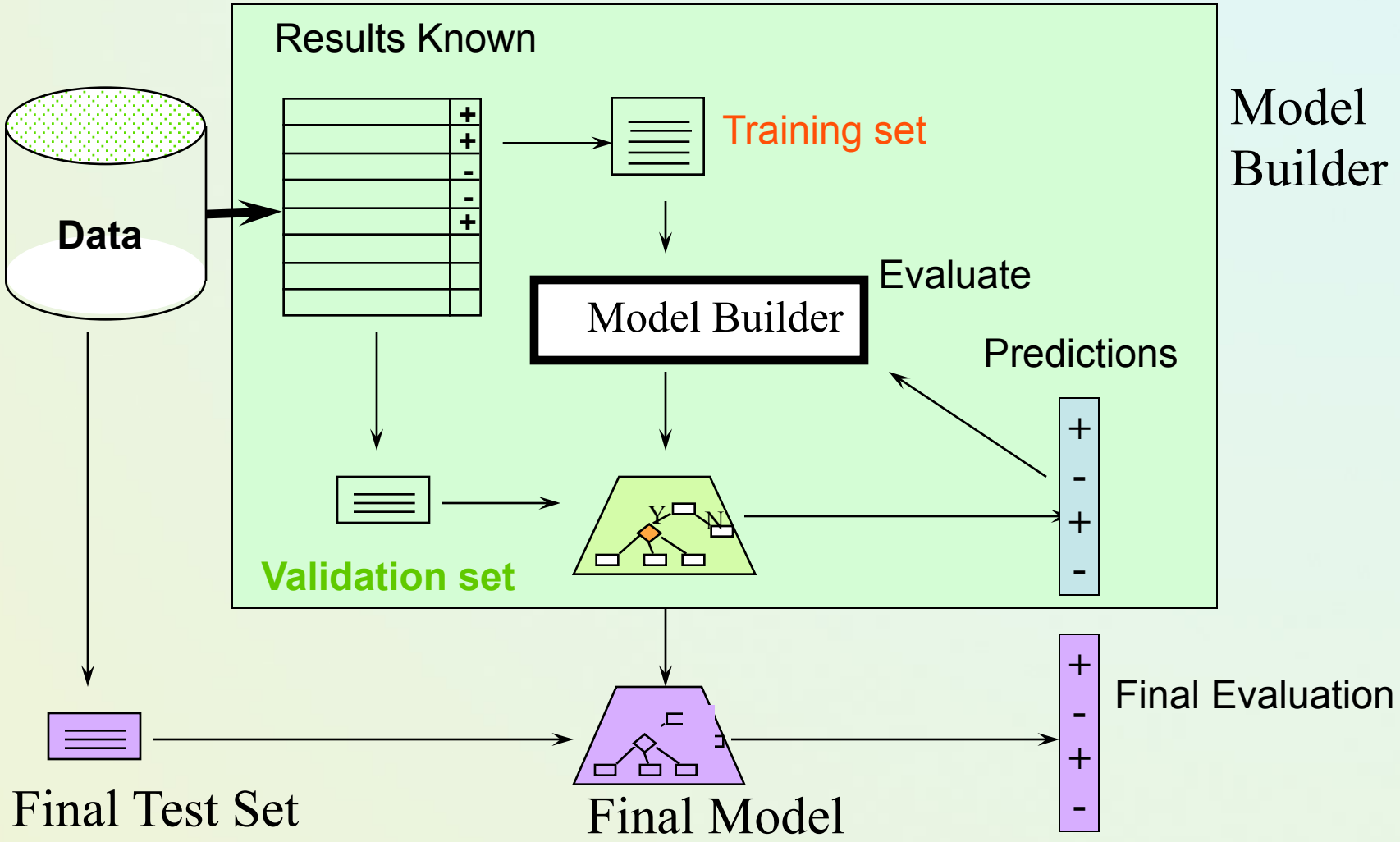
- Approaches to determine the correct final tree size:
  - Different approaches to error estimates
  - Separate training and testing sets or use cross-validation.
  - Use all the data for training, but apply a statistical test to estimate whether expanding or pruning a node may improve over entire distribution.
- Rule post-pruning (C4.5): converting to rules before pruning.
- C4.5 method – estimate of pessimistic error
  - Derive confidence interval from training data
  - Use a heuristic limit, derived from this, for pruning
  - Shaky statistical assumptions (based on training data)
  - Seems to work OK in practice
  - Option *c* parameter – default value 0,25:
    the smaller value, the stronger pruning!

# Reduced post-pruning

- Separate training and testing sets or use an extra validation (pruning one).



# Classification: Train, Validation, Test split



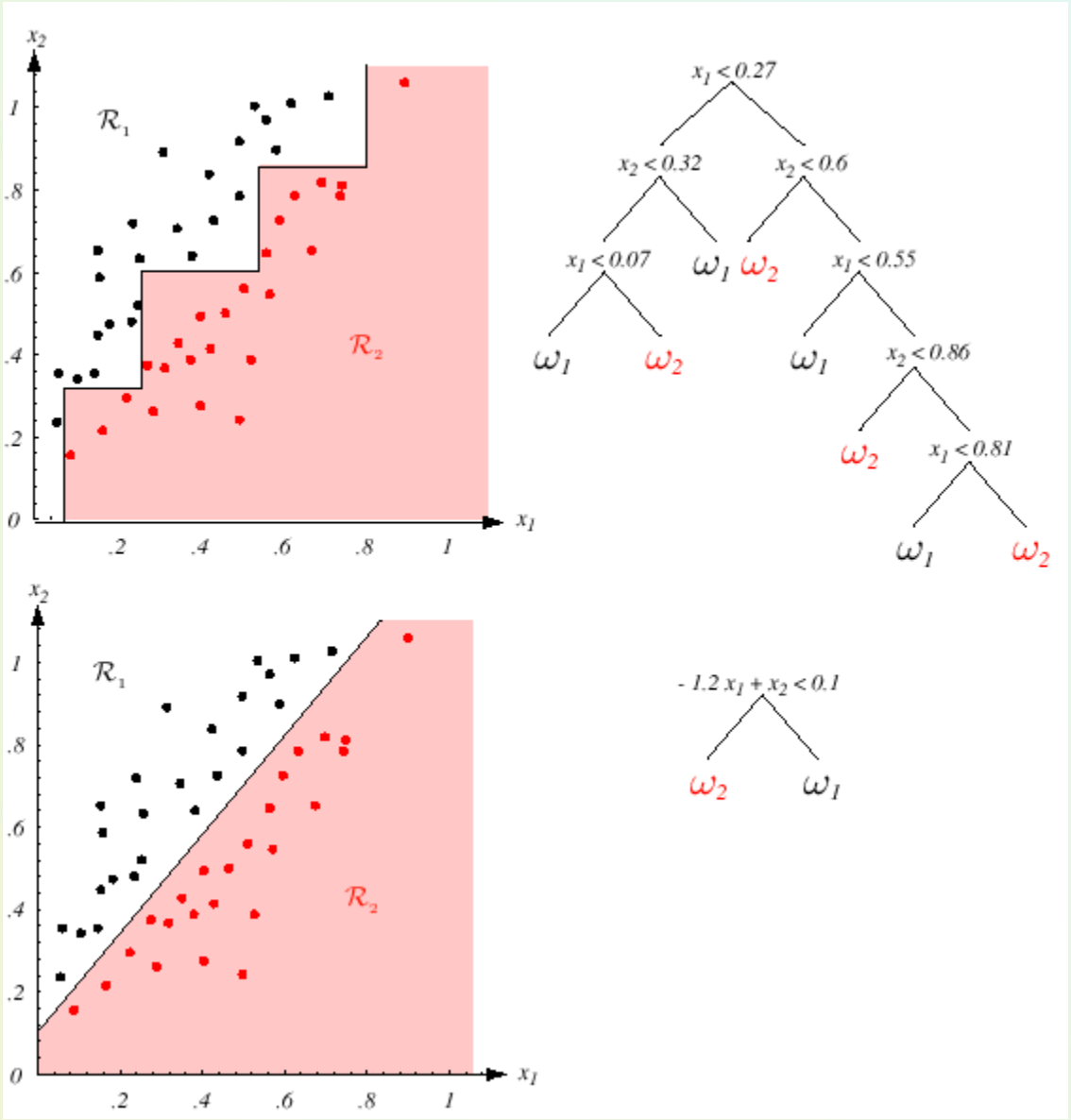
# From trees to rules (C4.5)

- Simple way: one rule for each leaf
- C4.5rules: greedily prune conditions from each rule if this reduces its estimated error
  - Can produce duplicate rules
  - Check for this at the end
- Then
  - look at each class in turn
  - consider the rules for that class
  - find a “good” subset (guided by MDL)
- Then rank the subsets to avoid conflicts
- Finally, remove rules (greedily) if this decreases error on the training data

# *Complexity of tree induction

- Assume
  - $m$ attributes
  - $n$ training instances
  - tree depth $O(\log n)$
- Building a tree $O(m\, n \log n)$
- Subtree replacement $O(n)$
- Subtree raising $O(n (\log n)^2)$
  - Every instance may have to be redistributed at every node between its leaf and the root
  - Cost for redistribution (on average): $O(\log n)$
- Total cost: $O(m\, n \log n) + O(n (\log n)^2)$

# Oblique trees – skośnie warunki




Univariate, or monothetic trees,

mult-variate, or oblique trees.

Figure from Duda, Hart & Stork, Chap. 8

# Software implementations

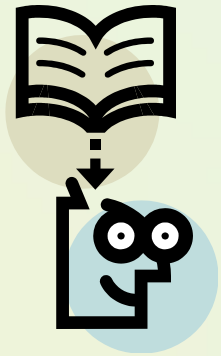
- ID3 and C4.5 was easier to get free (with source code in C)
  - J4.8 available in WEKA
  - The best version C5.0 is a commercial tool www.rulequest.com
- CART → was not so easy to get free
  - Commercialy distributed by Salford Systems
    - www.salford-systems.com
  - Basis versions available in typical statistical / data mining software as Statsoft, SAS, SPSS-IBM, etc.
- Many others – see W.Buntine IND2

# Scaling Up

- ID3, C4.5, etc. assume data fits in main memory
  (OK for up to hundreds of thousands of examples)
- SPRINT, SLIQ: multiple sequential scans of data
  (OK for up to millions of examples)
- VFDT: at most one sequential scan
  (OK for up to billions of examples)

# Trochę książek

- Uczenie maszynowe i sieci neuronowe. Krawiec K., Stefanowski J., Wydawnictwo Politechniki Poznańskiej, Poznań, 2003 (kolejne wydanie 2004)
- Systemy uczące się. Cichosz P., WNT, Warszawa, 2000
- Statystyczne systemy uczące się. Koronacki J., Ćwik J. WNT Warszawa 2008
- Oraz czytamy książki anglojęzyczne



# Pytanie i komentarze?

# Dziękuję za uwagę – lecz czytajcie więcej