Indukcja drzew

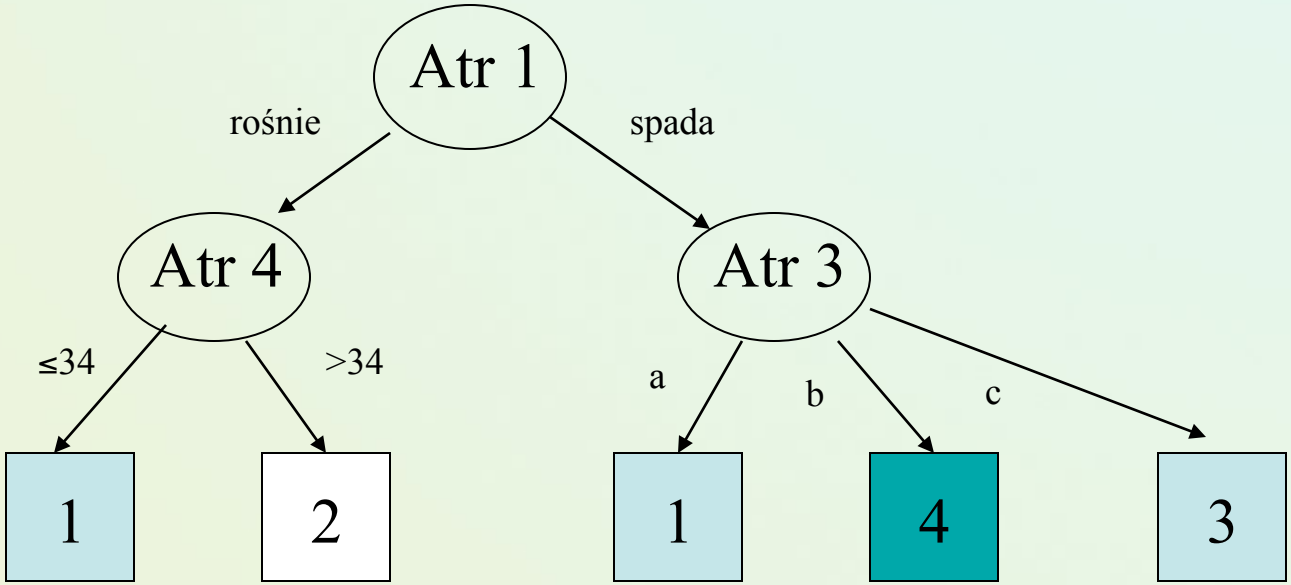
JERZY STEFANOWSKI
Instytut Informatyki
Politechnika Poznańska

Uwagi do wykładu dla Ucz. Maszynowe
Aktualizacja 2016 i 2019

Co to jest drzewo decyzyjne?

Jest to struktura grafu skierowanego z góry na dół:

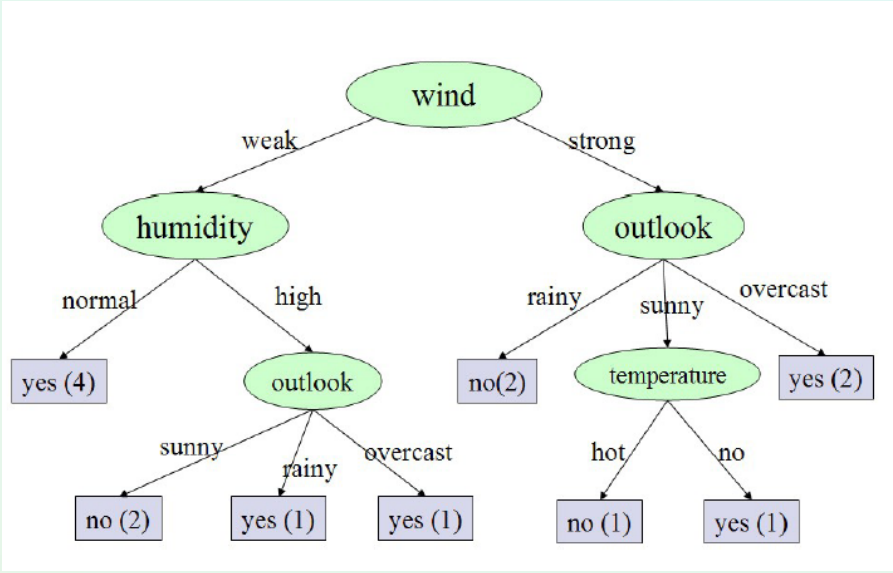
- Węzły reprezentują pytanie o wartości cech
- Z węzłów wychodzą gałęzie które reprezentują wynik pytania
- Liście reprezentują klasy decyzyjne

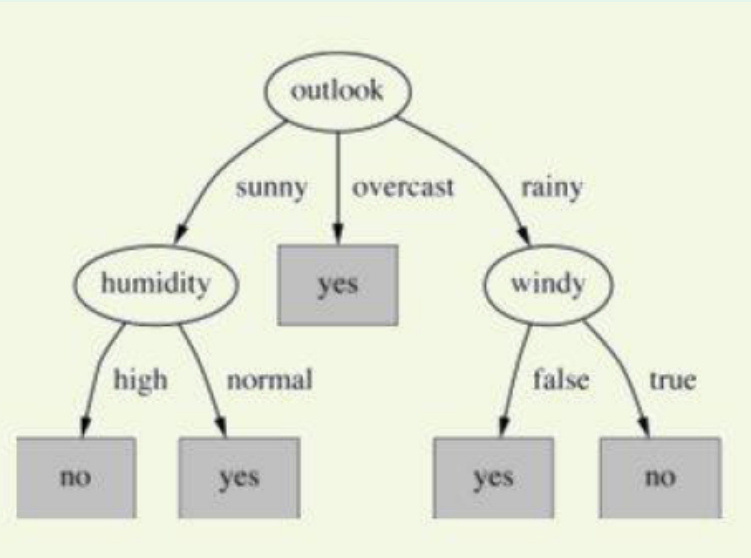


Poszukiwanie dobrych drzew

Play or not (Quinlan)

x	outlook	Temperature	humidity	wind	play(x)
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cold	normal	weak	yes
6	rain	cold	normal	strong	no
7	overcast	cold	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cold	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

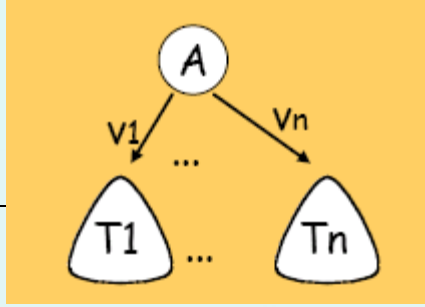




Jak poszukiwać drzew?

- Rozważ wszystkie możliwe testy
- Duża liczba hipotez, dla N binarnych atrybutów:
  - 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
  - i dalej, …
- Przestrzeń przeszukiwania rośnie wykładniczo od rozmiaru liczby atrybutów
  - Podejścia heurystyczne

Metody indukcji drzew decyzyjnych



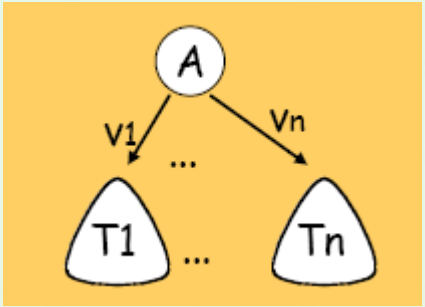
- Podejście obejmuje dwa etapy:
  - Konstrukcja drzewa (rekurencyjna procedura)
    - Na początku wszystkie przykłady w węźle.
    - Rekurencyjnie dziel przykłady w oparciu o wybrane testy na wartościach atrybutu (kryterium wyboru najlepszego atrybutu).
    - Zatrzymaj gdy wszystkie przykłady „w gałęzi” należą do jednej klasy
  - Upraszczanie drzewa - „Tree pruning”
    - Usuwanie poddrzew, które mogą prowadzić do błędnych decyzji podczas klasyfikacji przypadków testowych.
  - Przykłady algorytmów: ID3, C4.5, CART,…

Ogólny schemat ID3

TDIDT - Top Down Iterative Decision Tree

```
function DT(E: zbiór przykładów) returns drzewo;
    T' := buduj_drzewo(E);
    T := obetnij_drzewo(T');
    return T;
```

```
function buduj_drzewo(E: zbiór przyk.) returns drzewo;
        T := generuj_tests_atr_A(E);
        t := najlepszy_test(T, E);
        P := podział E indukowany przez t;
        if kryterium_stopu(E, P)
        then return liść(info(E))
        else
                for all E_j in P: t_j := buduj_drzewo(E_j);
                return węzeł(t, {(j,t_j)};
```

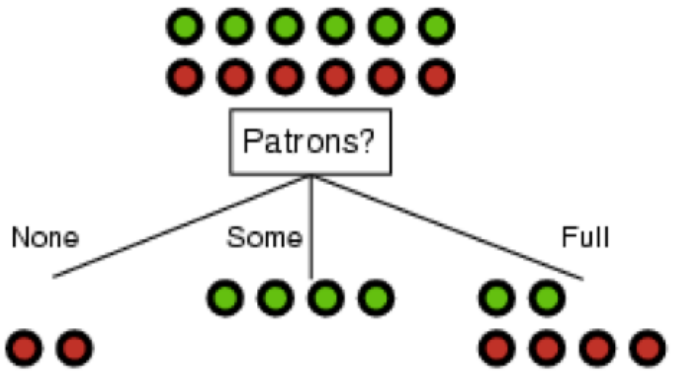


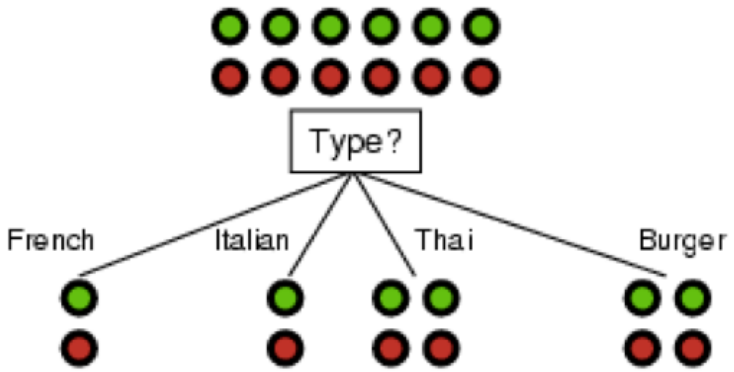
Intuicja wyboru atrybutu

Przykład decyzji o wyborze restauracji [Rusell, Norvig]

Split condition -

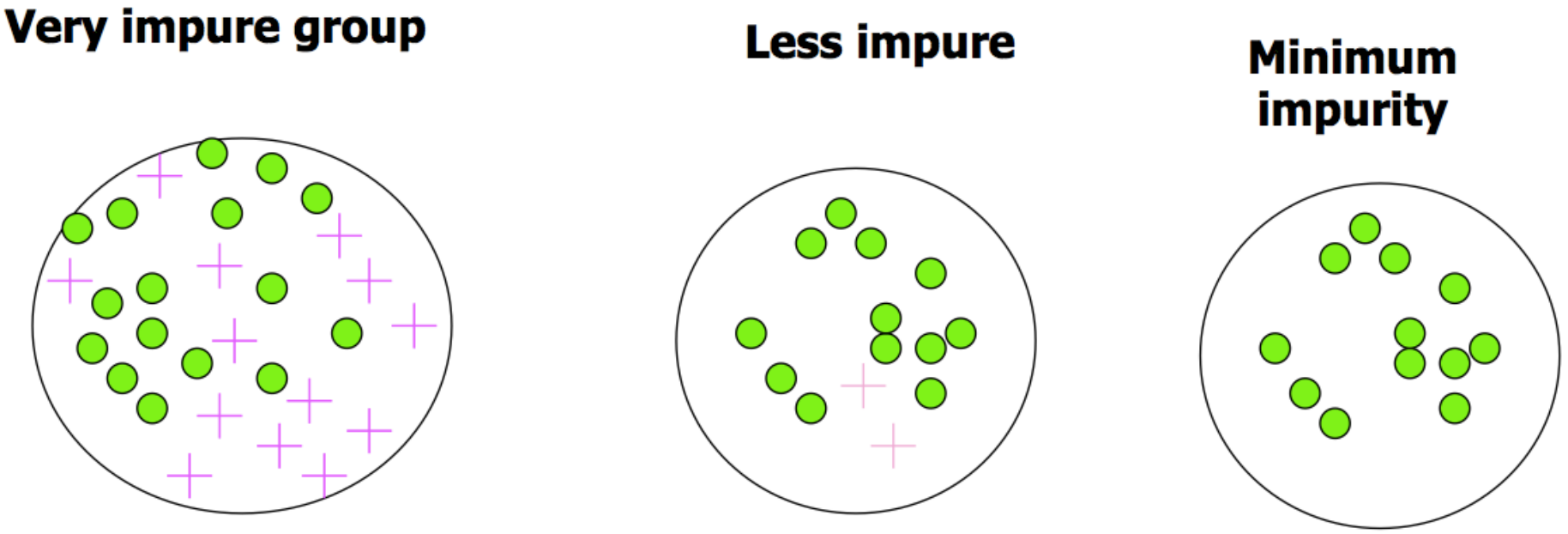
Dobry atrybut powinien podzielić zbiór przykładów S na podzbiory S1, S2,…, które są możliwie jednoznaczne (purity) wskazać klasy decyzyjne – poszukiwanie możliwie najprostszego drzewa zgodnego z przykładami uczącymi





Which split is more informative: *Patrons?* or *Type?*

Impurity functions



Intuicja funkcji “impurity class assignments”

Za wykładem Eric Eaton

Oczekiwania wobec funkcji wyboru atrybutu

Impurity functions:

- Given a random variable *x* with *k* discrete values, distributed according to P={p1,p2,…pk}, a impurity function Φ should satisfies:
  - $\Phi(P) \geq 0$; $\Phi(P)$ is minimal if $\exists i$ such that pi=1;
    $\Phi(P)$ is maximal if $\forall i\ 1 \leq i \leq k$, pi=1/k
    $\Phi(P)$ is symmetrical and differentiable everywhere in its range
- The goodness of split is a reduction in impurity of the target concept after partitioning S.
- Popular function: *information gain*
  - Information gain increases with the average purity of the subsets that an attribute produces

Wybór atrybutu

Który atrybut może tworzyć dobry podział

p_+ i p_- - proporcje w lewej i prawej gałęzi.

Zbiór przykładów *S*

Ile informacji zawiera dany podział ?
Średnia l. bitów do zakodowania dowolnego przykładu z *S* wynosi:

$$\text{entropy}(S) = -p_1 \log p_1 - p_2 \log p_2 \ldots - p_n \log p_n$$

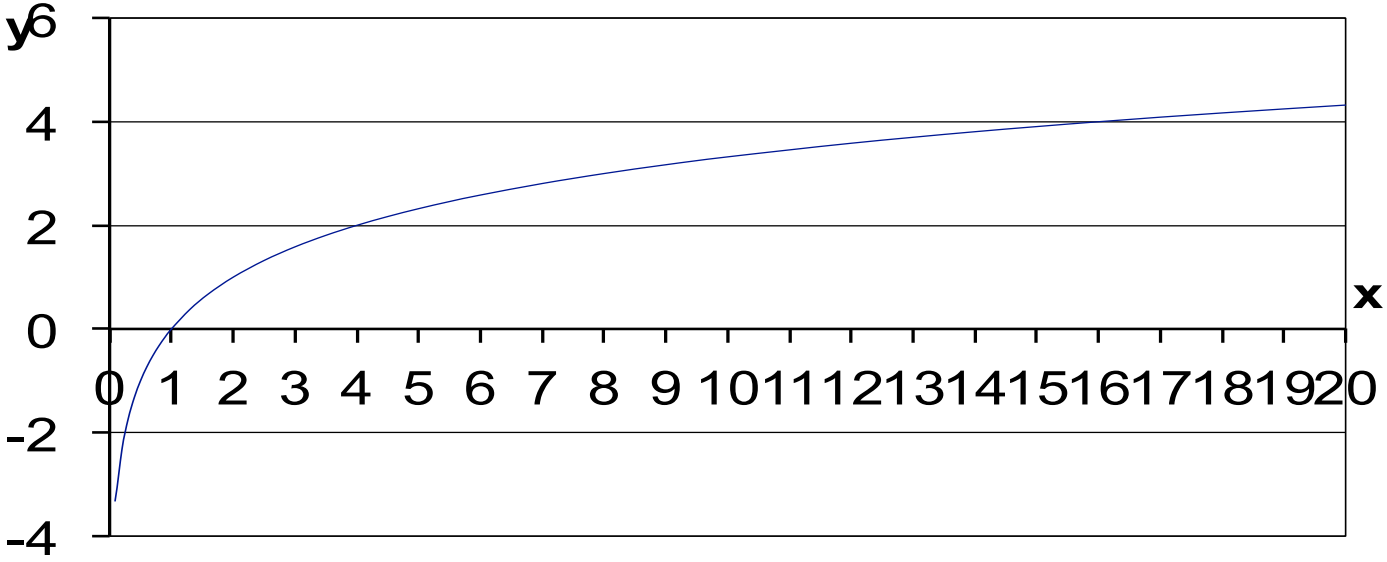
$$entropy(S \mid A) = \sum_{i=1}^{m} \frac{|S_i|}{|S|} \cdot entropy(S)$$

Informacja dla czystych węzłów = 0;

jest max dla najbardziej pomieszanych.

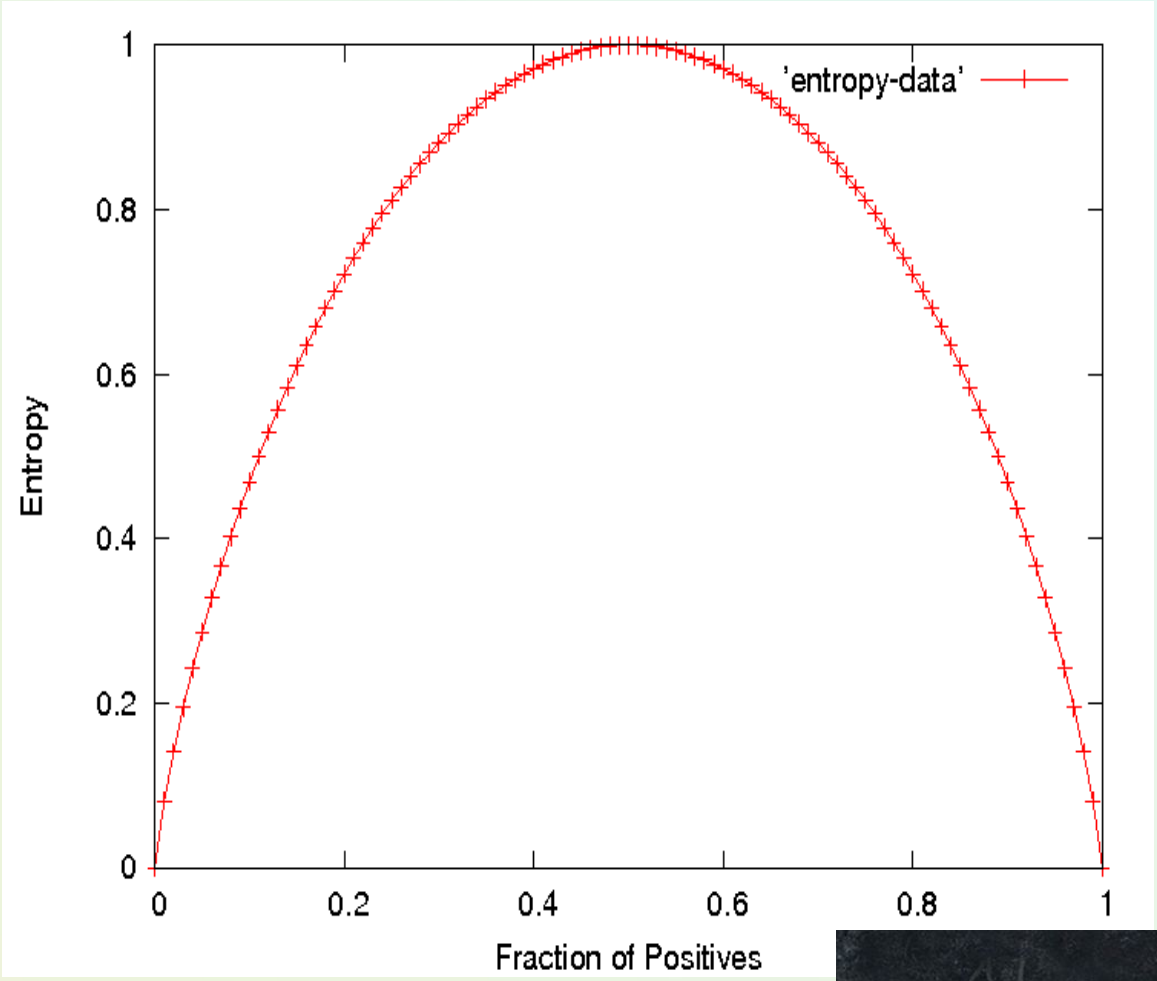
Przypomnienie logarytmów

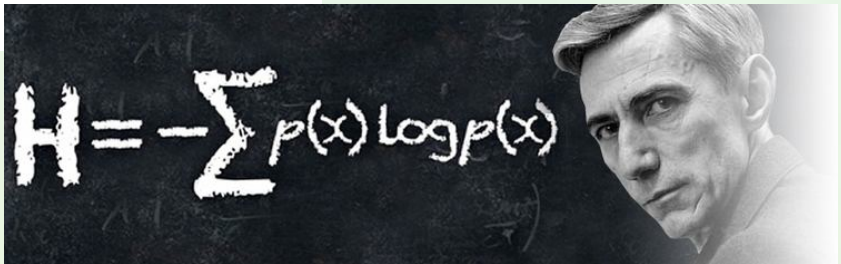
- Funkcja log. $y = \log_a x$
- a – podstawa logarytmu $x = a^y$
- Rozważmy funkcję logarytmiczną dla $a = 2$ (tj. $\log_2 x$)



x	1/8	1/4	1/2	1	2	4	8
y	-3	-2	-1	0	1	2	3

Entropy Plot for Binary Classification





Weather Data: Play or not Play?

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 attributes are nominal

Entropia dla przykładu Quinlana golf

Nie oceniamy podziału atrybutem, tylko rozkład wartości klas decyzyjnych

Dwie klasy : *yes* and *no*

Z 14 przykładów 9 etykietowanych jako *yes*, reszta jako *no*

$$p_{yes} = -\left(\frac{9}{14}\right)\log_2\left(\frac{9}{14}\right) = 0.41$$

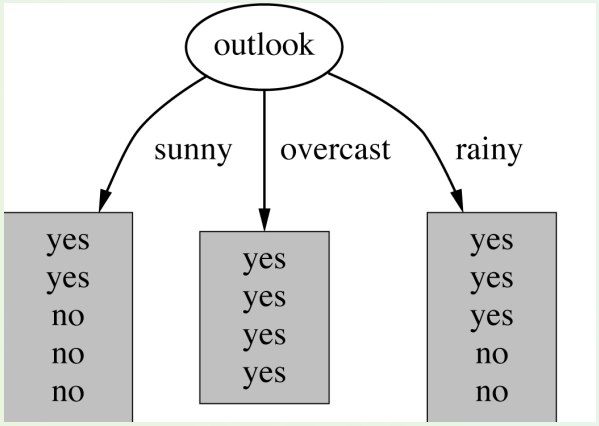
$$p_{no} = -\left(\frac{5}{14}\right)\log_2\left(\frac{5}{14}\right) = 0.53$$

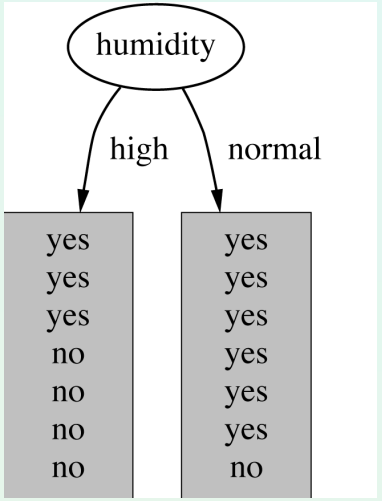
$$E(S) = p_{yes} + p_{no} = 0.94$$

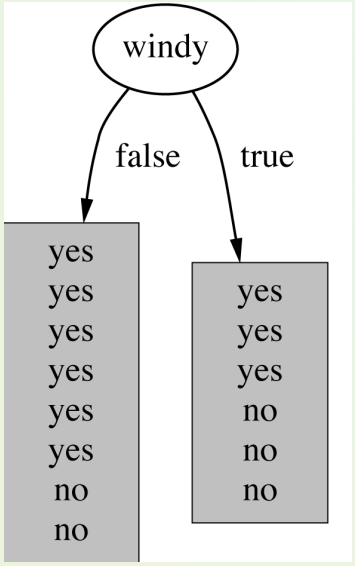
Outlook	Temp.	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes

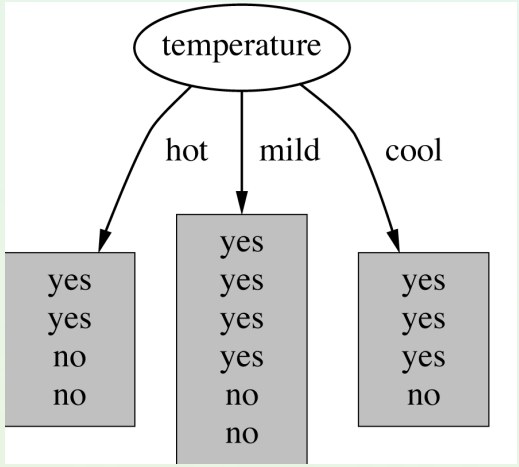
Outlook	Temp.	Humidity	Windy	play
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Korzeń drzewa – który atrybut









witten&eibe

Information gain

- Entropia warunkowa: entropia po podziale zbioru przykładów przy pomocy atrybutu A (załóżmy że A przyjmuje *v* możliwych wartości):

$$EntropiaWarunkowa(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

- Zysk informacyjny (*Information Gain*): redukcja entropii przy wykorzystaniu danego atrybutu:

$$IG(A) = I - EntropiaWarunkowa(A)$$

Wybierz atrybut "Outlook"

- "Outlook" = "Sunny":

$$\text{info}([2,3]) = \text{entropy}(2/5,3/5) = -2/5\log(2/5) - 3/5\log(3/5) = 0.971$$

- "Outlook" = "Overcast":

$$\text{info}([4,0]) = \text{entropy}(1,0) = -1\log(1) - 0\log(0) = 0$$

*Note: log(0) is not defined, but we evaluate 0*log(0) as zero*

- "Outlook" = "Rainy":

$$\text{info}([3,2]) = \text{entropy}(3/5,2/5) = -3/5\log(3/5) - 2/5\log(2/5) = 0.971$$

- Expected information for attribute:

$$\text{info}([3,2],[4,0],[3,2]) = (5/14)\times 0.971 + (4/14)\times 0 + (5/14)\times 0.971$$

$$= 0.693$$

Oblicz zysk informatyczny dla każdego atrybutu

- Information gain:

(information before split) – (information after split)

$$\text{gain("Outlook")} = \text{info}([9,5]) - \text{info}([2,3],[4,0],[3,2]) = 0.940 - 0.693 = 0.247$$

- Information gain for attributes from weather data:

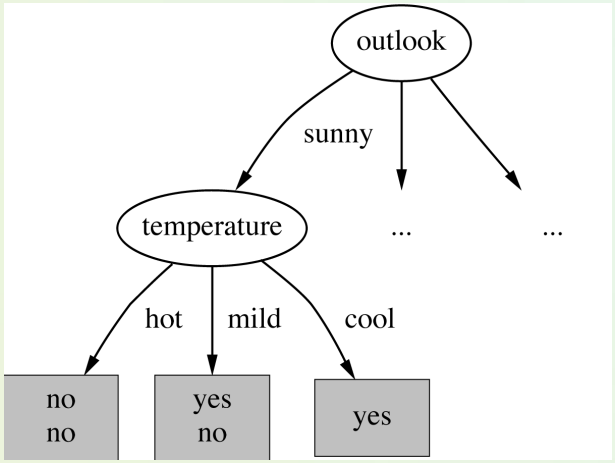
$$\text{gain("Outlook")} = 0.247$$

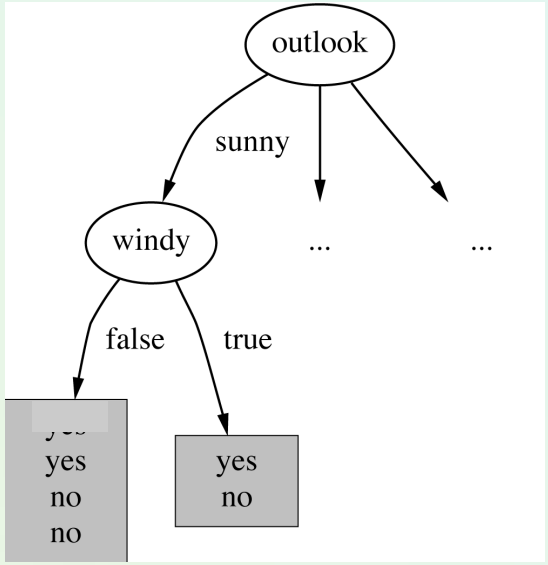
$$\text{gain("Temperature")} = 0.029$$

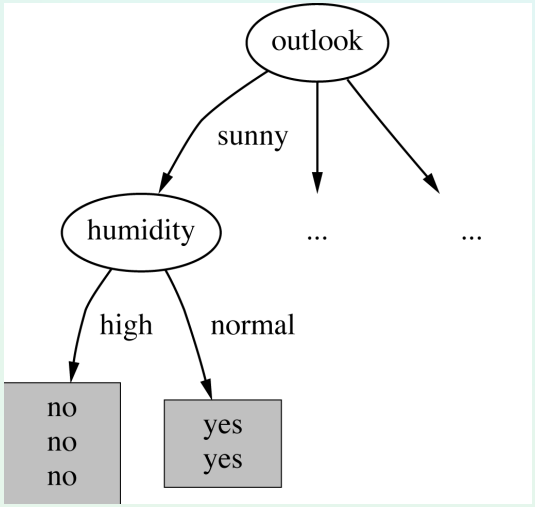
$$\text{gain("Humidity")} = 0.152$$

$$\text{gain("Windy")} = 0.048$$

Rozbuduj drzewo





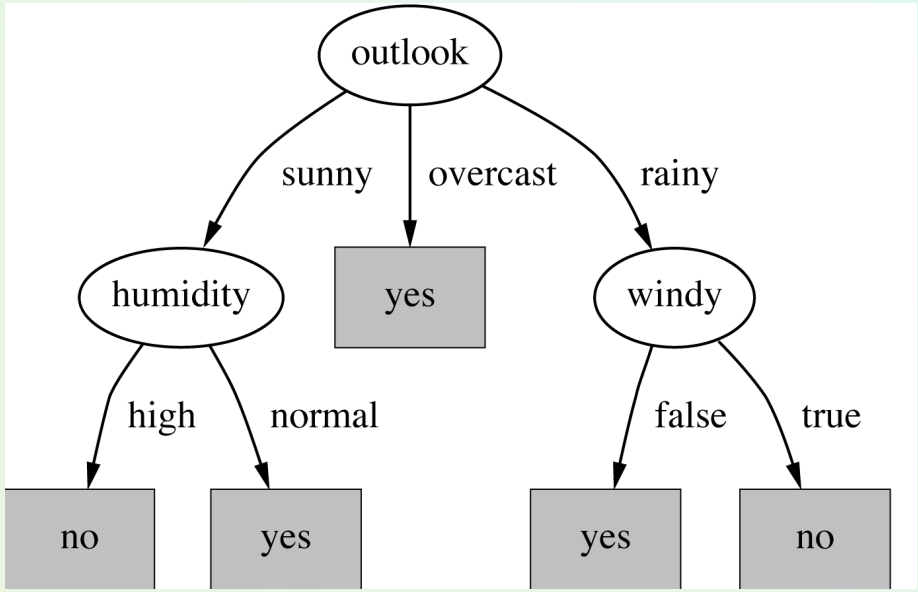


gain("Humidity") = 0.971

gain("Temperature") = 0.571

gain("Windy") = 0.020

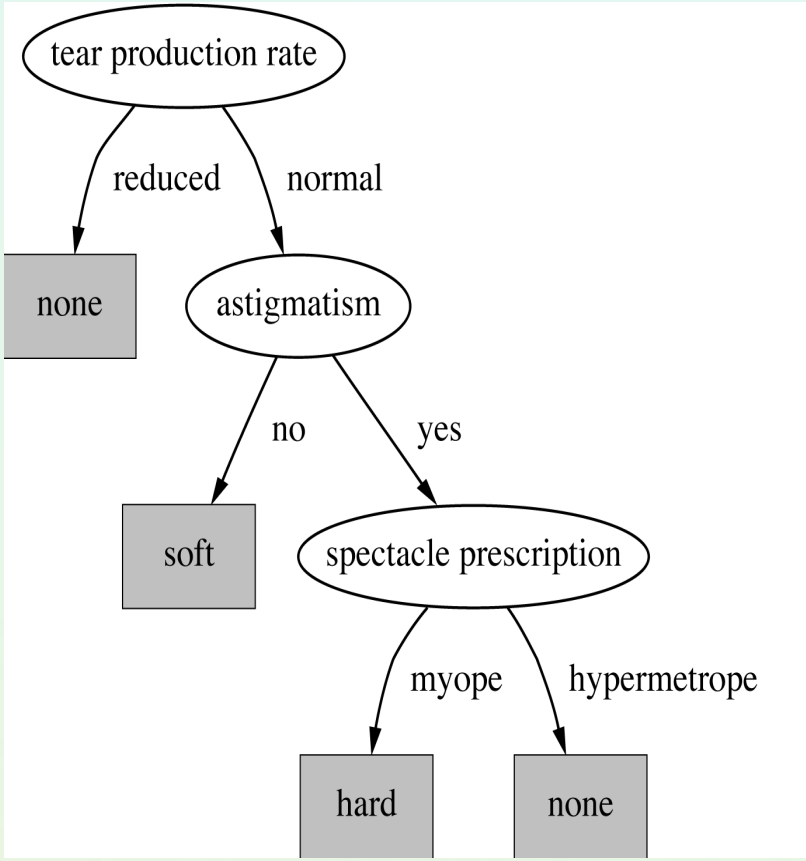
Końcowe drzewo



ID3 - Quinlan

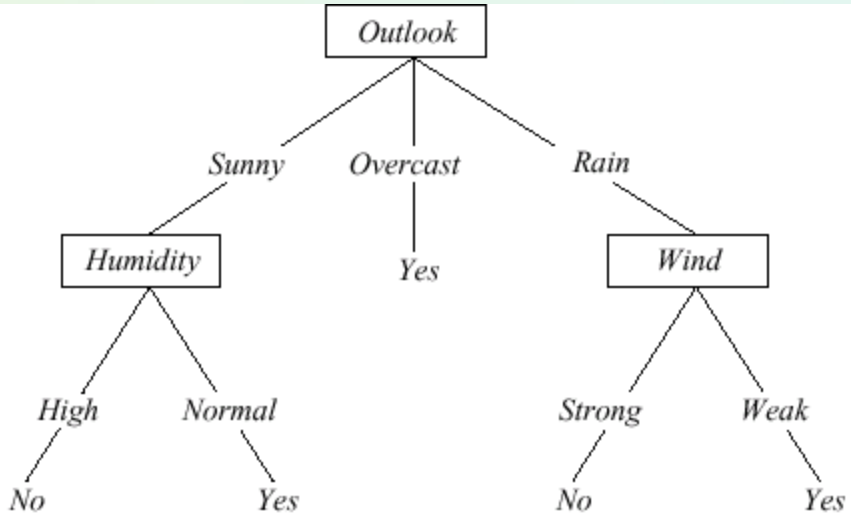
Wykorzystanie drzewa

- Bezpośrednio:
  - sprawdzaj wartości atrybutu nowego przykładu zaczynając od korzenia do liści
- Pośrednio:
  - zamień strukturę drzewa na zbiór reguł decyzyjnych (upraszczając nadmiarowe warunki)
  - reguły uważa się za czytelniejszą reprezentacje



DT => reguły

Zamień DT na reguły i uprość: łatwo ocenić, które reguły można usunąć i optymalizować pozostałe.



IF $(Outlook = Sunny) \wedge (Humidity = High)$ THEN $PlayTennis = No$

IF $(Outlook = Sunny) \wedge (Humidity = Normal)$ THEN $PlayTennis = Yes$

Zamiana na zbiór reguł klasyfikacyjnych

Poprzedni przykład:

```
If outlook = sunny and humidity = high then play = no
If outlook = rainy and windy = true then play = no
If outlook = overcast then play = yes
If humidity = normal then play = yes
If none of the above then play = yes
```

Lecz pamiętaj:

- Dropping redundant conditions in rules and rule post-pruning
- Classification strategies with rule sets are necessary

Brzytwa Ockhama

Czemu preferować prostsze drzewa?

1. Mało prostych hipotez, 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.

Przetrenowanie

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

Istnieje model *H'* taki, że:

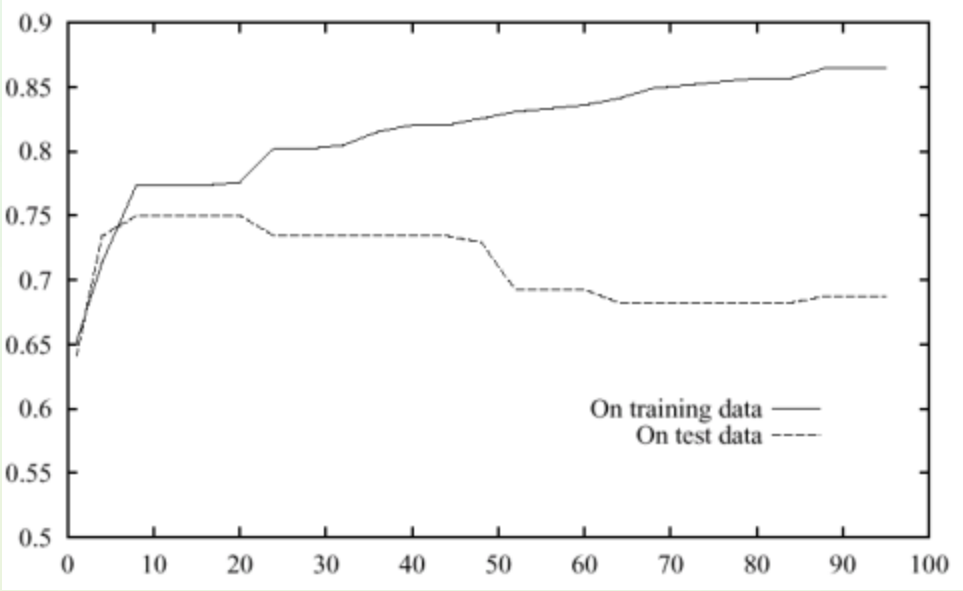
Błąd-treningowy(*H*) < Błąd-treningowy(*H'*)

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

Zbyt szczegółowe wnioski przy dla danej populacji przypadków treningowych.

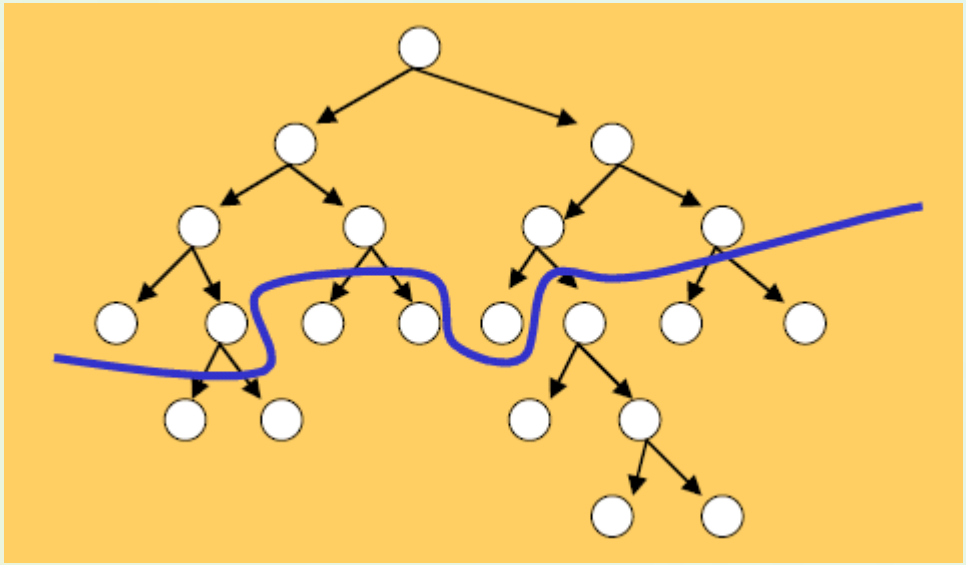
Dokładność jako funkcja liczby węzłów drzewa.

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.

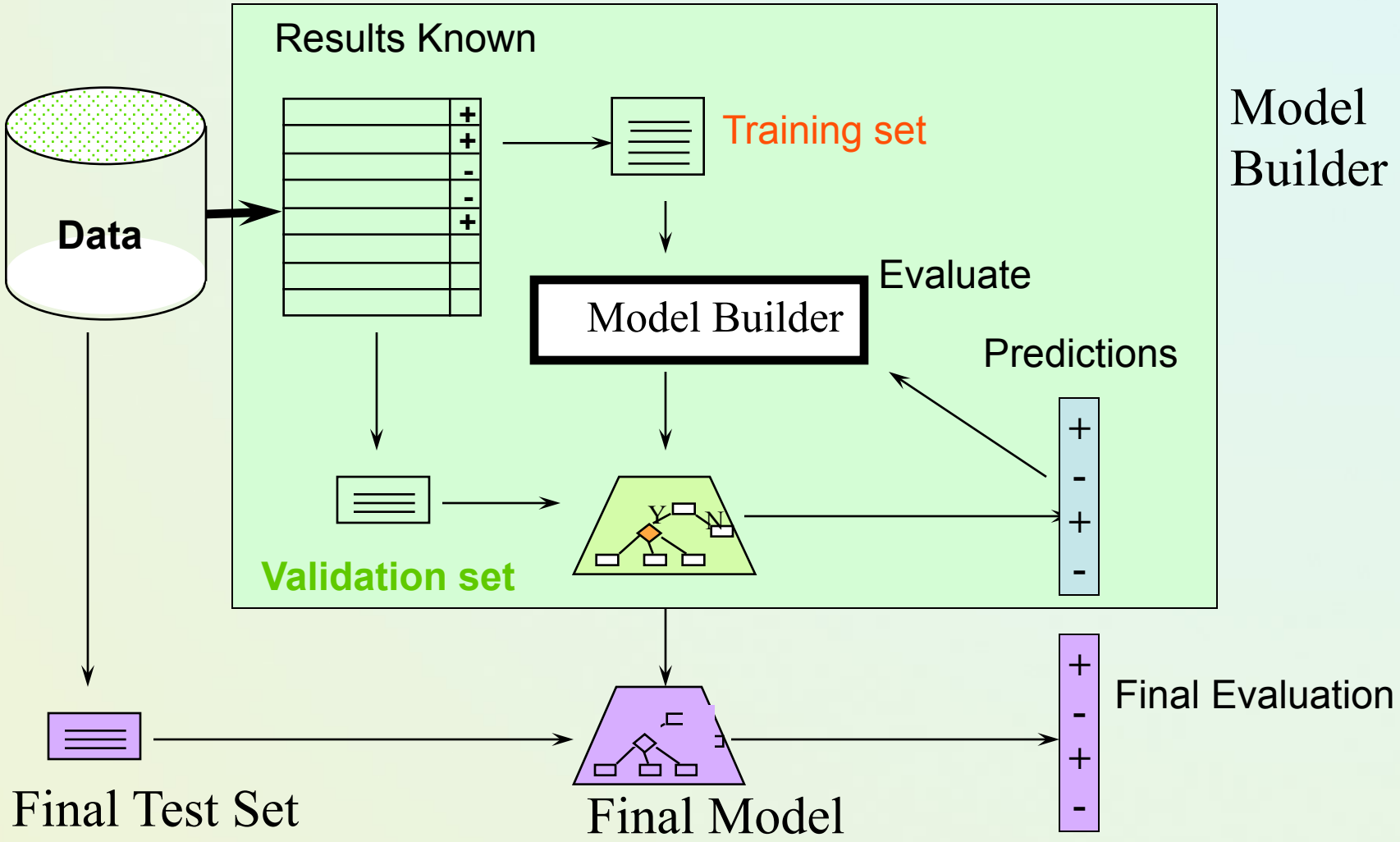
Unikanie przetrenowania

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.

Classification: Train, Validation, Test split



Reduced Error Pruning

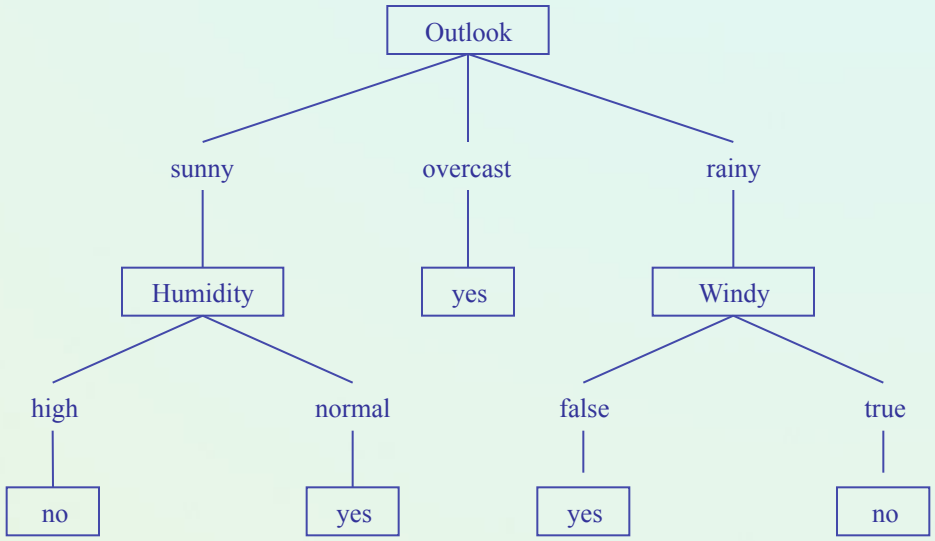
Split data into training and validation sets.

Pruning a decision node *d* consists of:

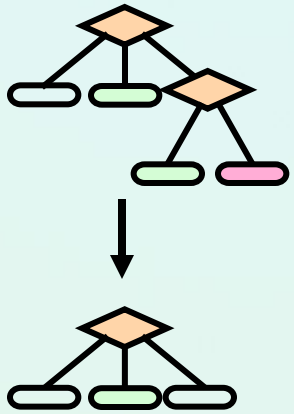
1. removing the subtree rooted at *d*.
2. making *d* a leaf node.
3. assigning *d* the most common classification of the training instances associated with *d*.

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.



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*

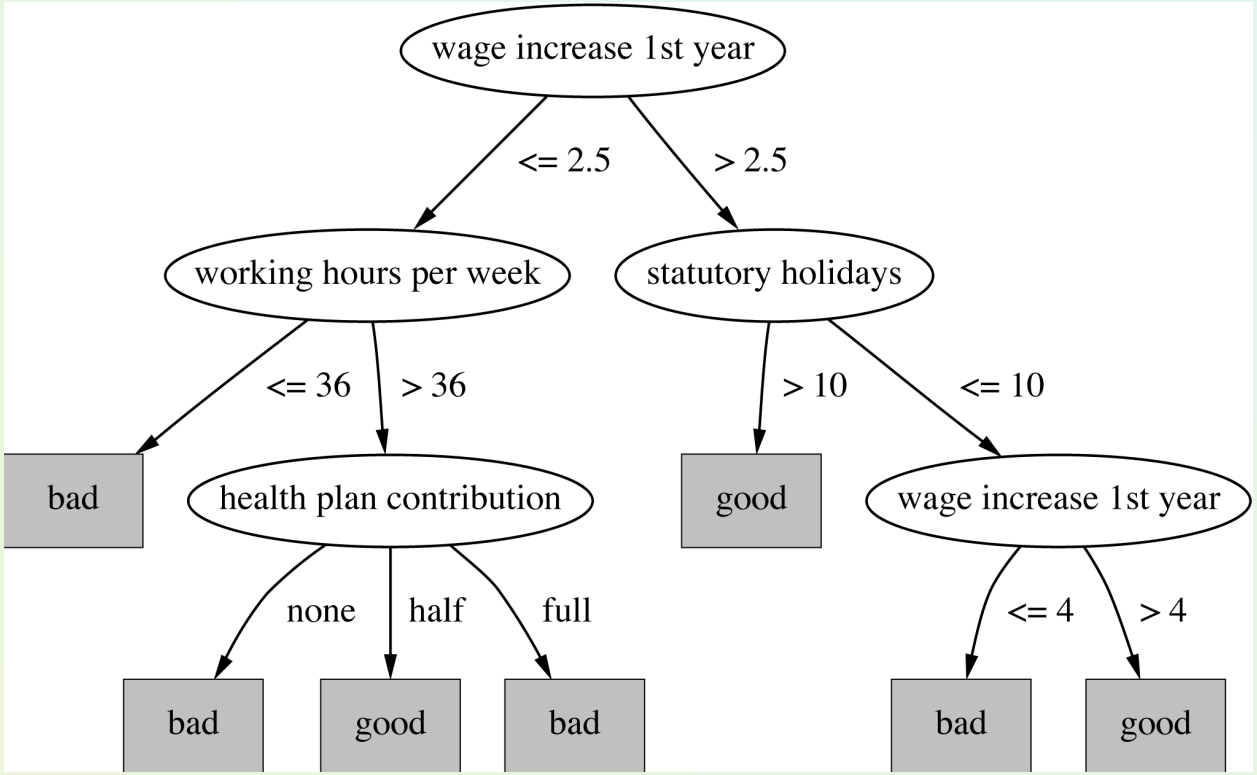
    $\textbf{\textit{Temp}[candidate]} \leftarrow \textbf{\textit{Prune}}\ (T, \textit{candidate})$

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

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

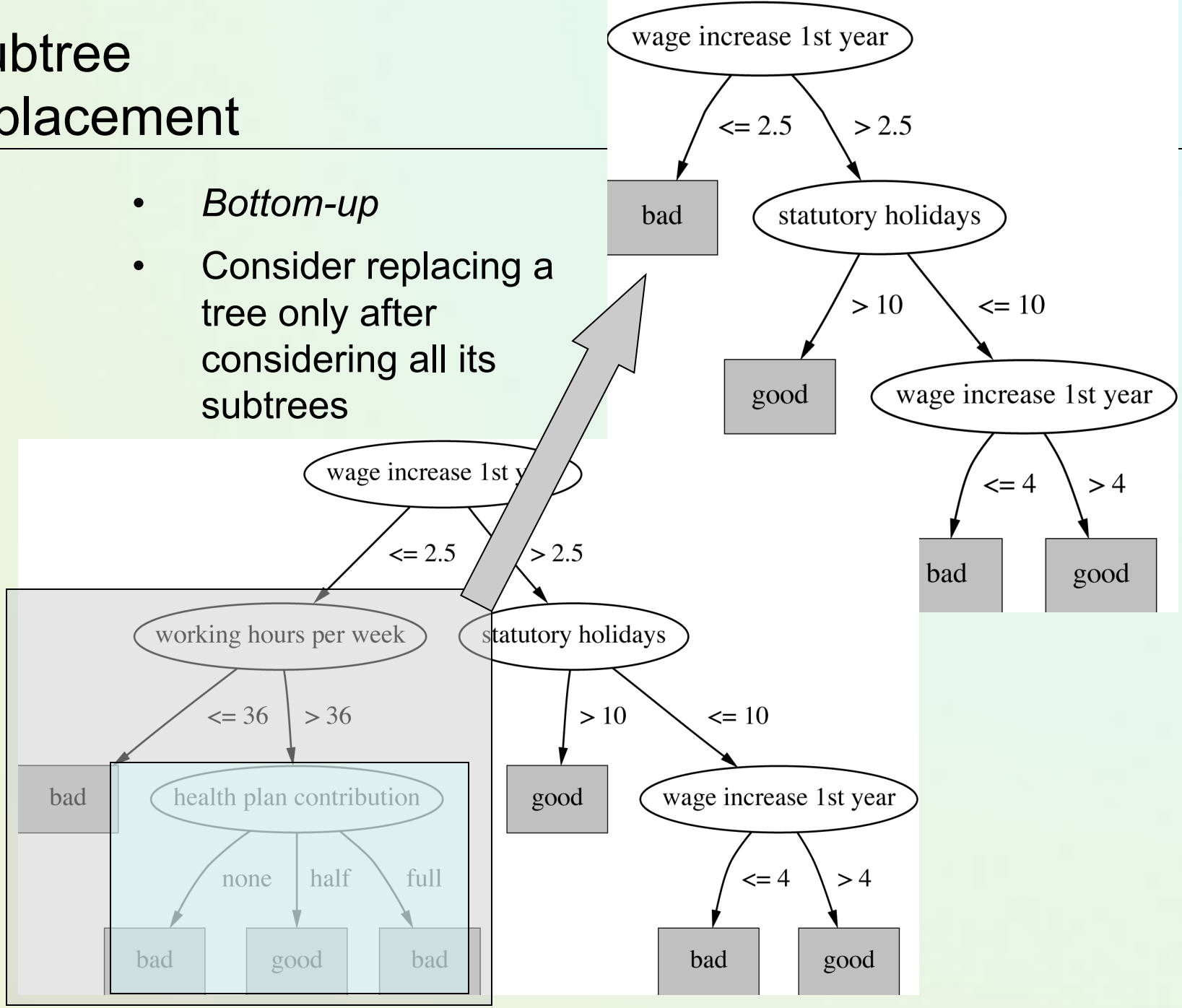
Post-pruning

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



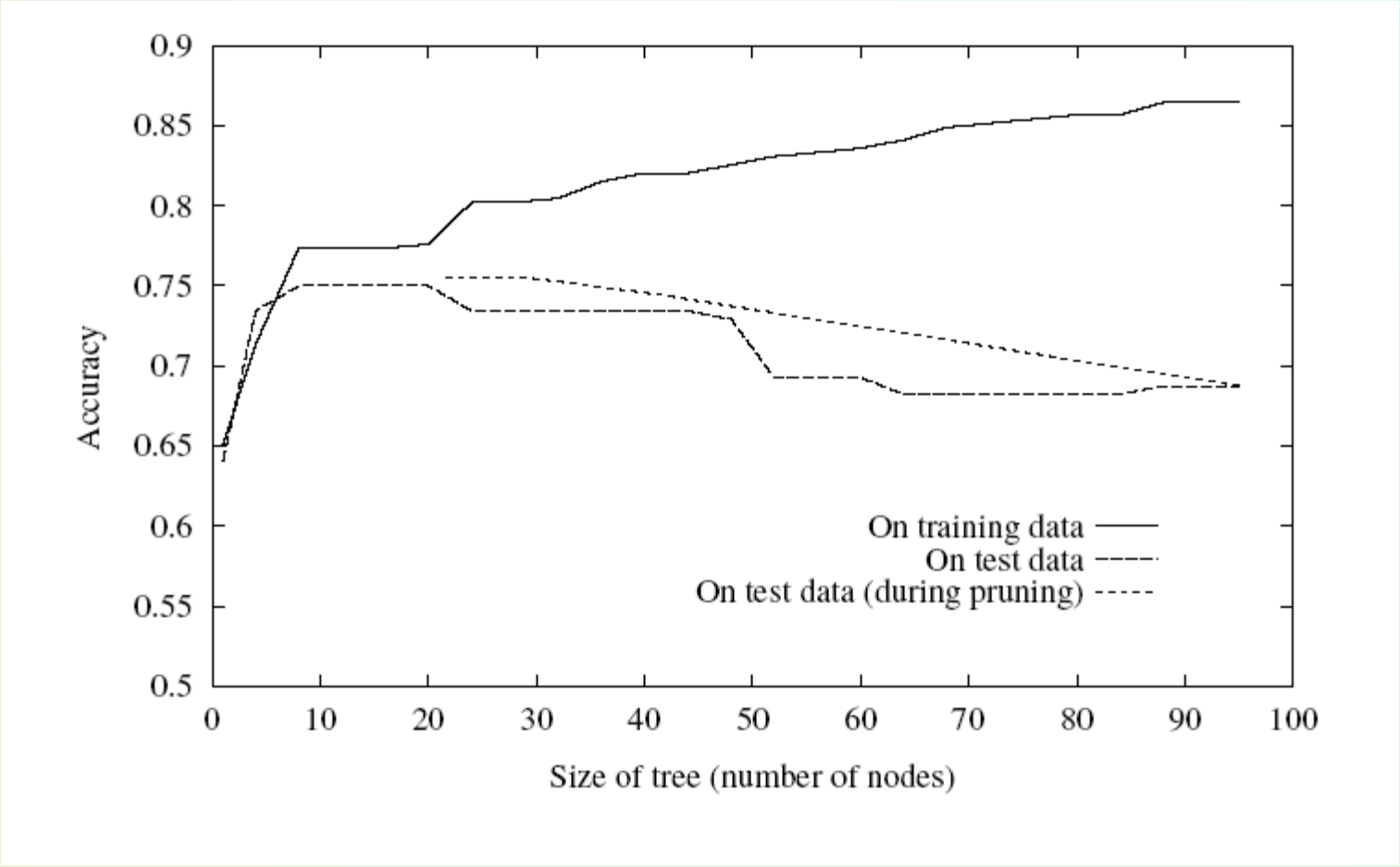
Subtree replacement

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



Reduced post-pruning

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



Pytanie i komentarze?