FP Project

Implementatie van ID3 Algortime

KdG Karel de Grote Hogeschool

Beslissingsbomen

ID3 is een algoritme om beslissingsbomen te creeëren op basis van een dataset

Probleemomschrijving: een beslissingsboom leren op basis van een dataset

---

Gegeven: een dataset T die N voorbeelden i bevat $(x_{i,1}, x_{i,2}, \dots, x_{i,D-1}, x_{i,D})$, i=1..N, en elke $x_{ij} \in A_j$ met A_j de verzameling van waarden die x_{ij} kan aannemen

Vind: een beslissingsboom t over de verzameling van (X,Y,T) die overeenkomt met een functie f: X → Y zodat $f(x_{i,1}, x_{i,2}, \dots, x_{i,D-1}) = x_{i,D}$ met $X = A_1 \times A_2 \times \dots \times A_{D-1}, Y = A_D$ en $T = \{\tau_i | 1 \leq i \leq D-1\}$ en $\tau_i(x_{i,1}, x_{i,2}, \dots, x_{i,D-1}) = x_i$

→ Moeilijk

→ Oplossen met Road Map

→ Hier en daar pseudocode gegeven

Road Map

1. Begrijpen van ID3 algortime (*)
2. Haskell datastructuren maken (*)
3. Utility functies (*)
4. Inlezen en parsen van de weather dataset (**)
5. ID3 hulpfuncties schrijven (***)
6. ID3 splitsfuncties (***)
7. ID3 Tree functie (**)
8. Main functie

Opmerking : een '' geeft de moeilijkheidsgraad*

ID3 Algoritme (*)

‘Weather dataset’

D = 5

$(x_{i,1}, x_{i,2}, \ldots, x_{i,D-1})$

$x_{i,D}$

N = 14

Outlook	Temperature	Humidity	Windy	Play
overcast	hot	high	false	yes
overcast	cool	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
rainy	mild	normal	false	yes
rainy	mild	high	true	no
sunny	hot	high	false	no
sunny	hot	high	true	no
sunny	mild	high	false	no
sunny	cool	normal	false	yes
sunny	mild	normal	true	yes

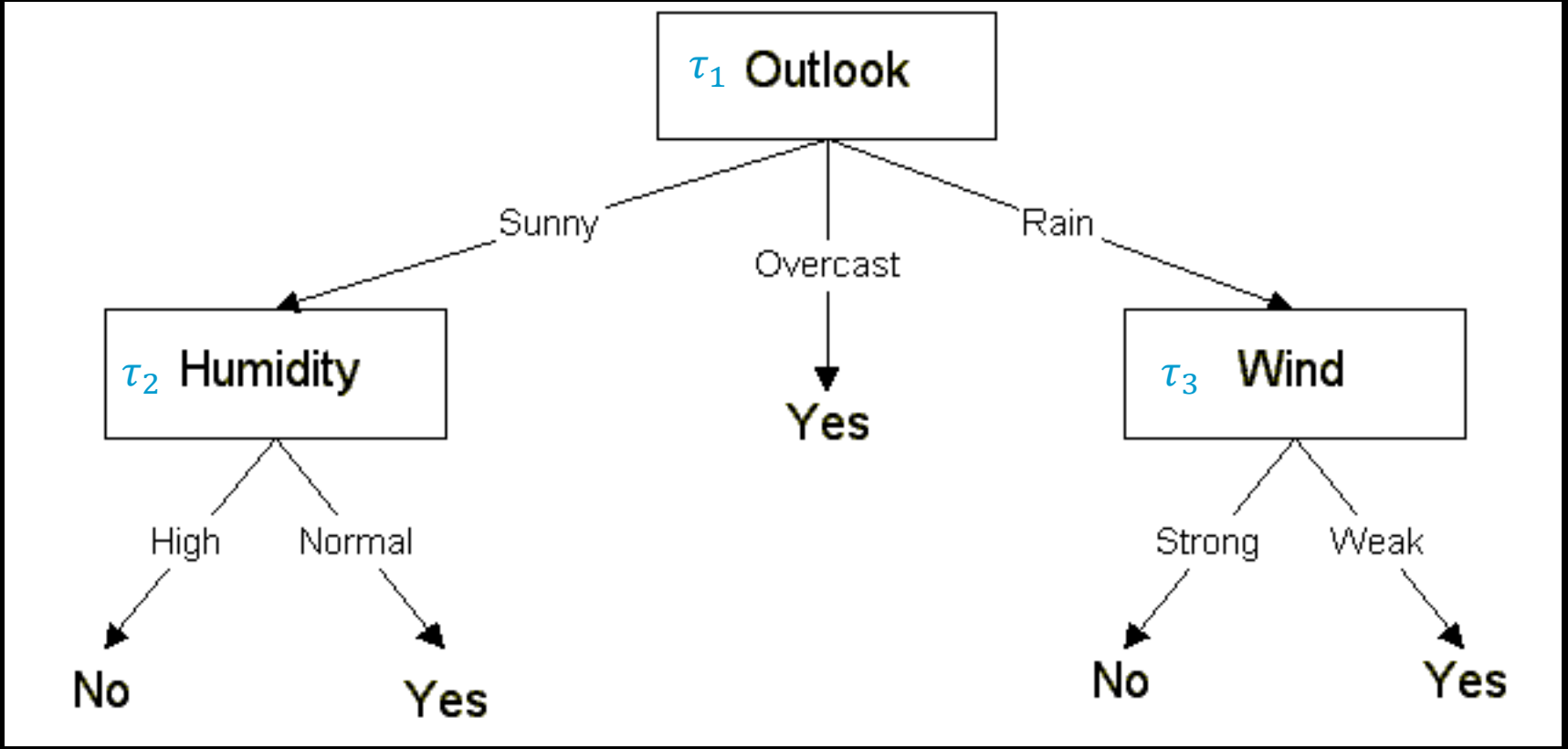
X

Y

Zie BB: weather.csv

ID3 Tree

Zorg dat je het ID3 algoritme eerst volledig begrijpt. Zie volgende slides.



Algoritme

Dataset met voorbeelden

function ID3(S : set of examples) returns decision tree

if $E(S) = 0$ *then return Leaf*

Beslissingsboom heeft Leafs

else

E = entropy van verzameling

for each attribute A_i*:*

for each value v_j *of* A_i*:*

IG = Information Gain

$S_{ij} = \{\boldsymbol{x} \in S \ : \ A_i(\boldsymbol{x}) = v_j\}$

E = entropy van verzameling

Zie BB: weather.*xlsx* voor interactive Excel template.
Zeer handig om algoritme te begrijpen.

let m be such that $IG_m \geq IG_i$ *for all* $i \neq m$

Maximale IGm

for all S_{mj}:

if $S_{mj} = \emptyset$ *then* $t_j = Leaf$

else $t_j = ID3(S_{mj})$

Recursie

return $Node(\tau, \{j, t_j\})$

Beslissingsboom heeft Nodes
met test 'τ' en verzameling subtrees

Entropy berekening

$$E(S) = -\sum_j \frac{|S_{ij}|}{|S|} \cdot \log_2\left(\frac{|S_{ij}|}{|S|}\right)$$

$$S_{i1} = \{1,2,3,4,5,6,7,8,13,14\}, |S_{i1}| = 9$$

$$S_{i2} = \{7,9,10,11,12\}, |S_{i2}| = 5$$

$$E(S) = -\frac{9}{14} \cdot \log_2\left(\frac{9}{14}\right) - \frac{5}{14} \cdot \log_2\left(\frac{5}{14}\right)$$

$$E(S) = 0{,}9403 \neq 0 \rightarrow niet\ zuiver$$

$$v_1 = "yes"$$

$$v_2 = "no"$$

Reken zelf na

Doe het nu!

Nu! Zeg ik !

	Outlook	Temperature	Humidity	Windy	Play
1	overcast	hot	high	false	yes
2	overcast	cool	normal	true	yes
3	Overcast	mild	high	true	yes
4	overcast	hot	normal	false	yes
5	rainy	mild	high	false	yes
6	rainy	cool	normal	false	yes
7	rainy	cool	normal	true	no
8	rainy	mild	normal	false	yes
9	rainy	mild	high	true	no
10	sunny	hot	high	false	no
11	sunny	hot	high	true	no
12	sunny	mild	high	false	no
13	sunny	cool	normal	false	yes
14	sunny	mild	normal	true	yes

Information Gain berekening

Attribute	Waarden	"yes"	"no"	$E(S_{ij})$	$IG(S_{ij})$
Outlook	overcast	4	0	$-\frac{4}{4}\cdot\log_2(\frac{4}{4}) - \frac{0}{4}\cdot\log_2(\frac{0}{4}) = 0$	$0.9403 - \frac{4}{14}\cdot 0 - \frac{5}{14}\cdot 0.971 \frac{5}{14}\cdot 0.971$ $= 0{,}2467$
	rainy	3	2	$-\frac{3}{5}\cdot\log_2(\frac{3}{5}) - \frac{2}{5}\cdot\log_2(\frac{2}{5}) = 0{,}971$	
	sunny	2	3	$-\frac{2}{5}\cdot\log_2(\frac{2}{5}) - \frac{3}{5}\cdot\log_2(\frac{3}{5}) = 0{,}971$	

	Outlook	Temperature	Humidity	Windy	Play
1	overcast	hot	high	false	yes
2	overcast	cool	normal	true	yes
3	overcast	mild	high	true	yes
4	overcast	hot	normal	false	yes
5	rainy	mild	high	false	yes
6	rainy	cool	normal	false	yes
7	rainy	cool	normal	true	no
8	rainy	mild	normal	false	yes
9	rainy	mild	high	true	no
10	sunny	hot	high	false	no
11	sunny	hot	high	true	no
12	sunny	mild	high	false	no
13	sunny	cool	normal	false	yes
14	sunny	mild	normal	true	yes

Information Gain berekening

Attribute	Waarden	"yes"	"no"	$E(S_{ij})$	$IG(S_{ij})$
Temperature	Hot	2	2	1	0,0294
	mild	4	2	0,918	
	cool	3	1	0,811	

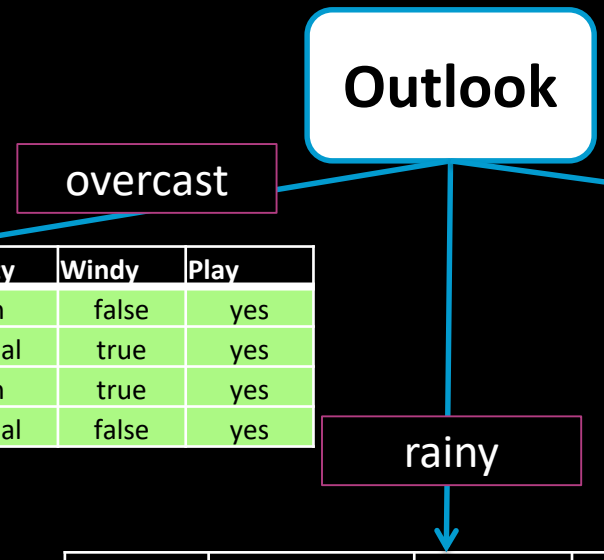
Attribute	Waarden	"yes"	"no"	$E(S_{ij})$	$IG(S_{ij})$
Humidity	high	3	4	0,985	0,152
	normal	6	1	0,592	

Attribute	Waarden	"yes"	"no"	$E(S_{ij})$	$IG(S_{ij})$
Windy	true	3	3	1	0,0483
	false	6	2	0,811	

Information Gain berekening

Attribute Outlook heeft hoogste IG

→ dataset S opsplisten in drie subtabellen volgens waarden van Outlook
→ Zelfde procedure ***herhalen*** op subtabellen

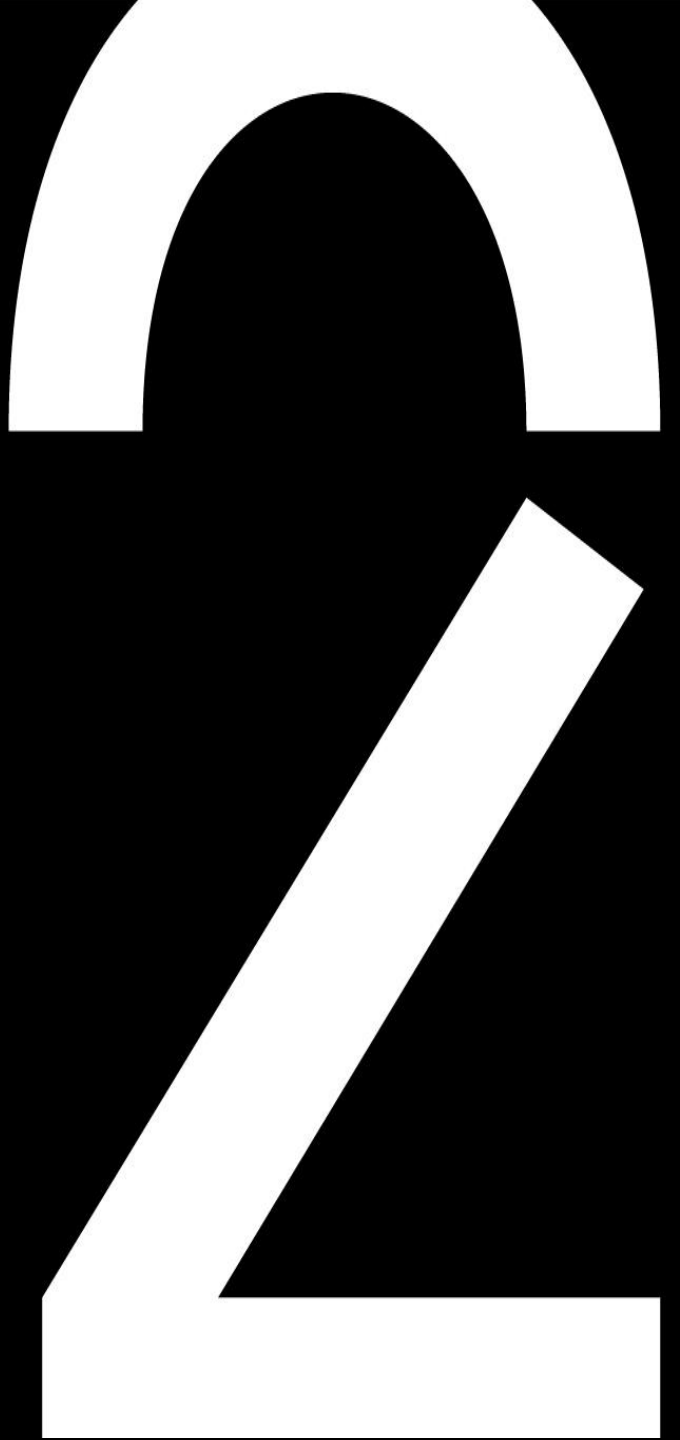


Outlook	Temperature	Humidity	Windy	Play
overcast	hot	high	false	yes
overcast	cool	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	false	no
sunny	hot	high	true	no
sunny	mild	high	false	no
sunny	cool	normal	false	yes
sunny	mild	normal	true	yes

Outlook	Temperature	Humidity	Windy	Play
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
rainy	mild	normal	false	yes
rainy	mild	high	true	no

Haskell Datastructuren (*)



Stap 2: Haskell Datastructuren (*)

Creëer volgende types synomys en datastructuren

1. Type synomys voor String
   - Filename, AttributeName, DomainValue, TargetValue
2. Type synomys voor [String]
   - Instance

3. Datastructuren
   - Gebruik *Record Syntax* waar mogelijk

     Google: record syntax site:http://learnyouahaskell.com/
   - Attribute is een **tuple** van een AttributeName, en een array van DomainValue
   - Set is datastructuur met als value constructor DataSet
   - Tree is recursieve datastructuur met twee value constructors:
     - Leaf
     - Node: een Node heeft een **tuple** (AttributeName,DomainValue) en een array ‘el’ van van Tree elementen

Opm.: vergeet ‘deriving Show’ niet

Utility functies (*)

Stap 3: utility functies (*)

A. `unique :: (Ord a) => [a] -> [a]`

a) Geeft de unieke elementen van een rij terug

b) Gebruik Currying en Partial Application

Google curring site:http://learnyouahaskell.com

Google partial application site:http://learnyouahaskell.com

B. `argmax :: (Ord a) => [a] -> Int`

```
argmax xs = …
```

a) Geeft de index terug van het grootste element in de rij

b) Tip: transformeer de rij van *elementen* in een rij van *(index, element) – tuples*

→ `[(0,e1),(1,e2), ...]`

c) Handige functies: `zip`, `snd`, `head`, `take`, `reverse`

Stap 3: utility functies (*)

C. `getAttributeNames :: Set -> [AttributeName]`

a) Deze functie extraheert alle attribuutnamen uit de Set structuur

b) Gebruik *Pattern Matching* om de Set te matchen met de Dataset value constructor zodat je aan de attributes en instances kan (http://learnyouahaskell.com/making-our-own-types-and-typeclasses)

c) Handige functies: `map`, `fst`, `init`

D. `getDomainValues :: Set -> AttributeName -> [DomainValue]`

a) Geeft alle domeinwaarde terug die horen bij één specifiek attribuut

b) Gebruik weer *Pattern Matching*

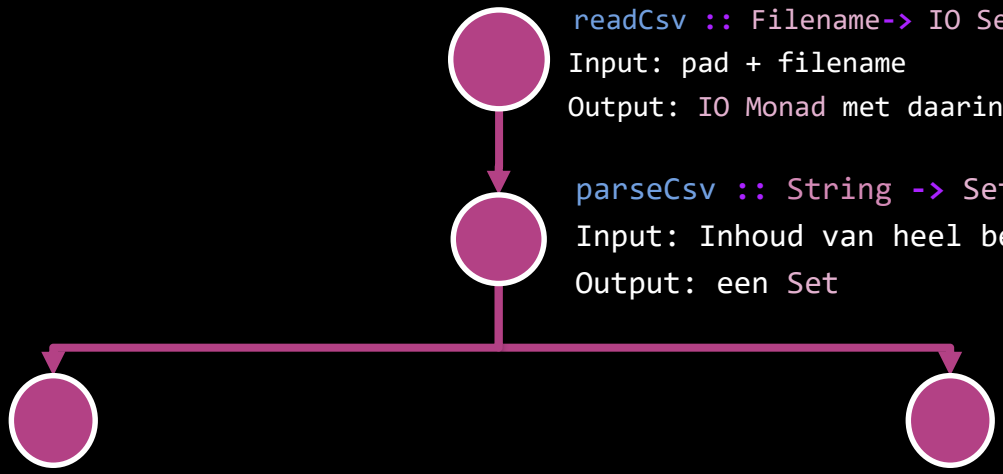
c) Gebruik een list comprehension

Inlezen en parsen van de data (**)

4

Stap 4: inlezen en parsen (**)

Gebeurt met verschillende functies en verscheidene hulpfuncties



`readCsv :: Filename-> IO Set`
Input: pad + filename
Output: IO Monad met daarin Set

`parseCsv :: String -> Set`
Input: Inhoud van heel bestand
Output: een Set

`createInstance :: String -> Instance`
Input: een lijn data uit bestand
Output: een Instance = [String]

`createAttributes :: [String] -> [Instance] -> [Attribute]`
Input 1: array van header waarden
Input 2: array van Instance = [Instance] = [[String]]
Output: array van Attribute = [(AttributeName, [DomainValue])]

Stap 4: inlezen en parsen (**)

A. `readCsv :: Filename -> IO Set`

`readCsv fileName = …`

a) Opent en leest het bestand in als een String
b) Geeft inhoud van bestand door aan `parseCsv`
c) Gebruik `do`
d) *Opmerking: Filename is een Type Synonym voor een String*

B. `parseCsv :: String -> Set`

`parseCsv contents = …`

a) Gebruik `let … in` constructie
b) Handige hulpfuncties: `lines, take, drop, map`
c) Creëert instances en attributes
d) Creëert Dataset m.b.v. instances en attributes en geeft dit terug

Stap 4: inlezen en parsen (**)

C. `createInstance :: String -> Instance`

`createInstance instanceStr = …`

a) Instance is een Type Synonym voor [String]

b) Gebruik *Curring* en *Partial Application*

c) Gebruik hulpfunctie `splitOn`

D. `createAttributes :: [String] -> [Instance] -> [Attribute]`

`createAttributes headerStr instances = …`

a) De header is de eerste lijn van de file, maar is toch een array element

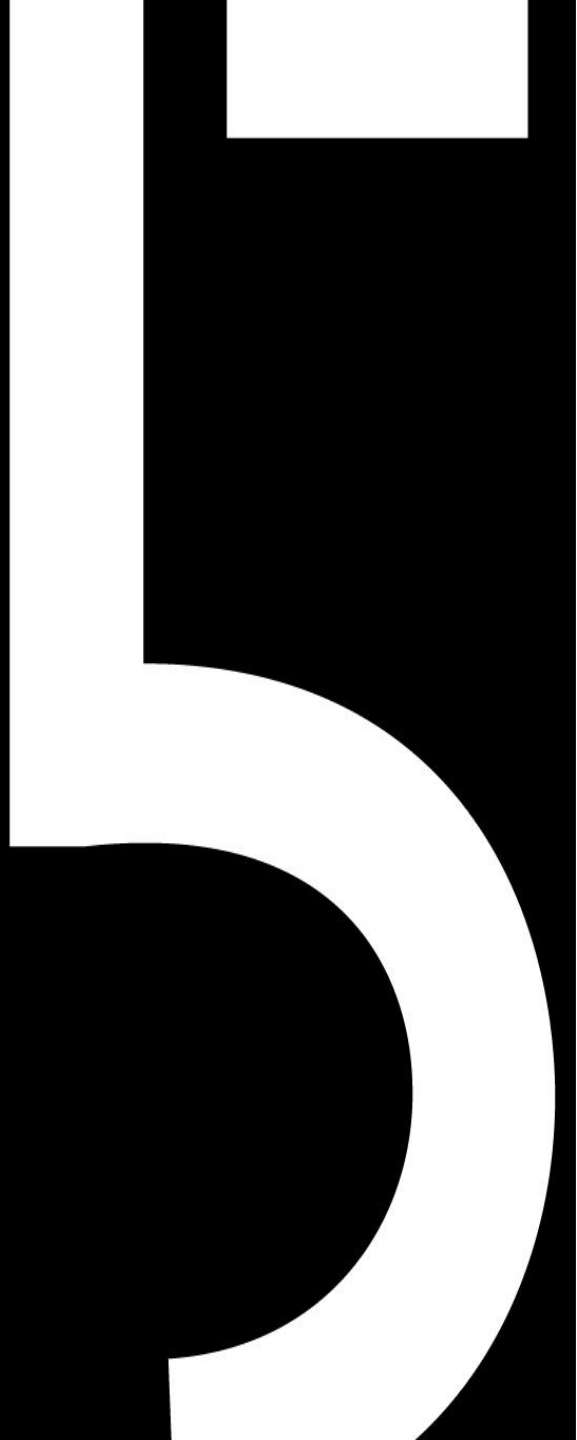
b) Gebruik `let … in`

c) Je moet [Attribute] maken, bekijk de definitie van Attribute nog eens goed

d) Handige functies: `map, transpose, zipWith`

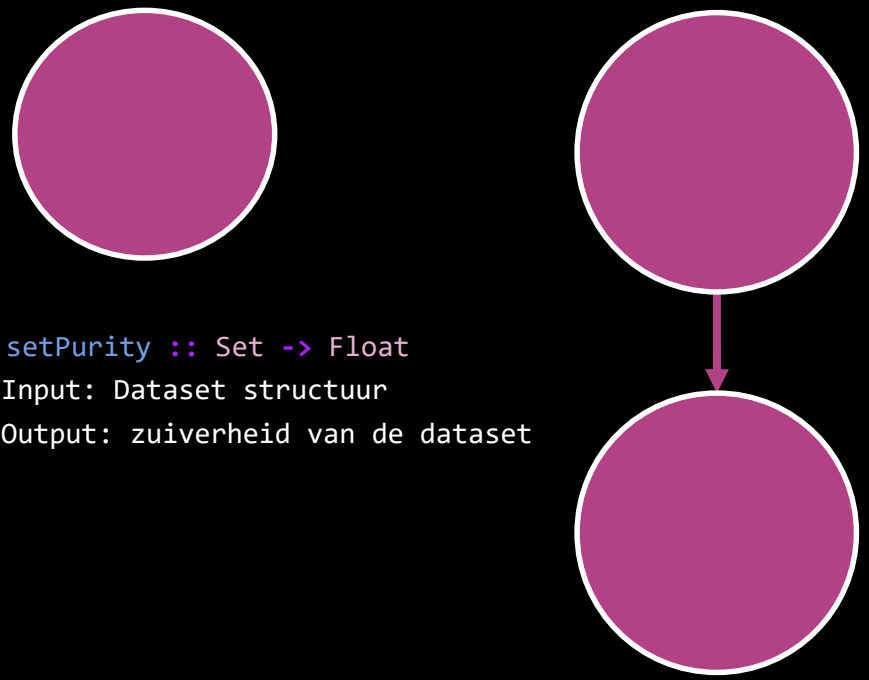
e) Tip: `map` werkt wel rijen maar niet op kolommen.

ID3 hulpfuncties schrijven (***)



Stap 5: ID3 hulpfuncties schrijven (***)

Gebeurt weer met verschillende functies en verscheidene hulpfuncties



```
setPurity :: Set -> Float
Input: Dataset structuur
Output: zuiverheid van de dataset
```

```
purity :: Set -> AttributeName -> Float
Input: Dataset + attribuutnaam
Output: zuiverheid voor dit attribuut
```

```
Entropy :: Float -> Float -> Float
Input: positieve en negatieve aantallen
Output: entropy waarde
```

Stap 5: ID3 hulpfuncties schrijven (***)

A. `entropy :: Float -> Float -> Float`

`entropy a b = …`

a) Berekent de entropy op basis van de aantallen a en b

b) Handige hulpfuncties: `logBase`, `isNaN`

c) *Opmerking: indien de entropy oneindig (Not A Number) is, geef je nul terug.*

B. `purity :: Set -> AttributeName -> Float`

`purity set@…`

a) Berekent de puurheid voor een attribuut binnen een Dataset

b) Gebruik eventueel `let … in …` `where` (where kan gebruikt worden voor definitie hulpfunctie)

c) Pseudocode

Haal de domainwaarden op die horen bij dit attribuut

Tel voor elke domeinwaarde het aantal positieve “yes” instanties in de dataset

Tel voor elke domeinwaarde het aantal negatieve “no” instanties in de dataset

Bereken de totale tellingen (=positieve + negatieve) en maak hier een array van

Bereken alle entropie-waarden voor deze tellingen

Bereken tot slot de entropie met bovenstaande gegevens

ID3 splitsfuncties (***)

Stap 6: ID3 splitfuncties (***)

A. `bestSplit :: Set -> AttributeName`

`bestSplit set@ …`

a) Berekent voor alle attribuutnamen de zuiverheid m.b.v. `purity` – functie

b) Berekent voor gehele set de zuiverheid m.b.v. `setPurity` – functie

c) Berekent uitvoorgaande alle Information Gains

d) Selecteert het attribuut dat overeenkomt met de maximale IG m.b.v `argmax`

B. `splitSet :: Set -> [Set]`

`splitSet set@…`

a) Splitst een Set in een verzameling subsets zoals getoond op deze slide

b) Gebruik eventueel `let … in … where` (where kan weer gebruikt worden voor definitie hulpfunctie)

c) Pseudocode

Als de zuiverheid van de set ≤ 0 geef je een Dataset terug die leeg is
Anders: zoek attribuut dat de dataset het beste splits m.b.v. `bestSplit`

Zoek de bijhorende domeinwaarden van dit attribuut m.b.v. `getDomainValues`

Maak voor elke domeinwaarde een nieuwe set. Hiervoor kan je best een hulpfunctie `createSet :: Set -> AttributeName -> DomainValue -> Set` schrijven die je in de where-clause definieert

d) Handige functies: `guards`, `map`, `filter`, `list comprehension`

ID3 Tree functie (**)

Stap 7: ID3 Tree functie (**)

A. `buildTree :: (AttributeName, DomainValue, Set) -> Tree Set`

   a) Klassieke recursieve functie met basisgeval en algemeen geval

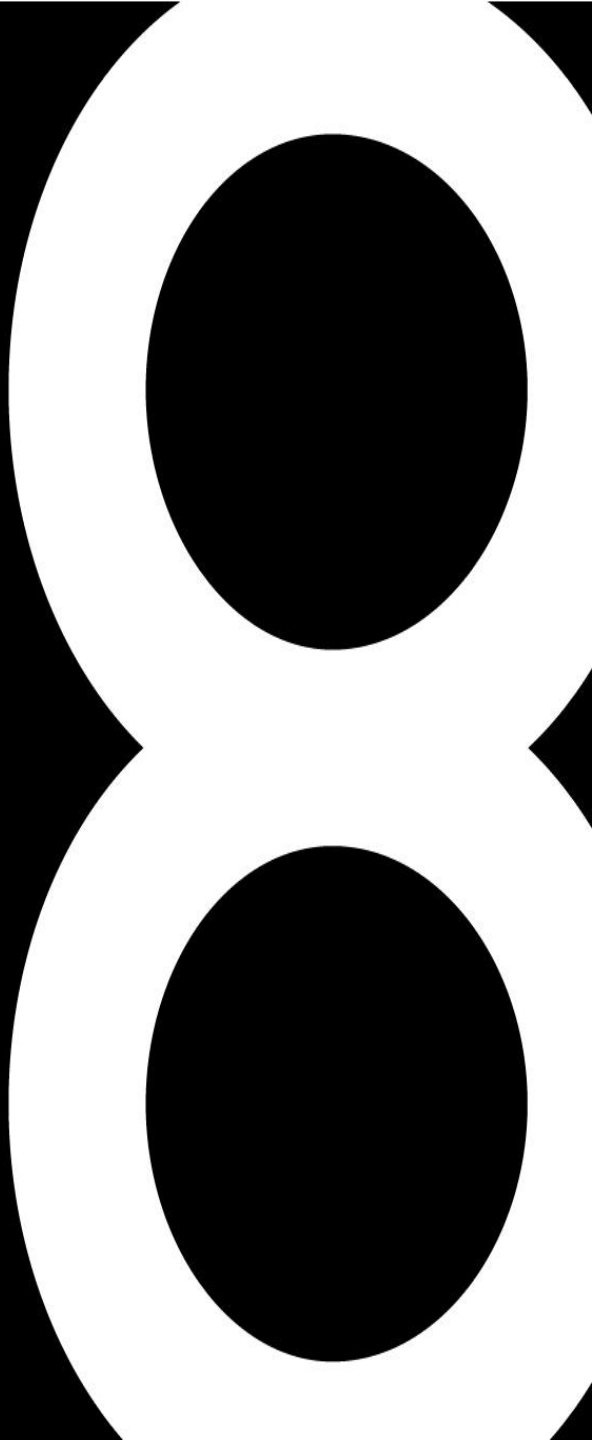
      a) Basisgeval: Leaf toevoegen indien Dataset leeg is

      b) Algemeen geval: dataset splitsen op beste attribuut, domeinwaarden voor beste attribuut ophalen, nieuwe sets maken voor elke domeinwaarde en opnieuw toepassen op subsetten

B. `id3Tree :: Set -> Tree Set`

   a) Gewoon `buildTree` aanroepen met initiële waarden.

Main functie



Stap 8: main functie

```
module Main where
-- imports
-- declaratie type synonyms
-- declaratie datastructuren

main :: IO()
main = do
        -- No buffering on standard output
        hSetBuffering stdout NoBuffering
        -- read and parse data.csv to [Instance]
        dataset <- readCsv "data/weather.csv"
        let first = bestSplit dataset
        print first
        return ()
```