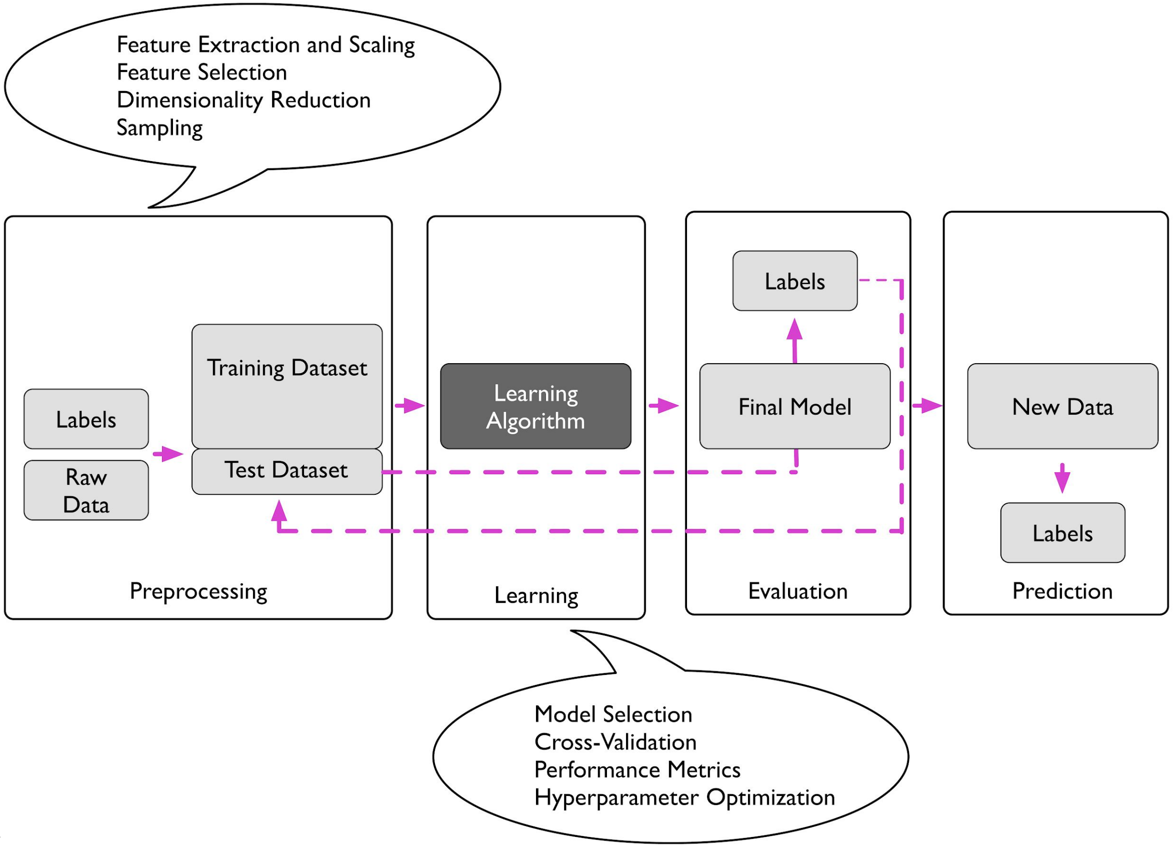
Lecture 12

Decision trees

https://github.com/dalcimar/MA28CP-Intro-to-Machine-Learning

UTFPR - Federal University of Technology - Paraná
https://www.dalcimar.com/

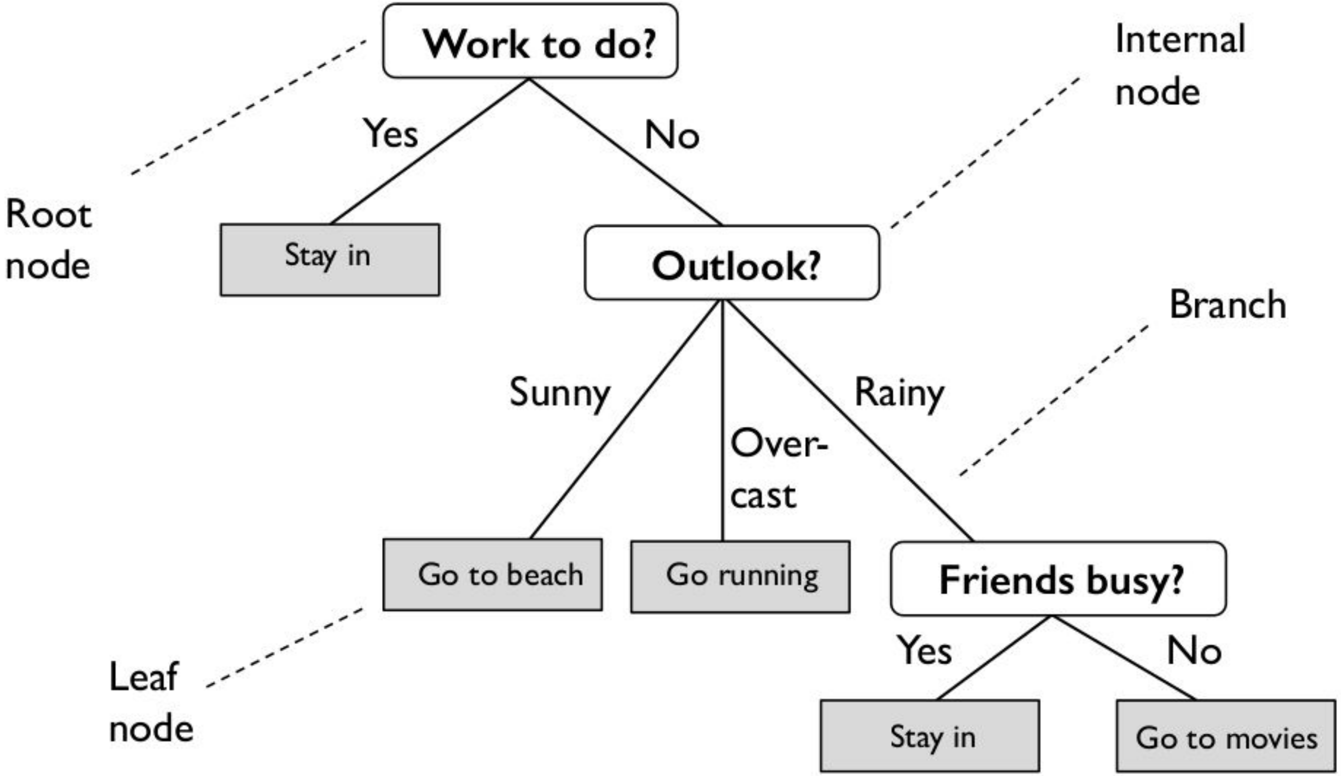
Machine learning pipeline



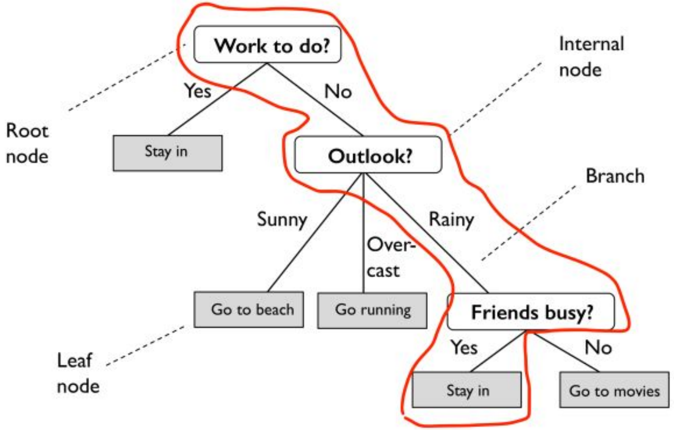
Topics

1. **Intro to decision trees**
2. Recursive and divide & conquer strategy
3. Types of decision trees
4. Splitting criteria
5. Gini & Entropy vs misclassification error
6. Improvements & dealing with overfitting
7. Code example

Decision tree terminology



Decision tree as rulersets



IF ______________________ ___ ______________________

___ ______________________

THEN ________ ___ __________

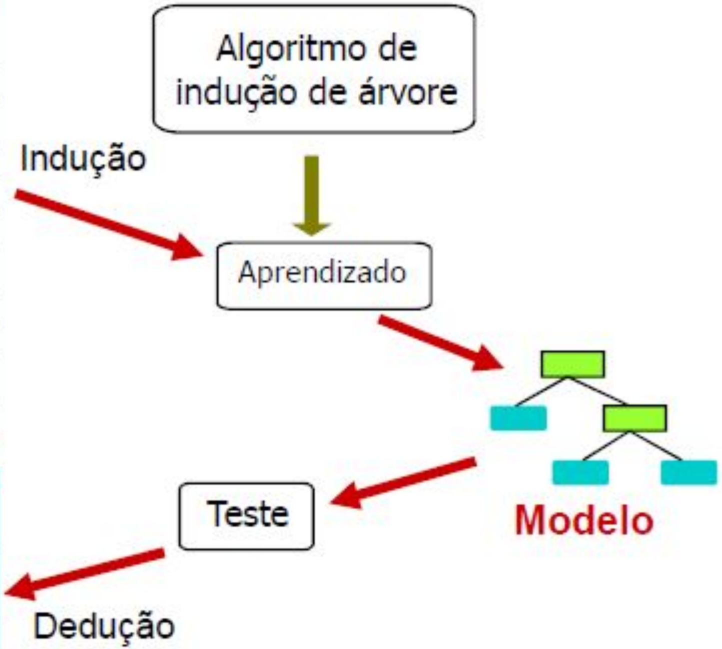
Decision tree pipeline

Conjunto de treinamento

Id	E Credor	Estado Civil	Salário	Calote
1	Sim	Solteiro	125K	Não
2	Não	Casado	100K	Não
3	Não	Solteiro	70K	Não
4	Sim	Casado	120K	Não
5	Não	Divorciado	95K	Sim
6	Não	Casado	60K	Não
7	Sim	Divorciado	220K	Não
8	Não	Solteiro	85K	Sim
9	Não	Casado	75K	Não
10	Não	Solteiro	90K	Sim

Conjunto de teste

Id	E Credor	Estado Civil	Salário	Calote
11	Não	Casado	80K	?
12	Não	Solteiro	100K	?
13	Sim	Solteiro	100K	?
14	Não	Casado	120K	?
15	Sim	Solteiro	80K	?

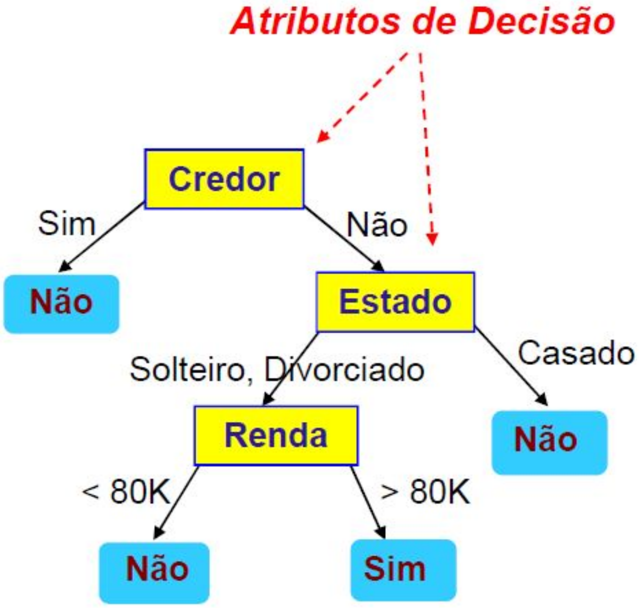


Training

- Works on categorical (binary, nominal or ordinal) and
- Works also with numeric (continuos) attributes

Id	E Credor	Estado Civil	Salário	Calote
1	Sim	Solteiro	125K	Não
2	Não	Casado	100K	Não
3	Não	Solteiro	70K	Não
4	Sim	Casado	120K	Não
5	Não	Divorciado	95K	Sim
6	Não	Casado	60K	Não
7	Sim	Divorciado	220K	Não
8	Não	Solteiro	85K	Sim
9	Não	Casado	75K	Não
10	Não	Solteiro	90K	Sim

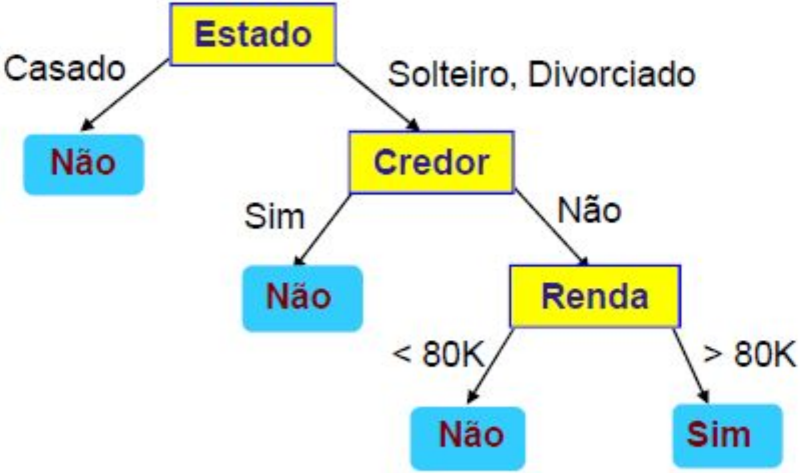
Dados de Treinamento



Modelo: Árvore de Decisão

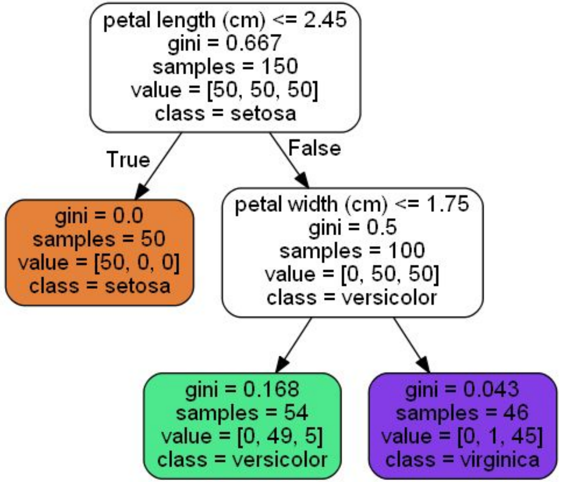
Training

Id	E Credor	Estado Civil	Salário	Calote
1	Sim	Solteiro	125K	Não
2	Não	Casado	100K	Não
3	Não	Solteiro	70K	Não
4	Sim	Casado	120K	Não
5	Não	Divorciado	95K	Sim
6	Não	Casado	60K	Não
7	Sim	Divorciado	220K	Não
8	Não	Solteiro	85K	Sim
9	Não	Casado	75K	Não
10	Não	Solteiro	90K	Sim



Diferentes árvores podem ser ajustadas para os mesmos dados !

Decision tree on Iris dataset (2 features)



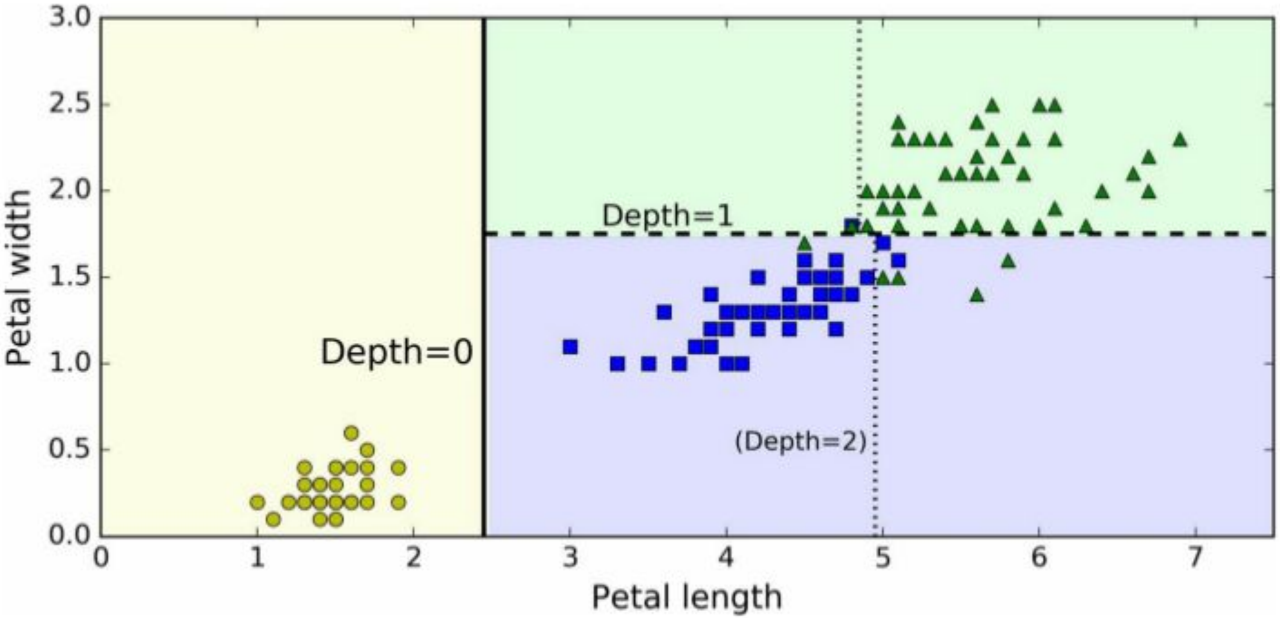
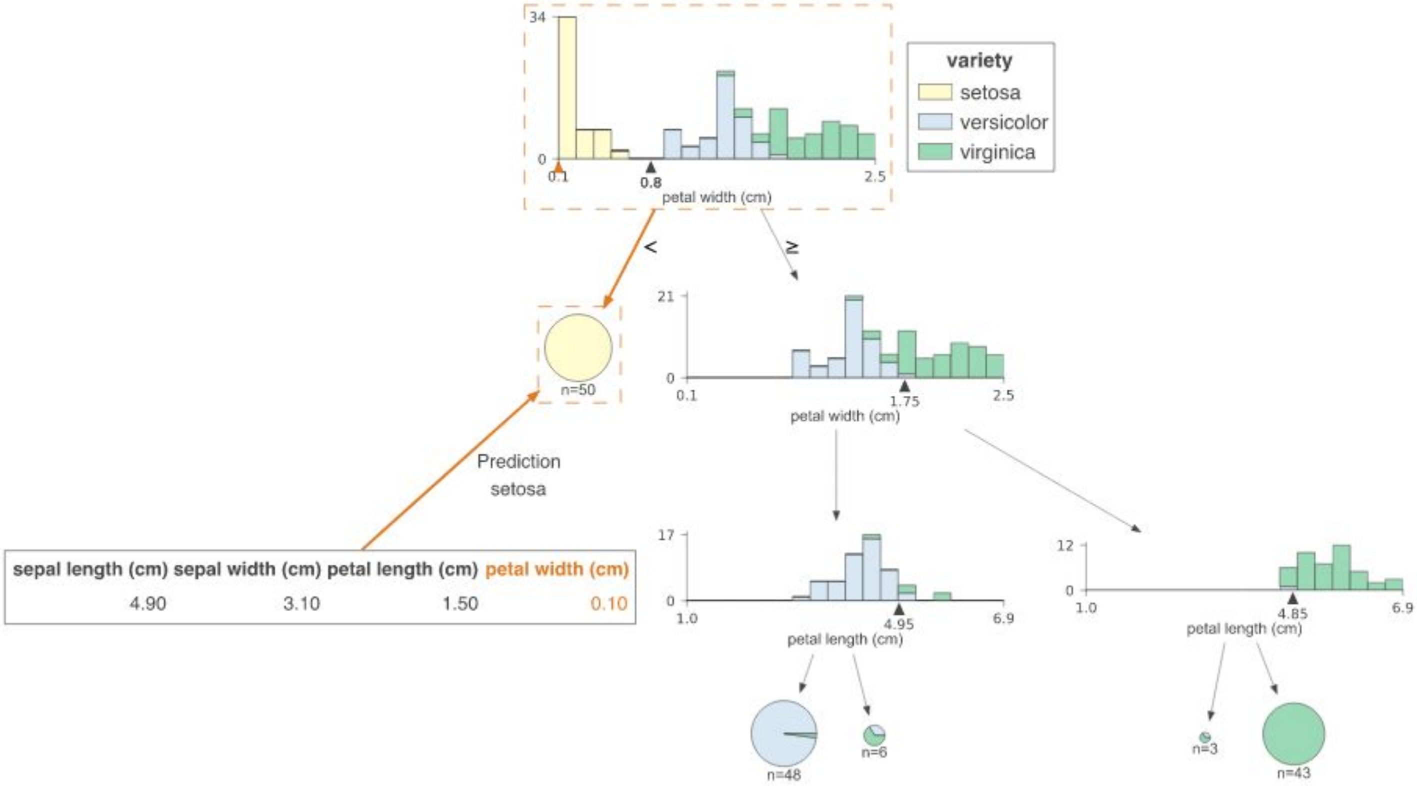


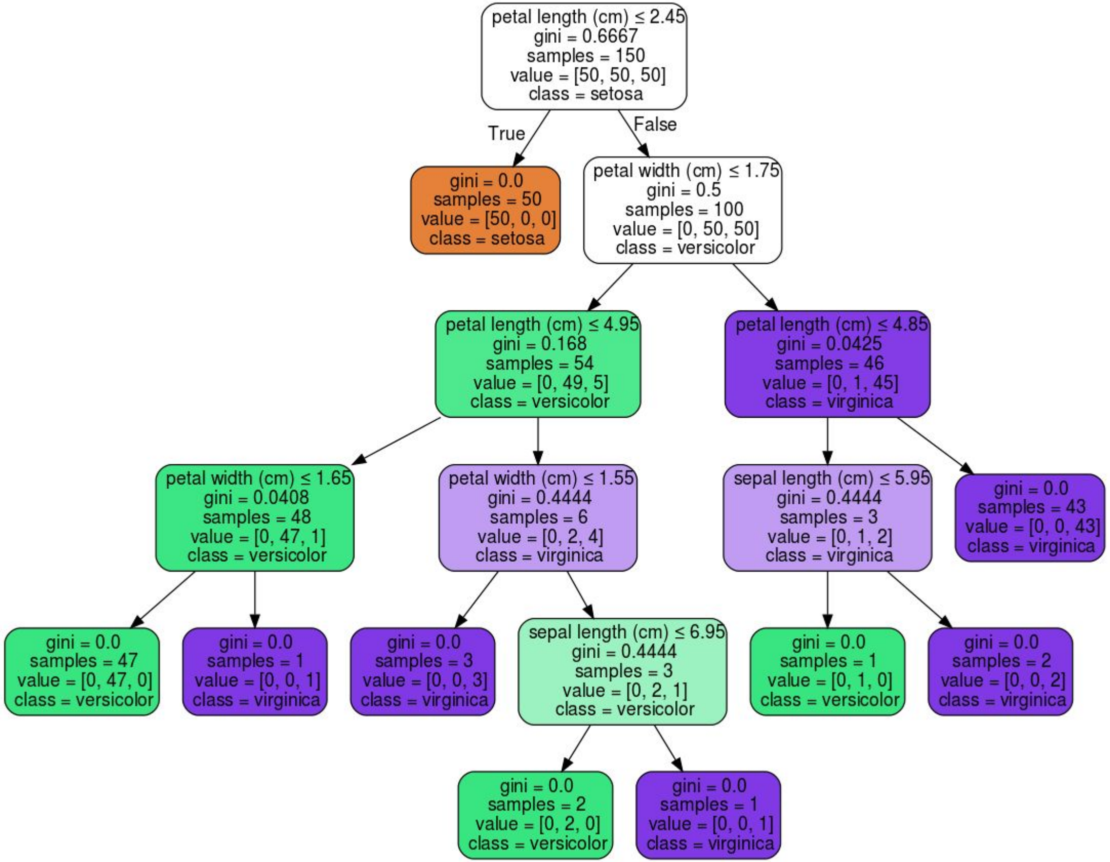
Figure 6-2. Decision Tree decision boundaries

Decision tree on Iris dataset (2 features)

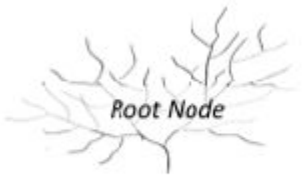


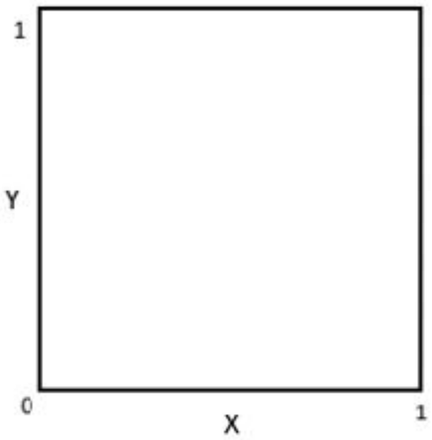
https://explained.ai/decision-tree-viz/images/samples/iris-TD-3-X.svg

Decision tree on Iris dataset (4 features)



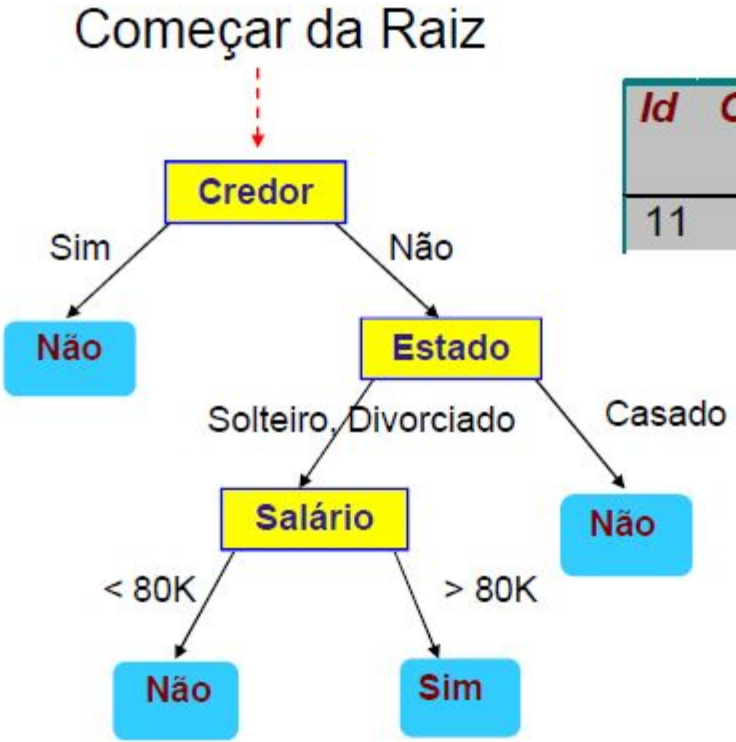
Growing depth process





For more tutorials: annalyzin.wordpress.com

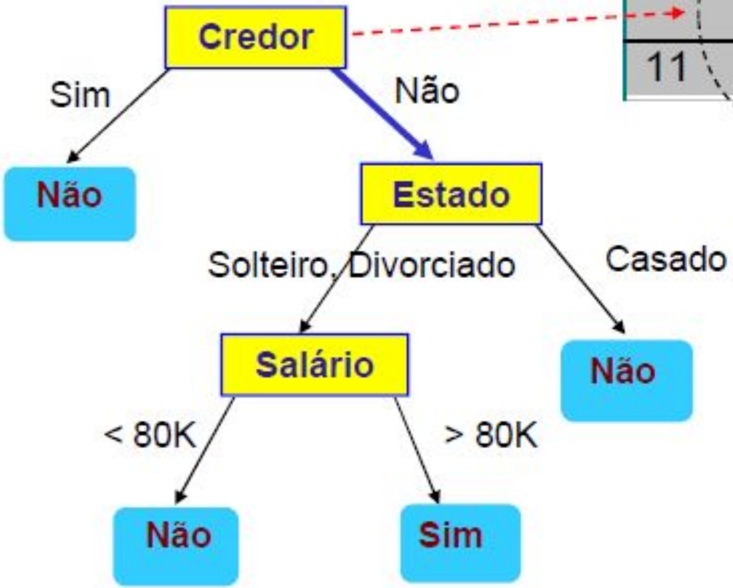
Inference



Dados de Teste

Id	*Credor*	*Estado Civil*	*Salário*	*Calote*
11	Não	Casado	80K	???

Inference



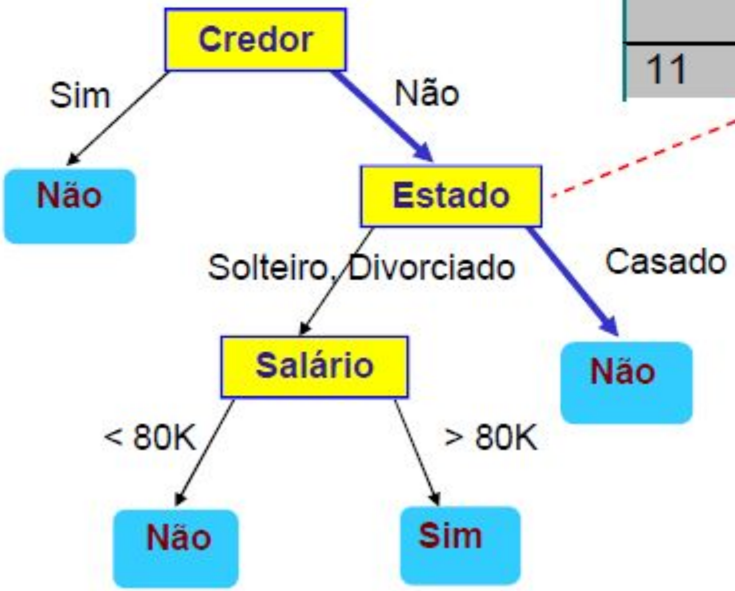
Dados de Teste

Id	Credor	Estado Civil	Salário	Calote
11	Não	Casado	80K	???

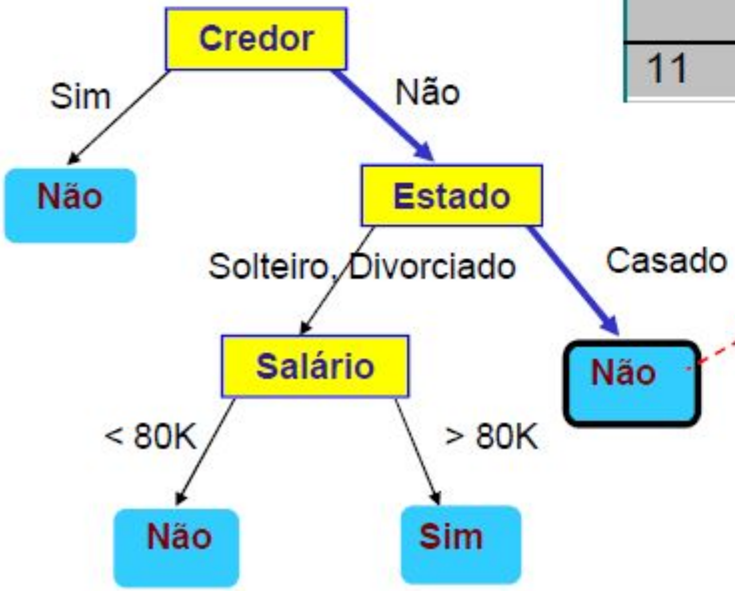
Inference

Dados de Teste

Id	*Credor*	*Estado Civil*	*Salário*	*Calote*
11	Não	Casado	80K	???



Inference



Dados de Teste

Id	*Credor*	*Estado Civil*	*Salário*	*Calote*
11	Não	Casado	80K	???

Atribui classe **Não**

Tree depth overfitting

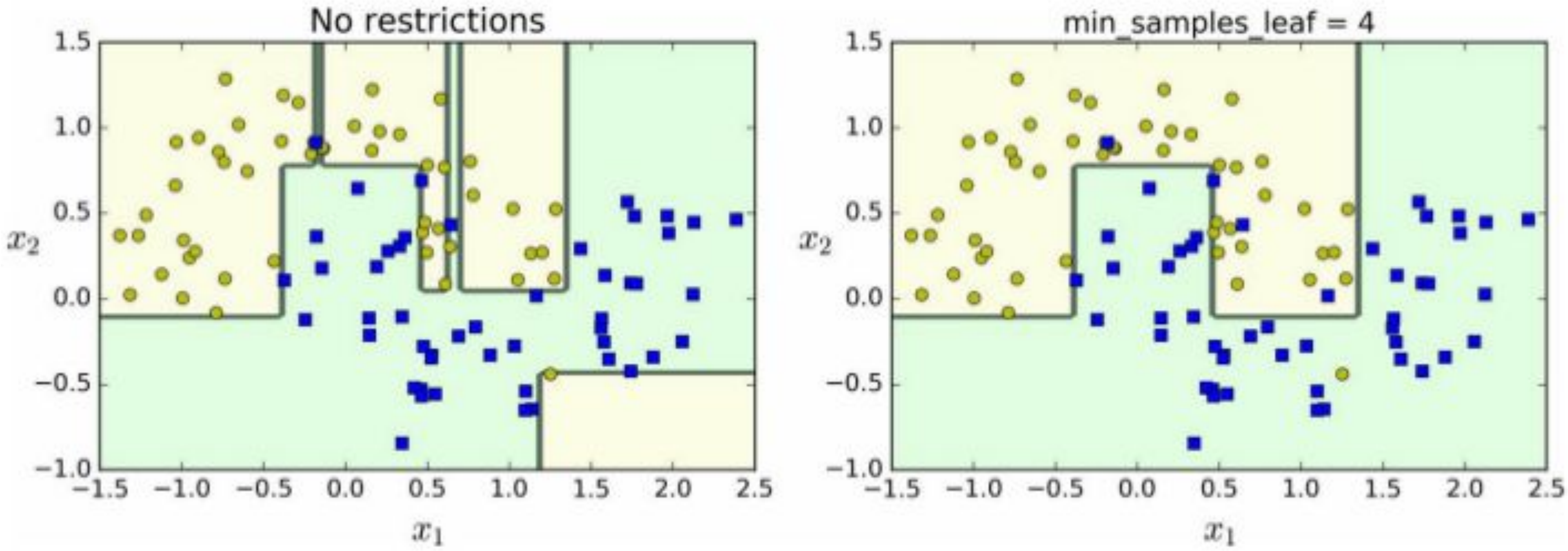


Figure 6-3. Regularization using min_samples_leaf

Animations

Amazing animation of decision tree

http://www.r2d3.us/visual-intro-to-machine-learning-part-1/

Topics

1. Intro to decision trees
2. **Recursive and divide & conquer strategy**
3. Types of decision trees
4. Splitting criteria
5. Gini & Entropy vs misclassification error
6. Improvements & dealing with overfitting
7. Code example

Recursion / Recursive Algorithms

What does this function do?

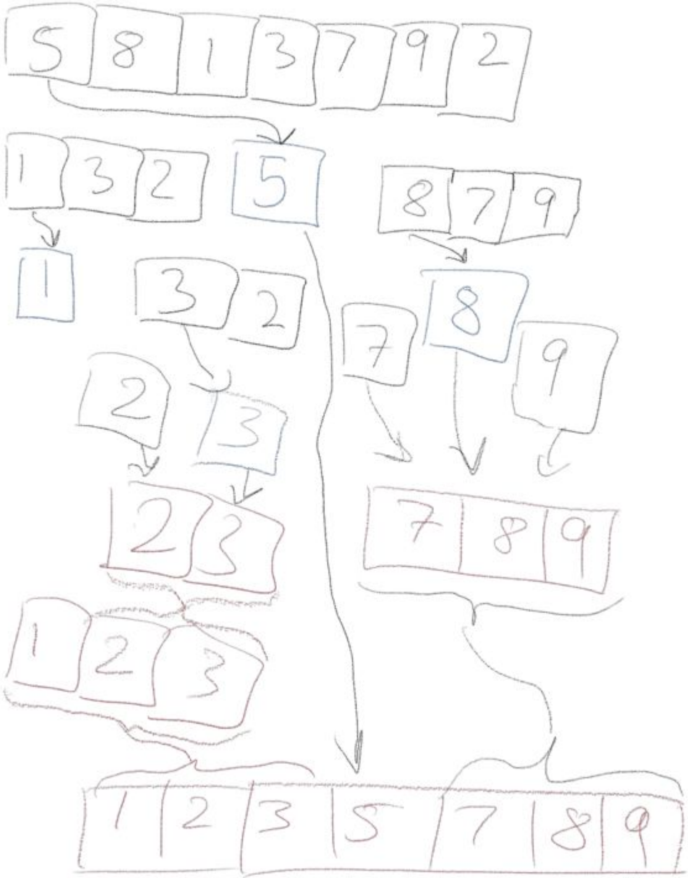
```
def some_fun(x):
    if x == []:
        return 0
    else:
        return 1 + some_fun(x[1:])
```

Divide & Conquer Algorithms: Quicksort

```
def quicksort(array):
    if len(array) < 2:
        return array
    else:
        pivot = array[0]
        smaller, bigger = [], []
        for ele in array[1:]:
            if ele <= pivot:
                smaller.append(ele)
            else:
                bigger.append(ele)
        return quicksort(smaller) + [pivot] + quicksort(bigger)
```

Divide & Conquer Algorithms: Quicksort

```
def quicksort(array):
    if len(array) < 2:
        return array
    else:
        pivot = array[0]
        smaller, bigger = [], []
        for ele in array[1:]:
            if ele <= pivot:
                smaller.append(ele)
            else:
                bigger.append(ele)
        return quicksort(smaller) + [pivot] + quicksort(bigger)
```



Time complexity of quicksort

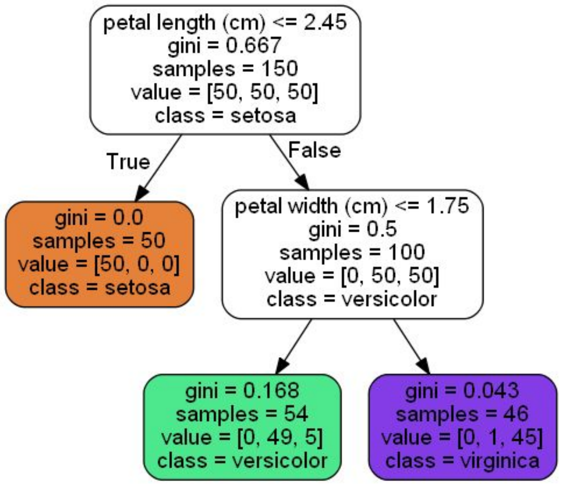
O(n log n)

```
def quicksort(array):
    if len(array) < 2:
        return array
    else:
        pivot = array[0]
        smaller, bigger = [], []
        for ele in array[1:]:
            if ele <= pivot:
                smaller.append(ele)
            else:
                bigger.append(ele)
        return quicksort(smaller) + [pivot] + quicksort(bigger)
```

Time and space complexity of sorting algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n^2)$	$O(\log(n))$
Mergesort	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(n)$
Timsort	$\Omega(n)$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(n)$
Heapsort	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(1)$
Bubble Sort	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$
Insertion Sort	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$
Selection Sort	$\Omega(n^2)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$
Tree Sort	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n^2)$	$O(n)$
Shell Sort	$\Omega(n \log(n))$	$\Theta(n(\log(n))^2)$	$O(n(\log(n))^2)$	$O(1)$
Bucket Sort	$\Omega(n+k)$	$\Theta(n+k)$	$O(n^2)$	$O(n)$
Radix Sort	$\Omega(nk)$	$\Theta(nk)$	$O(nk)$	$O(n+k)$
Counting Sort	$\Omega(n+k)$	$\Theta(n+k)$	$O(n+k)$	$O(k)$
Cubesort	$\Omega(n)$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(n)$

Decision Tree in Pseudocode



GenerateTree($\mathcal{D}$):

- if $y = 1 \ \forall \ \langle \mathbf{x}, \mathbf{y} \rangle \in \mathcal{D}$ or $y = 0 \ \forall \ \langle \mathbf{x}, y \rangle \in \mathcal{D}$:
  - return Tree
- else:
  - Pick best feature x_j:
    - $\mathcal{D}_0$ at Child$_0$: $x_j = 0 \ \forall \ \langle \mathbf{x}, y \rangle \in \mathcal{D}$
    - $\mathcal{D}_1$ at Child$_1$: $x_j = 1 \ \forall \ \langle \mathbf{x}, y \rangle \in \mathcal{D}$

  return Node(x_j, GenerateTree($\mathcal{D}_0$), GenerateTree($\mathcal{D}_1$))

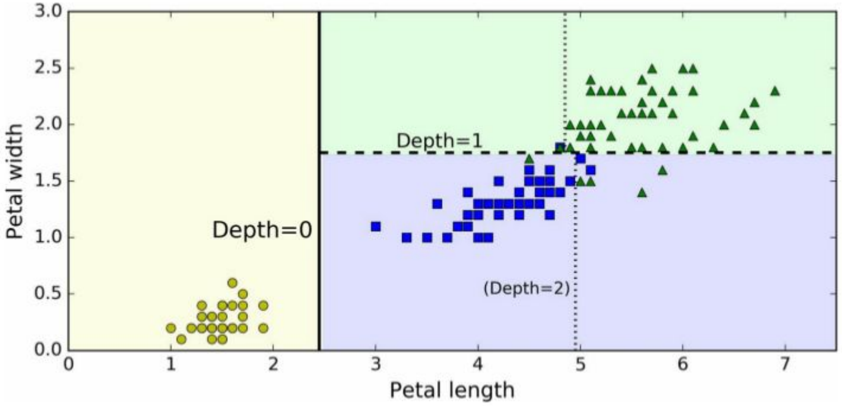


Figure 6-2. Decision Tree decision boundaries

Time complexity decision tree

Growing the tree (**fit()**): $O(m*n^2 \log n)$

- Assuming we have continuous features and perform binary splits, the runtime of the decision tree construction is
- Sorting the values of continuous features helps with determining a decision threshold. If we have n examples, the sorting has time complexity $O(n \log n)$
- If we have to compare sort m features, this becomes $O(m*n \log n)$
- Sorting step up to n/2 times $O(m*n^2 \log n)$

Querying the tree (**predict()**): $O(\log n)$

- depth of $\log_2 n$

Fayyad, Usama Mohammad. "On the induction of decision trees for multiple concept learning." (1992).

Topics

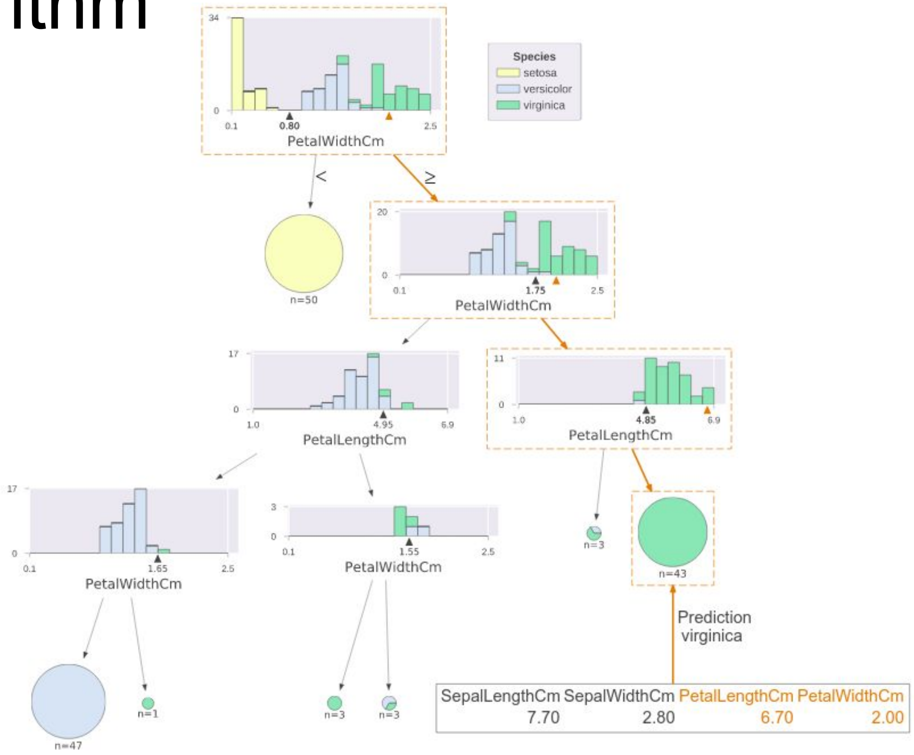
1. Intro to decision trees
2. Recursive and divide & conquer strategy
3. **Types of decision trees**
4. Splitting criteria
5. Gini & Entropy vs misclassification error
6. Improvements & dealing with overfitting
7. Code example

Generic Tree Growing Algorithm

GenerateTree($\mathcal{D}$):

- if $y = 1 \; \forall \; \langle \mathbf{x}, \mathbf{y} \rangle \in \mathcal{D}$ or $y = 0 \; \forall \; \langle \mathbf{x}, y \rangle \in \mathcal{D}$:
    - return Tree
- else:
    - Pick best feature x_j:
        - $\mathcal{D}_0$ at Child$_0$: $x_j = 0 \; \forall \; \langle \mathbf{x}, y \rangle \in \mathcal{D}$
        - $\mathcal{D}_1$ at Child$_1$: $x_j = 1 \; \forall \; \langle \mathbf{x}, y \rangle \in \mathcal{D}$

    return Node(x_j, GenerateTree($\mathcal{D}_0$), GenerateTree($\mathcal{D}_1$))



Generic Tree Growing Algorithm

1. Pick the feature that, when parent node is split, results in the largest information gain
2. Stop if child nodes are pure or information gain <= 0
3. Go back to step 1 for each of the two child nodes

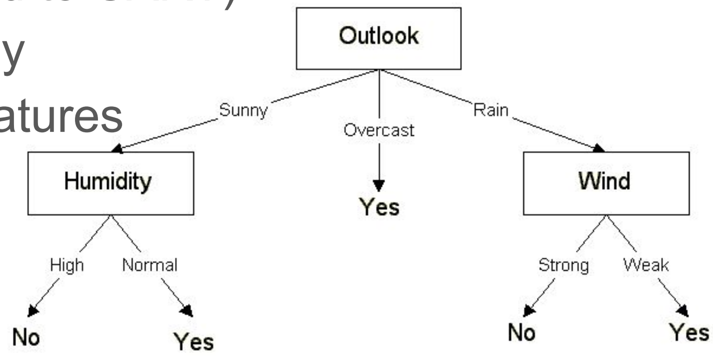
How make predictions of features in dataset not sufficient to make child nodes pure?

Design choices

- What kind of variables
  - Only binary features
  - Only binary or categorical features
  - Numeric features
- How to split
  - what measurement/criterion as measure of goodness
  - binary vs multi-category split
- When to stop
  - if leaf nodes contain only examples of the same class
  - feature values are all the same for all examples
  - statistical significance test

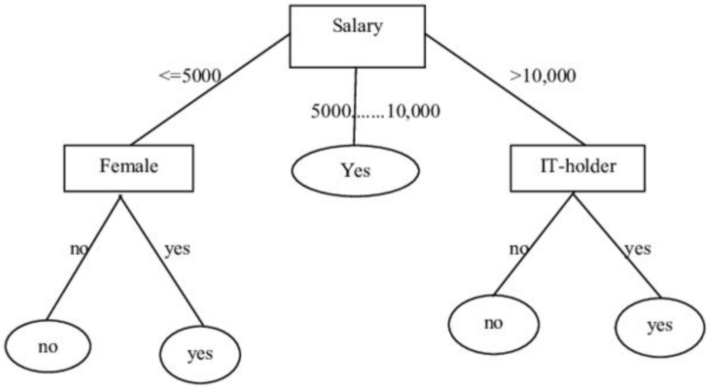
ID3 - Iterative Dichotomizer 3

- Quinlan, J. R. 1986. Induction of Decision Trees. Mach. Learn. 1, 1 (Mar. 1986), 81-106.
- one of the earlier/earliest decision tree algorithms
- **cannot handle numeric features**
- Trees are **grown to their maximum size** and then a **pruning step** is usually applied to improve the ability of the tree to generalise to unseen data
- **multiway tree**, short and wide trees (compared to CART)
- maximizing information gain/minimizing entropy
- discrete features, binary and multi-category features



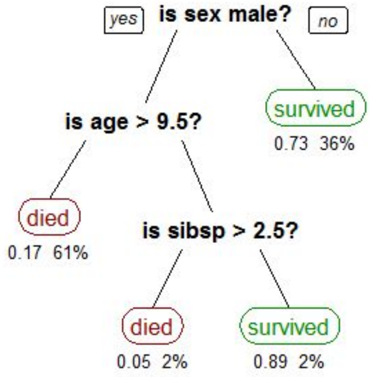
C4.5

- Ross Quinlan 1993, Quinlan, J. R. (1993). C4.5: Programming for machine learning. Morgan Kauffmann, 38, 48.
- **continuous and discrete features**
  - continuous is very expensive, because must consider all possible ranges
- handles missing attributes (ignores them in gain compute)
- **post-pruning** (bottom-up pruning)
- **Gain Ratio**



CART

- Breiman, L. (1984). Classification and regression trees. Belmont, Calif: Wadsworth International Group.
- **continuous and discrete features**
- variance reduction in regression trees
- **strictly binary splits** (taller trees than ID3, C4.5)
    - binary splits can generate better trees than C4.5, but tend to be larger and harder to interpret; k-attributes has a ways to create a binary partitioning
- Gini impurity, twoing criteria in classification trees
- cost complexity pruning



Others

- CHAID (CHi-squared Automatic Interaction Detector); Kass, G. V. (1980). "An exploratory technique for investigating large quantities of categorical data". Applied Statistics. 29 (2): 119–127.
- MARS (Multivariate adaptive regression splines); Friedman, J. H. (1991). "Multivariate Adaptive Regression Splines". The Annals of Statistics. 19: 1
- C5.0 (patented)

Topics

1. Intro to decision trees
2. Recursive and divide & conquer strategy
3. Types of decision trees
4. **Splitting criteria**
5. Gini & Entropy vs misclassification error
6. Improvements & dealing with overfitting
7. Code example

Splitting without criteria

Let D_t be the set of training records that are associated with node t and y = {y_1, y_2 ,...y_c }, where y is the target variable with c number of classes. The following is a recursive definition of **Hunt's algorithm**

Basis of many existing decision tree algorithm including ID3, C4.5 and CART.

Step 1 :

- **If all the records in D_t belong to the same class y_t**, then node t is a leaf node labeled as y_t

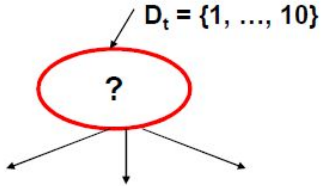
Step 2 :

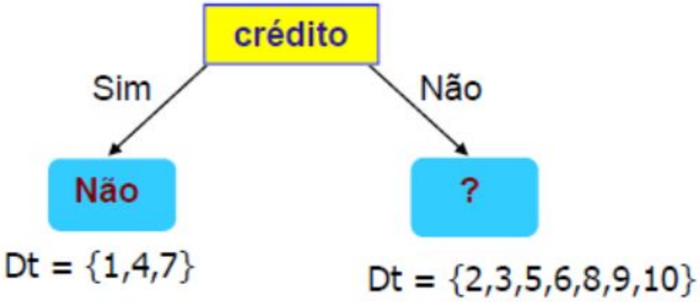
- **If D_t contains records that belong to more than one class**, an attribute test condition is selected to partition the records into smaller subsets. A child node is created for each outcome of the test condition and the records in D_t are distributed to the children based on the outcomes. The algorithm is then recursively applied to each child node.

Splitting without criteria

Hunt's algorithm

Id	Crédito	Estado Civil	Renda	Deve
1	Sim	Solteiro	125K	Não
2	Não	Casado	100K	Não
3	Não	Solteiro	70K	Não
4	Sim	Casado	120K	Não
5	Não	Divorciado	95K	Sim
6	Não	Casado	60K	Não
7	Sim	Divorciado	220K	Não
8	Não	Solteiro	85K	Sim
9	Não	Casado	75K	Não
10	Não	Solteiro	90K	Sim

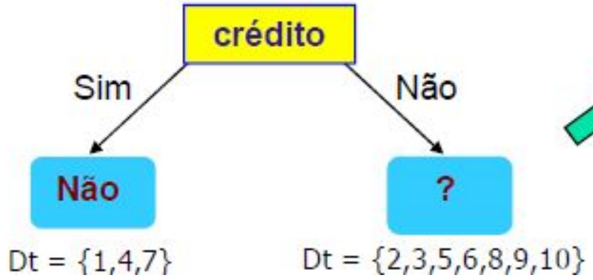


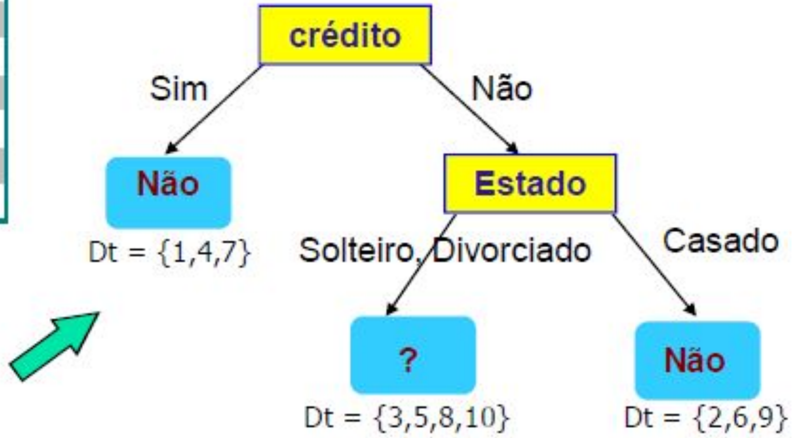


Splitting without criteria

Hunt's algorithm

Id	Crédito	Estado Civil	Renda	Deve
1	Sim	Solteiro	125K	Não
2	Não	Casado	100K	Não
3	Não	Solteiro	70K	Não
4	Sim	Casado	120K	Não
5	Não	Divorciado	95K	Sim
6	Não	Casado	60K	Não
7	Sim	Divorciado	220K	Não
8	Não	Solteiro	85K	Sim
9	Não	Casado	75K	Não
10	Não	Solteiro	90K	Sim

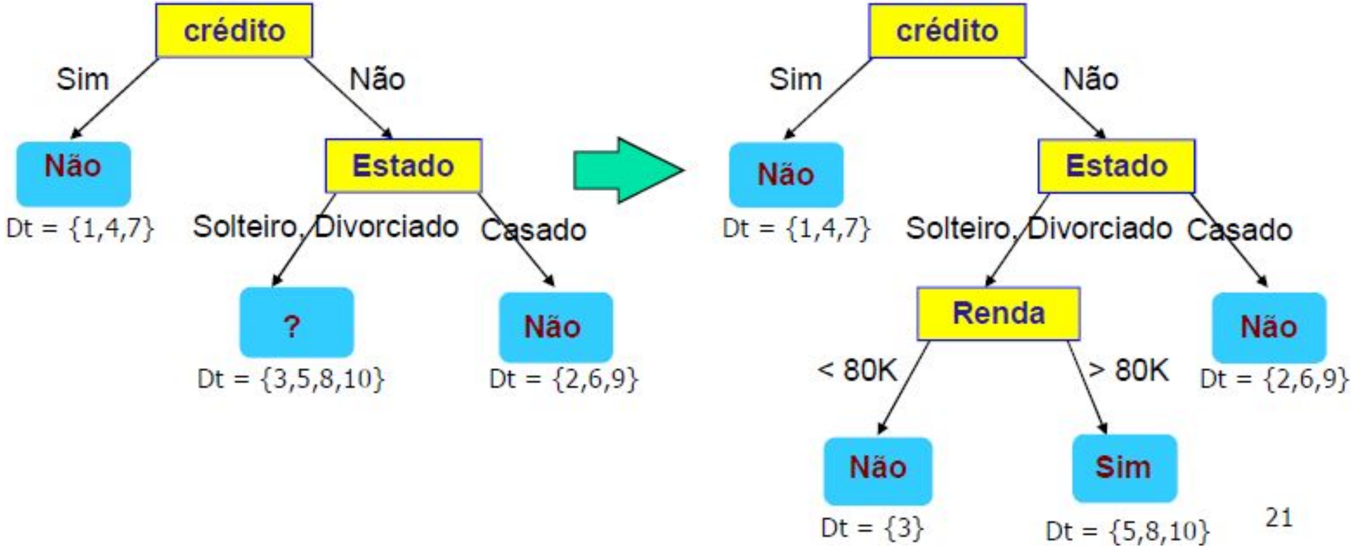




Splitting without criteria

Hunt's algorithm

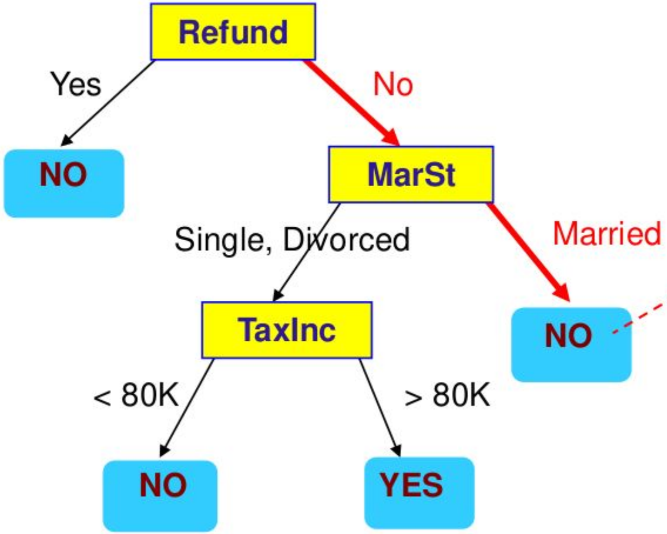
Id	Crédito	Estado Civil	Renda	Deve
1	Sim	Solteiro	125K	Não
2	Não	Casado	100K	Não
3	Não	Solteiro	70K	Não
4	Sim	Casado	120K	Não
5	Não	Divorciado	95K	Sim
6	Não	Casado	60K	Não
7	Sim	Divorciado	220K	Não
8	Não	Solteiro	85K	Sim
9	Não	Casado	75K	Não
10	Não	Solteiro	90K	Sim



Apply Model to Test Data

Hunt’s algorithm

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Assign Cheat to “No”

Design Issue of Decision Tree Induction

How should the training records be split?

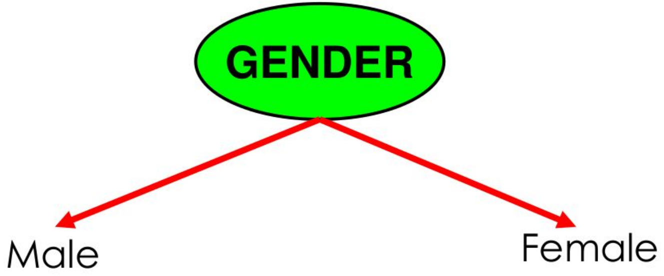
- **Which attribute test condition works better to classify the records?**
- What is the objective measures for **evaluating the goodness** of each test condition?

How should the splitting procedure **stop**?

- One strategy is to **continue expanding a node until all the records belong to the same class** or all the records have identical attributes values.
- Other criteria can also be imposed to allow the tree-growing procedure to **terminate earlier**.

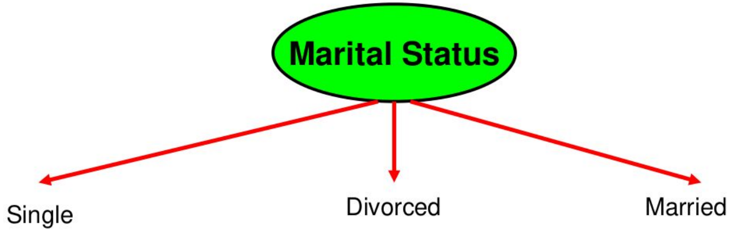
Methods for Expressing Attribute Test Condition

a) Binary Attributes → generates two possible outcomes (binary split)

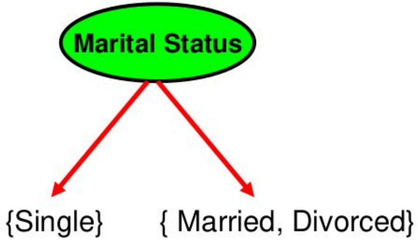


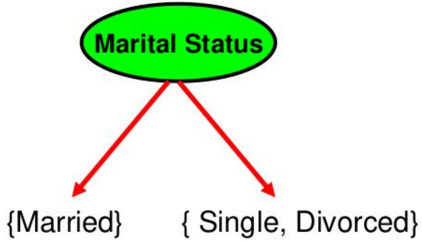
Methods for Expressing Attribute Test Condition

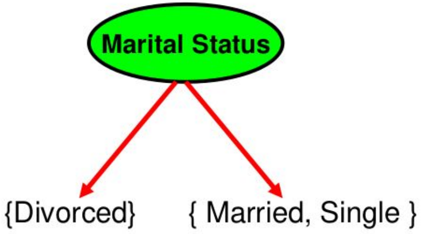
b) Nominal Attributes : Multiway split



b) Nominal Attributes : Binary split (eg : in CART)

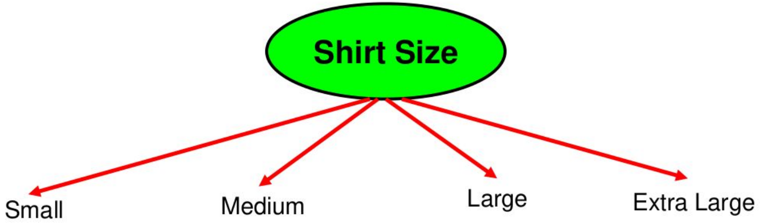




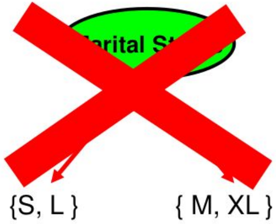


Methods for Expressing Attribute Test Condition

c) Ordinal Attributes : Multiway split

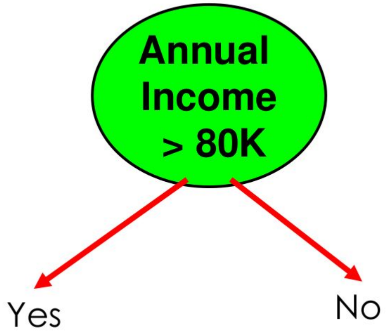


c) Ordinal Attributes : Binary split – as long as it does not violate the order property of the attribute values.

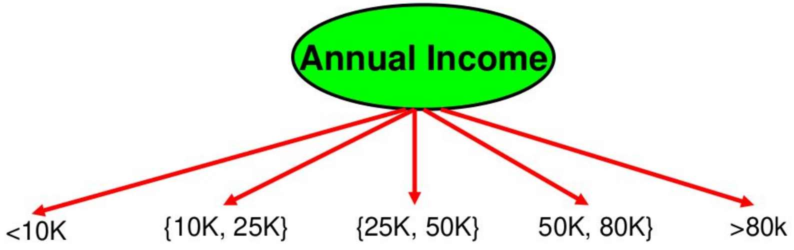


Methods for Expressing Attribute Test Condition

d) **Continuous Attributes** → Binary split



d) **Continuous Attributes** : Multiway split



Measures for selecting the Best Split

Baseada na ideia que:

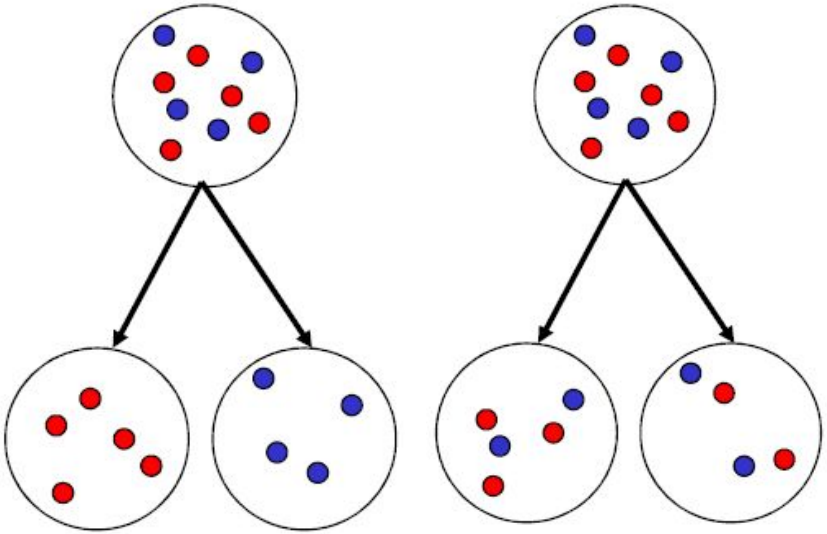
- Quanto mais balanceadas as classes em uma partição, pior
- A partição mais útil é aquela em que todos os exemplos das partes pertencem a uma mesma classe

Necessário uma medida de (im)pureza

$$\text{Entropia}(t) = -\sum_{i=1}^{c} p(i \mid t) \log_2 p(i \mid t)$$

$$\text{Gini\_Index}(t) = 1 - \sum_{i=1}^{c} [p(i \mid t)]^2$$

$$\text{Erro\_Class}(t) = 1 - \max_{i \in \{1,\ldots,c\}} [p(i \mid t)]$$



Measures for selecting the Best Split

These algorithm usually employ a **greedy strategy** making a series of locally optimal decisions about which attribute to use for partitioning the data

- Prefere nós com distribuição mais homogênea (pura) de classes
- Necessário uma medida de (im)pureza

$$Entropy(t) = -\sum_{i=0}^{c-1} p(i|t)\log_2 p(i|t)$$

$$Gini(t) = 1 - \sum_{i=0}^{c-1} [p(i|t)]^2$$

$$Classification\ error(t) = 1 - \max_i [p(i|t)]$$

C0: 5 C1: 5	C0: 9 C1: 1
Muito heterogênea	**Muito homogênea**
Alto grau de impureza	**Baixo grau de impureza**

Measure of impurity (I)

Parent Node	Count
Class = 0	4
Class = 1	14

Node N1	Count
Class = 0	0
Class = 1	6

Node N2	Count
Class = 0	1
Class = 1	5

Node N3	Count
Class = 0	3
Class = 1	3

Node N1:

$$Gini = 1 - (0/6)^2 - (6/6)^2 = 0$$

$$Entropy = -(0/6)\log_2(0/6) - (6/6)\log_2(6/6) = 0$$

$$Error = 1 - \max[(0/6),(6/6)] = 0$$

Measure of impurity (I)

Parent Node	Count
Class = 0	4
Class = 1	14

Node N1	Count
Class = 0	0
Class = 1	6

Node N2	Count
Class = 0	1
Class = 1	5

Node N3	Count
Class = 0	3
Class = 1	3

Node N2:

$$Gini = 1 - (1/6)^2 - (5/6)^2 = 0.278$$

$$Entropy = -(1/6)\log_2(1/6) - (5/6)\log_2(5/6) = 0.65$$

$$Error = 1 - \max[(1/6),(5/6)] = 0.167$$

Measure of impurity (I)

Parent Node	Count
Class = 0	4
Class = 1	14

Node N1	Count
Class = 0	0
Class = 1	6

Node N2	Count
Class = 0	1
Class = 1	5

Node N3	Count
Class = 0	3
Class = 1	3

Node N3:

$$Gini = 1 - (3/6)^2 - (3/6)^2 = 0.5$$

$$Entropy = -(3/6)\log_2(3/6) - (3/6)\log_2(3/6) = 1$$

$$Error = 1 - \max[(3/6),(3/6)] = 0.5$$

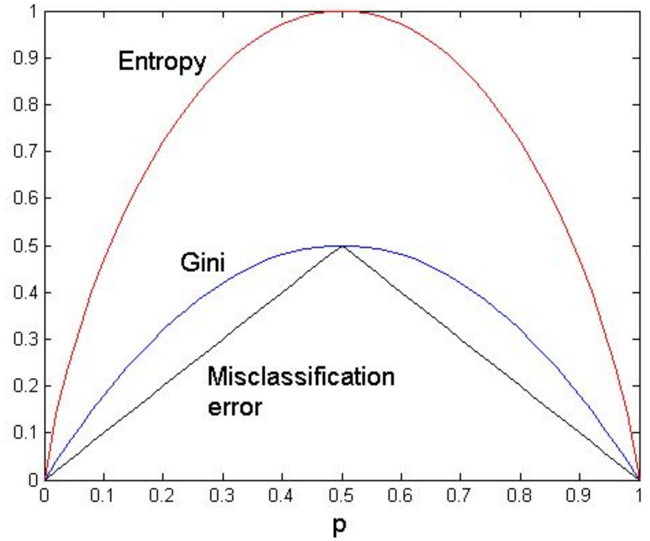
Measure of impurity (I)

Parent Node	Count
Class = 0	4
Class = 1	14

Node N1	Count
Class = 0	0
Class = 1	6
Gini	0
Entropy	0
Error	0

Node N2	Count
Class = 0	1
Class = 1	5
Gini	0.278
Entropy	0.650
Error	0.167

Node N3	Count
Class = 0	3
Class = 1	3
Gini	0.5
Entropy	1
Error	0.5



N1 has the lowest impurity value, followed by N2 and N3

Split with criteria

To determine how well a test condition performs, we need to **compare** the degree of impurity of the parent node (**before splitting**) and the child node (**after splitting**).

The larger the different, the better the test condition

1. **Weighted average of impurity**
2. **Information Gain**

The gain Δ, is a criterion that can be used to determine the goodness of a split.

$$\Delta = I(parent) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j)$$

Weighted Average Impurity

Where:

- **I(.)** is the impurity measure of a given node
- **N** is the total number of records at the parent node
- **k** is the number of attributes value (class)
- **$N(v_j)$** is the number of records associated with the child node v_j

Splitting with weighted average of impurity criteria

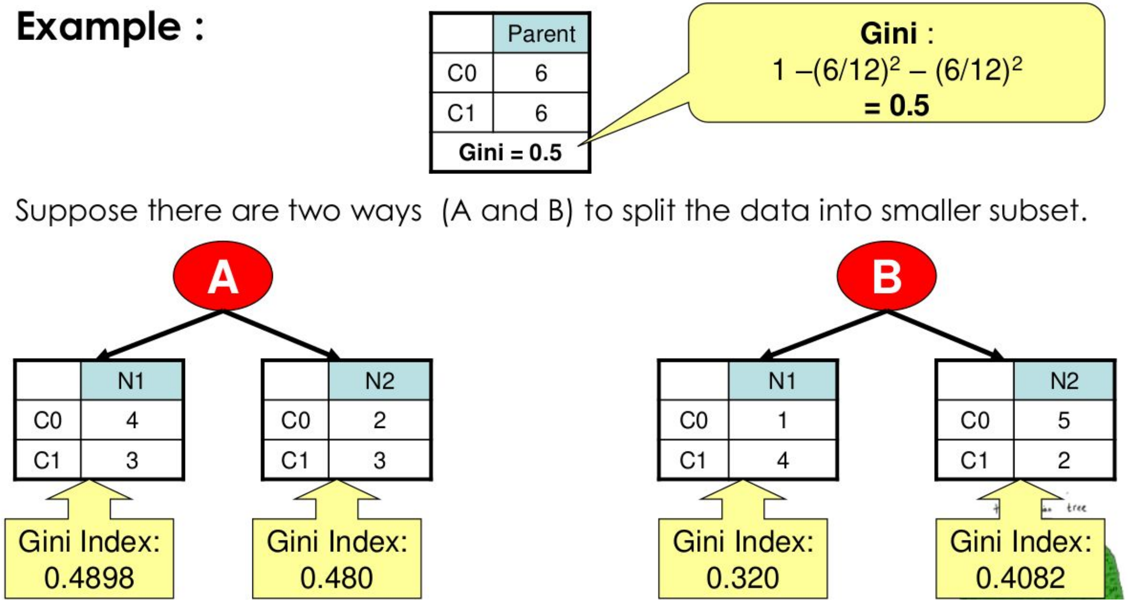
Since I(parent) is the same for all test condition, **maximizing the gain** is equivalent to **minimizing the weighted average impurity** measure of the child nodes.

$$\Delta = I(parent) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j)$$

Weighted Average Impurity

$$\text{Gini}_{divisão} = \sum_{t=1}^{k} \frac{N(v_t)}{N} \text{Gini}(v_t)$$

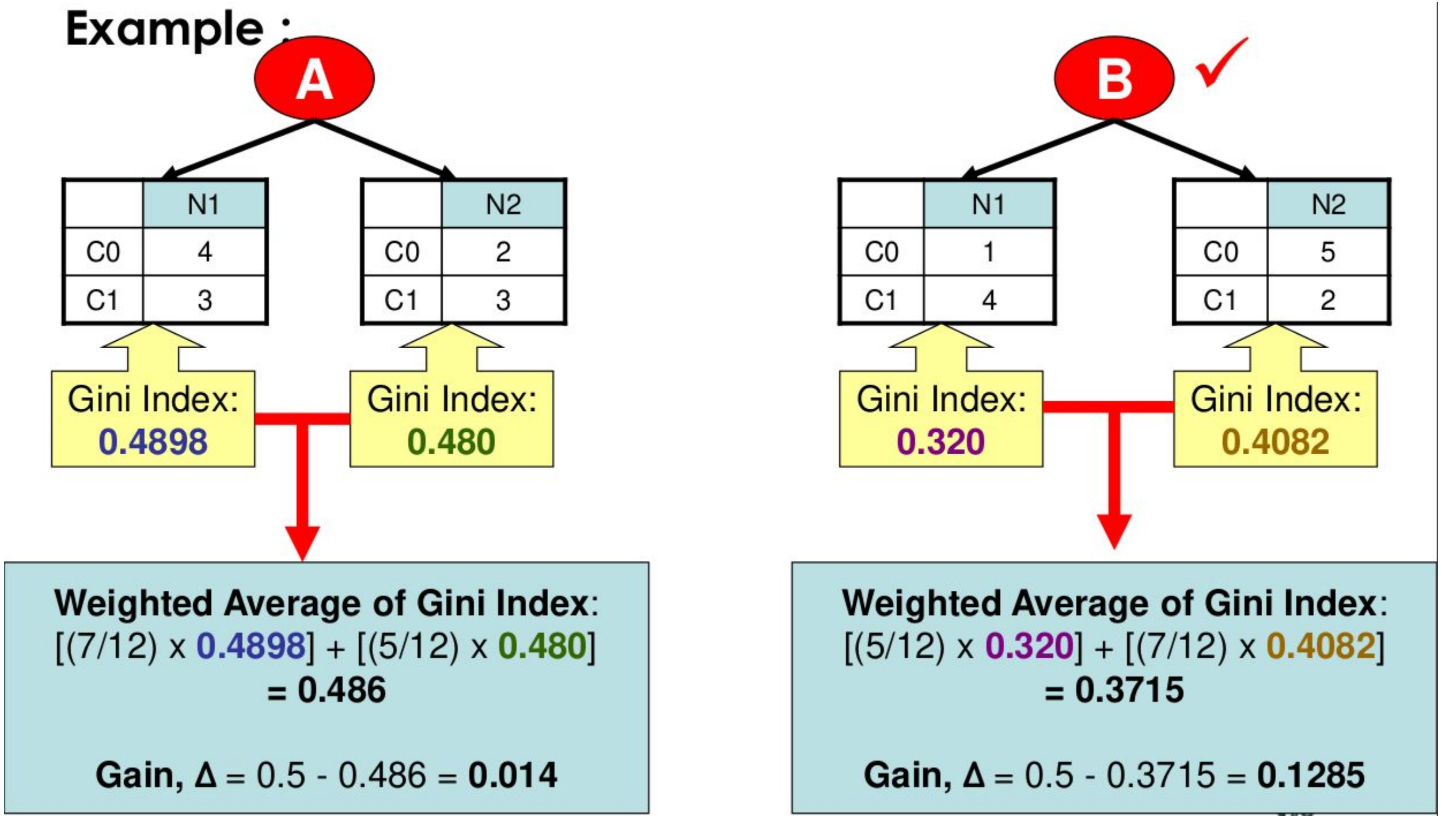
Splitting with weighted average of impurity criteria



Which one is a better split??

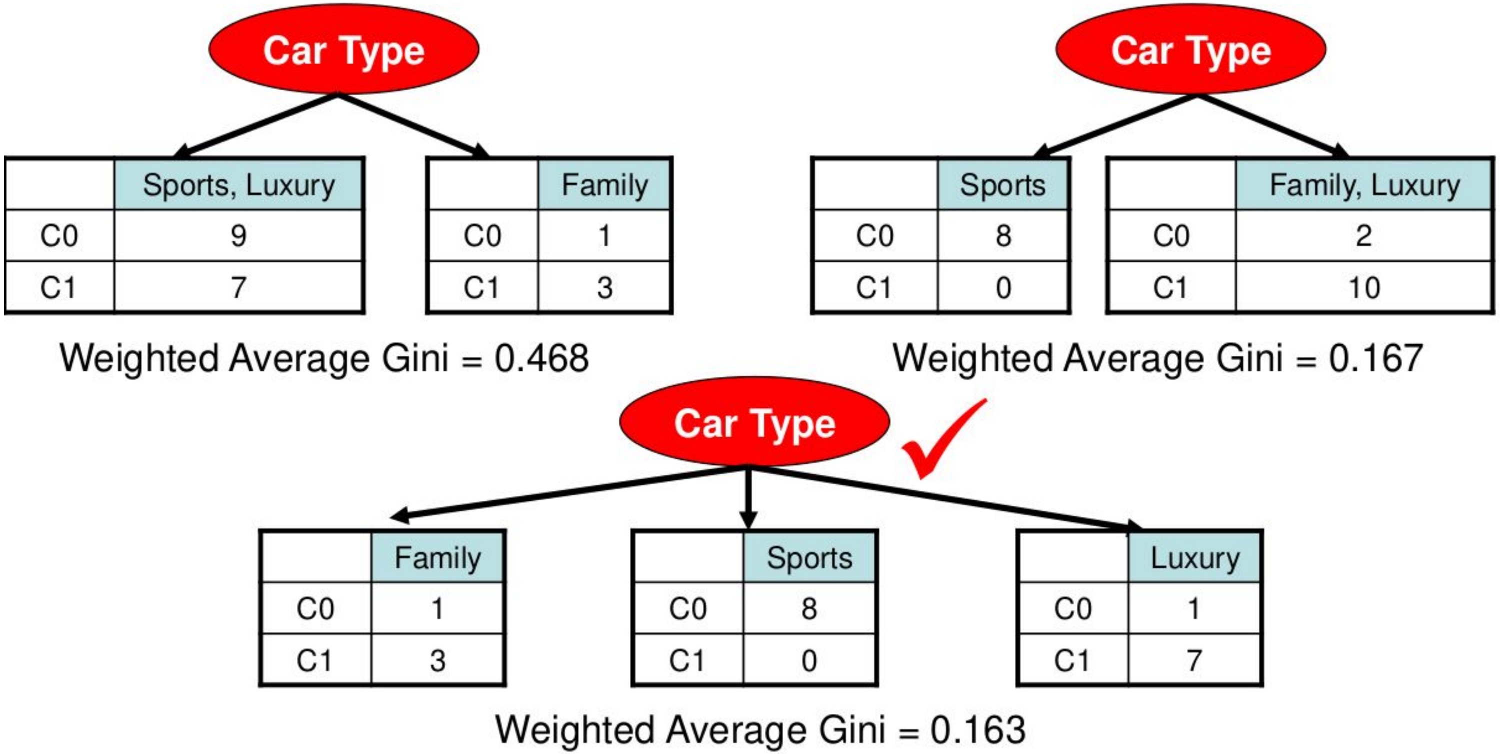
Compute the weighted of the Gini index of both attributes

Splitting with weighted average of impurity criteria



Splitting with weighted average of impurity criteria

Example : Which split is better? Binary or Multi-way splits.



Split with Information Gain (IG) criteria

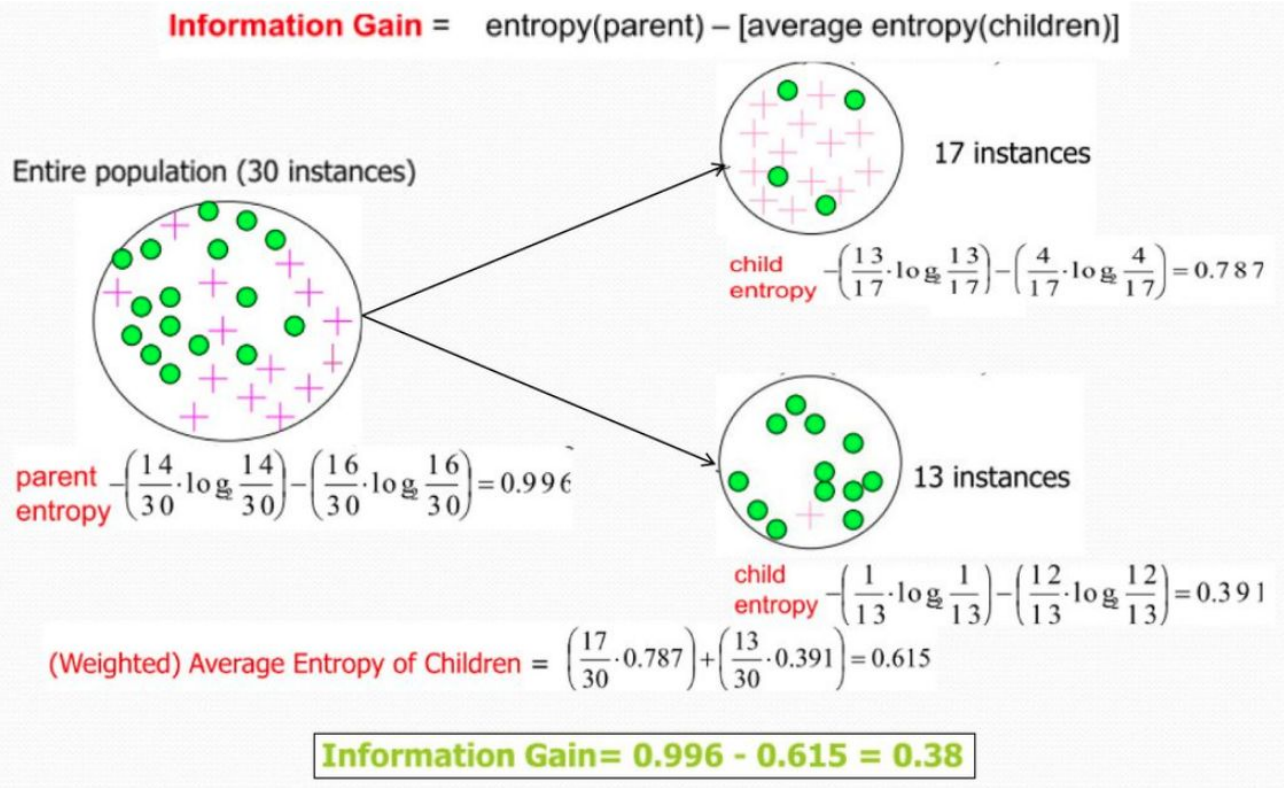
When **entropy is used as the impurity measure**, the difference in entropy is known as the **Information Gain Ratio (IGR) or Information Gain (IG)**

- Decision tree build using entropy tend to be quite bushy. Bushy tree with many multi-way split are undesirable as these splits lead to small numbers of records in each node.

$$\boxed{\Delta = I(parent) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j)}$$

Weighted Average Impurity

Split with Information Gain (IG) criteria



Splitting Continuous Attributes (using Gini)

A brute-force method is used to find the best split position (v) for a continuous attribute (eg: Annual Income).

Class	No	No	No	Yes	Yes	Yes	No	No	No	No
Annual Income (sorted)	60	70	75	85	90	95	100	120	125	220

Split position (mid points)	55		65		72		80		87		92		97		110		122		172		230	
	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0

- To reduce complexity, the training records are sorted based on the annual income.
- Candidate split positions (v) are identified by taking the midpoints between two adjacent sorted values.

	categorical	categorical	continuous	class
Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Splitting Continuous Attributes (using Gini)

We then compute the Gini index for each candidate and choose the one that gives the lowest value.

Class	No	No	No	Yes	Yes	Yes	No	No	No	No
Annual Income (sorted)	60	70	75	85	90	95	100	120	125	220

Split position (mid points)	55		65		72		80		87		92		97		110		122		172		230	
	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini	0.420		0.400		0.375		0.343		0.417		0.400		0.300		0.343		0.375		0.4		0.420	

Splitting Continuous Attributes (using Gini)

Class	No	No	No	Yes	Yes	Yes	No	No	No	No
Annual Income (sorted)	60	70	75	85	90	95	100	120	125	220

Split position (mid points)	55		65		72		80		87		92		97		110		122		172		230	
	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini	0.420		0.400		0.375		0.343		0.417		0.400		0.300		0.343		0.375		0.4		0.420	

Primeiro Candidato: **x** = 55
< 55
  Classe sim: 0
  Classe não: 0
  Gini N1 = 0
> 55
  Classe sim: 3
  Classe não: 7
  Gini N2 = 0.420
$Gini_d$ = 0x0 + 1x0.420 = 0.420

Segundo Candidato: **x** = 65
Atualiza distribuição do último candidato
< 65
  Classe sim: 0
  Classe não: 1 (0 + 1)
  Gini N1 = ?
> 65
  Classe sim: 3
  Classe não: 6 (7 – 1)
  Gini N2 = ?
$Gini_d$ = 0.400