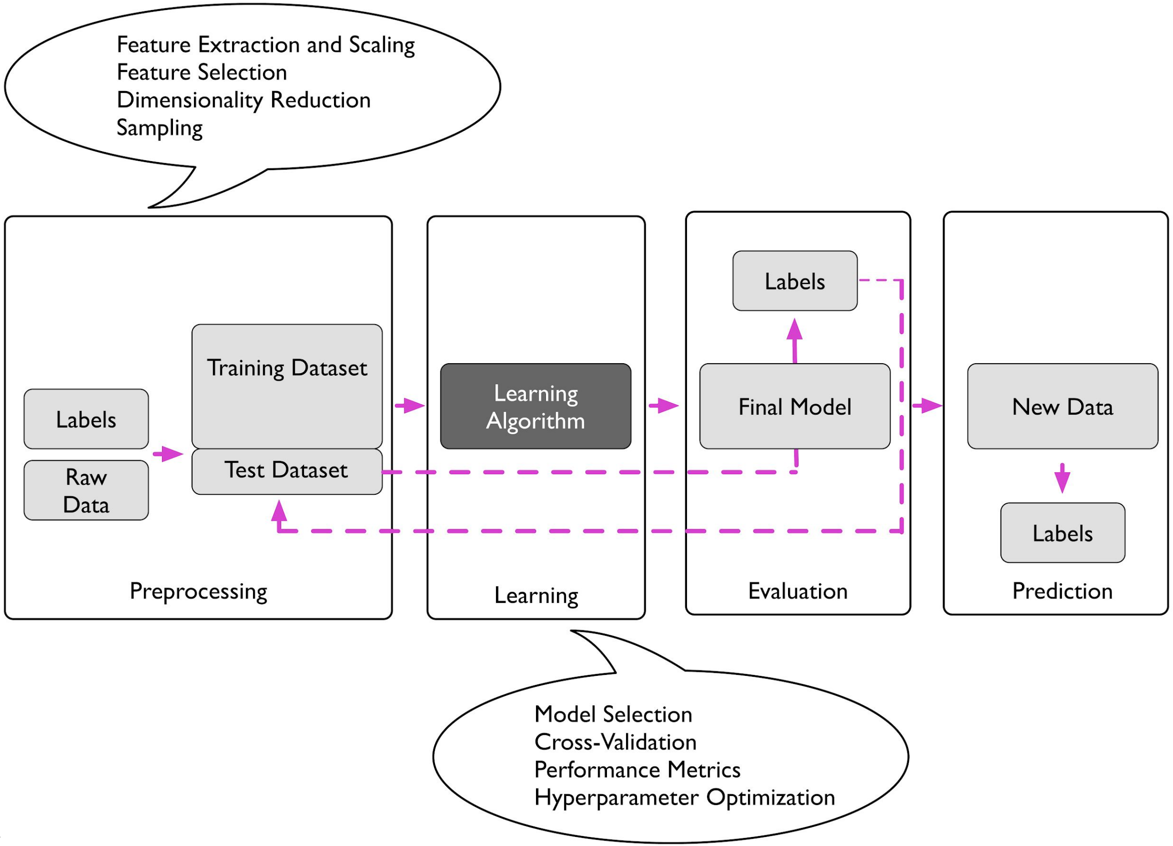
Lecture 12

Decision trees

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

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

Machine learning pipeline

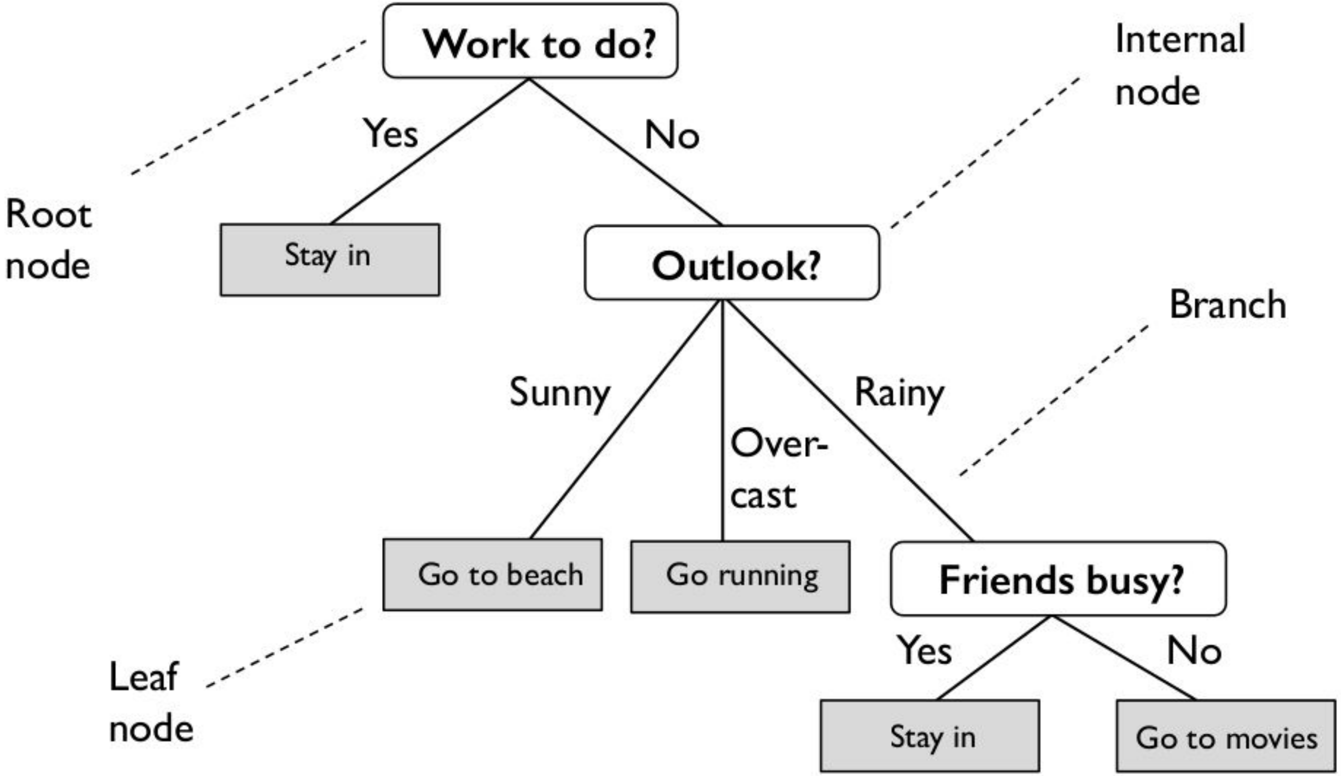


Python Machine Learning by Sebastian Raschka

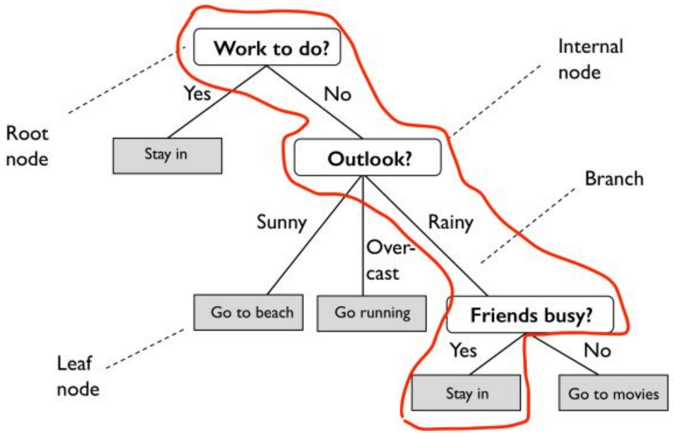
Topics

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 _______ ___ __________

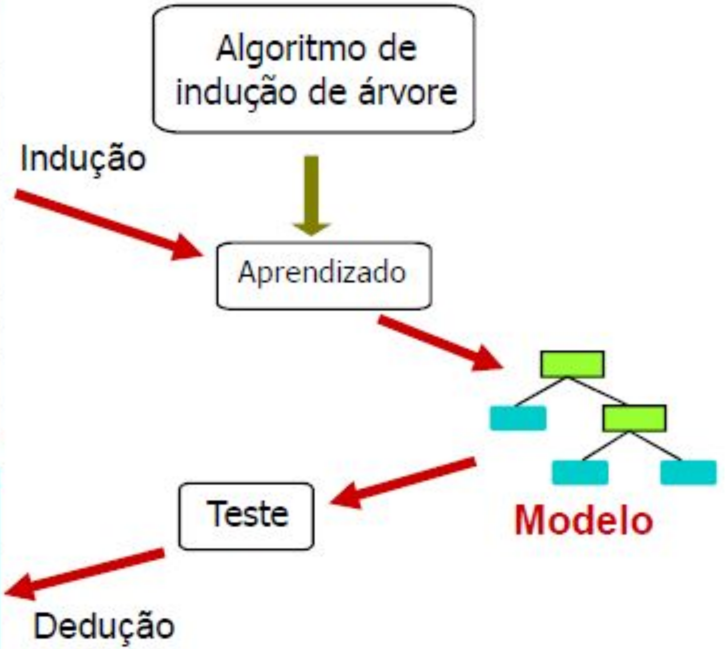
Decision tree pipeline

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	?

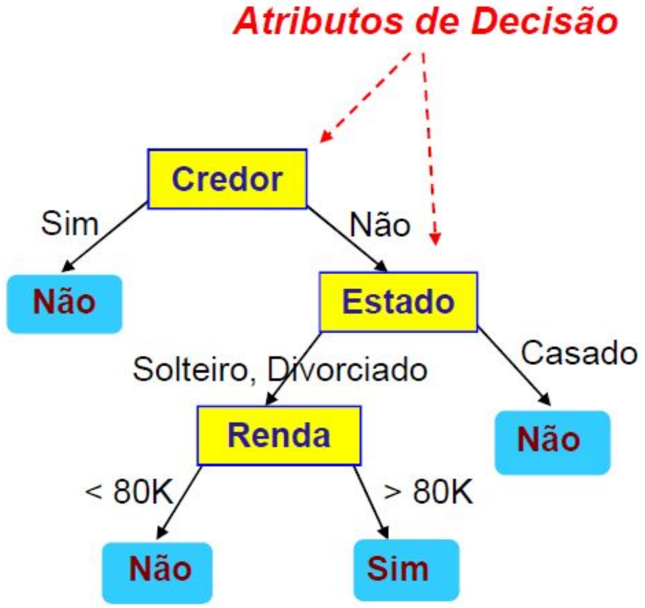


Training

- 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

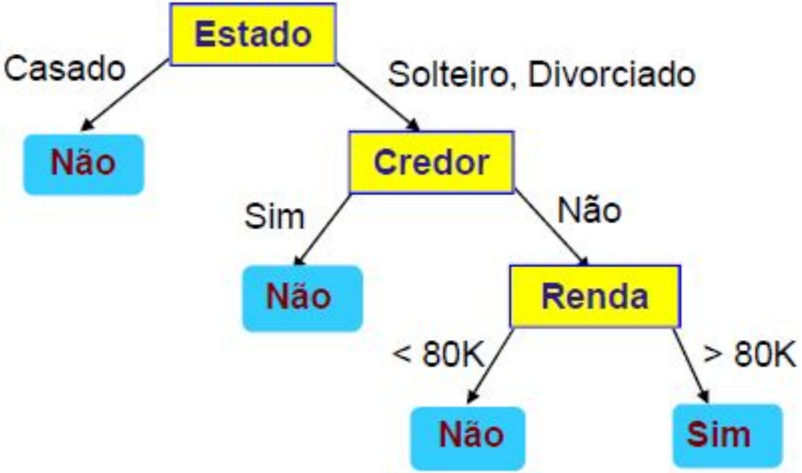
Dados de Treinamento



Modelo: Árvore de Decisão

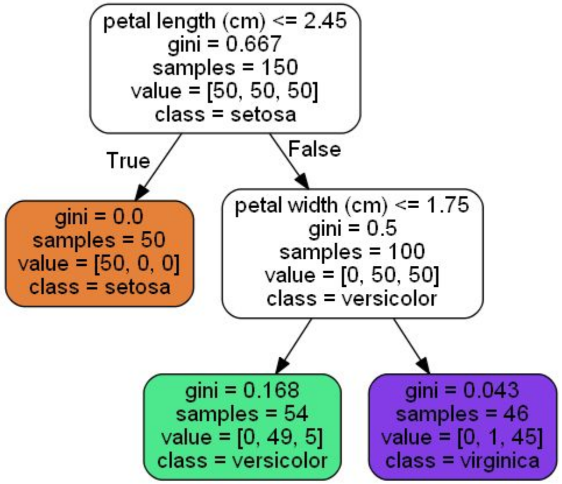
Training

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)



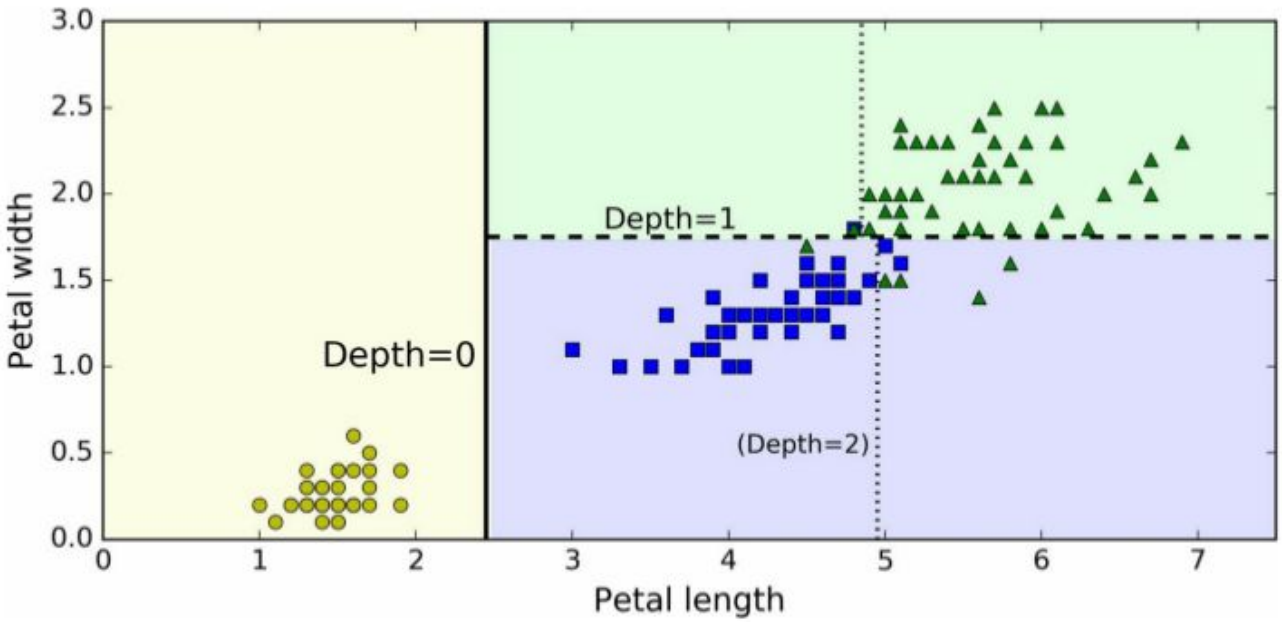
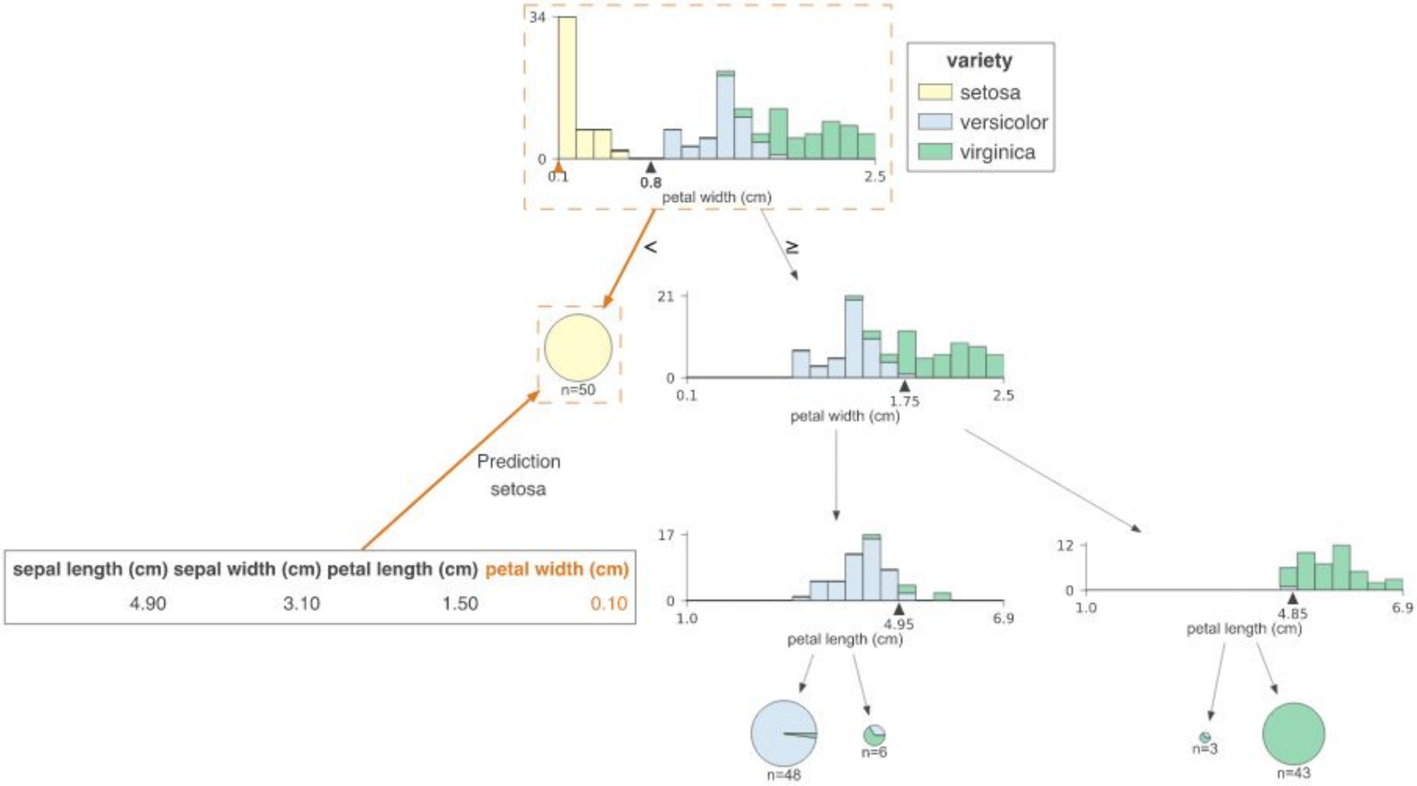


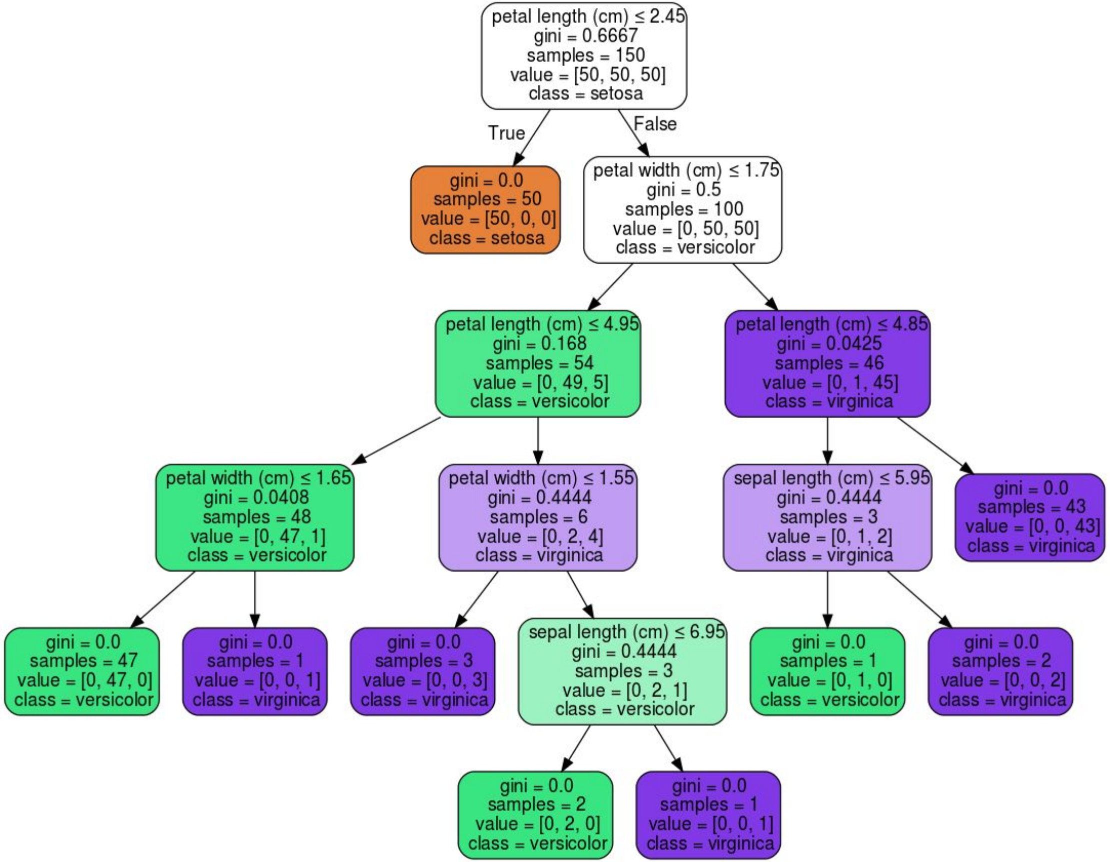
Figure 6-2. Decision Tree decision boundaries

Decision tree on Iris dataset (2 features)

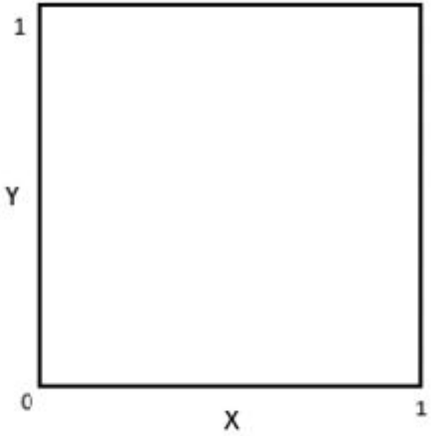


https://explained.ai/decision-tree-viz/images/samples/iris-TD-3-X.svg

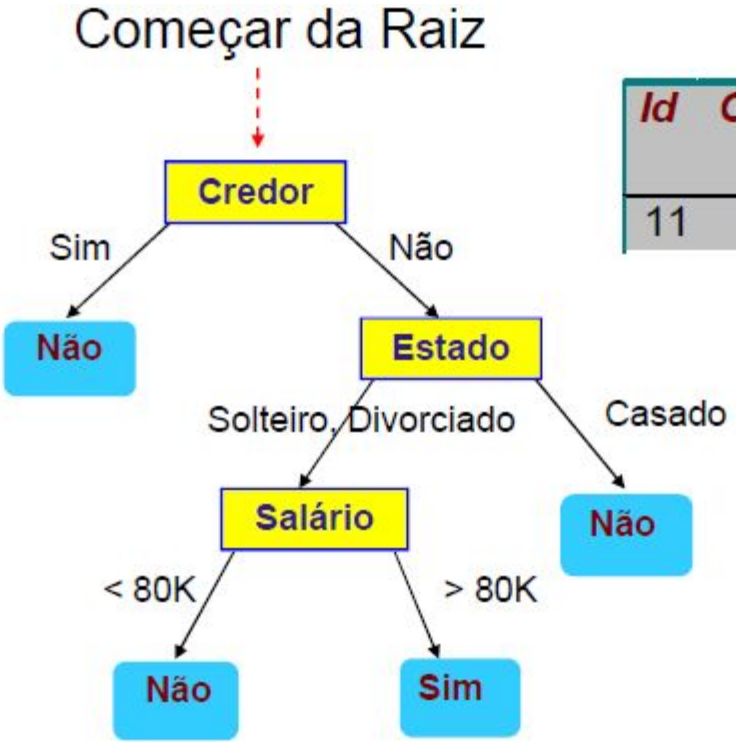
Decision tree on Iris dataset (4 features)



Growing depth process



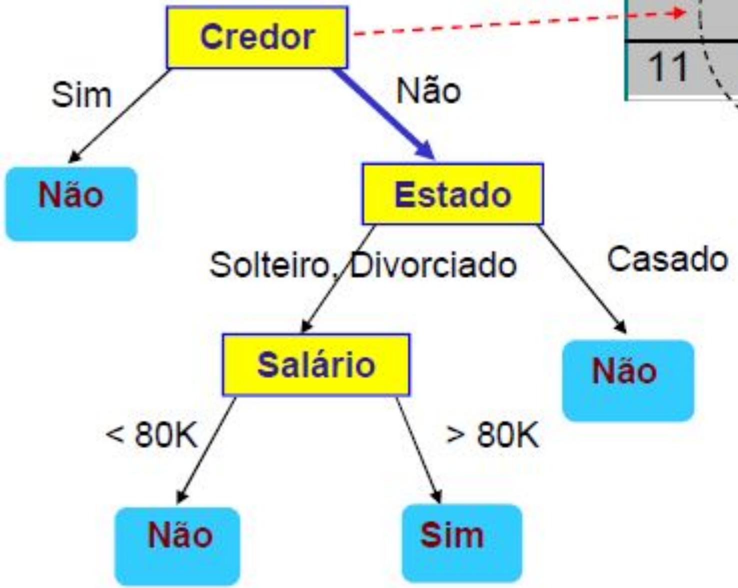
Inference



Dados de Teste

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

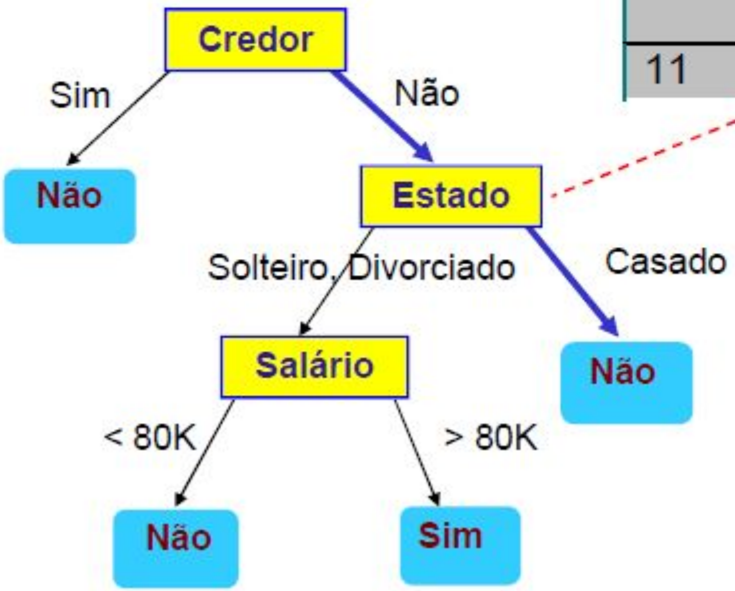
Inference



Dados de Teste

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

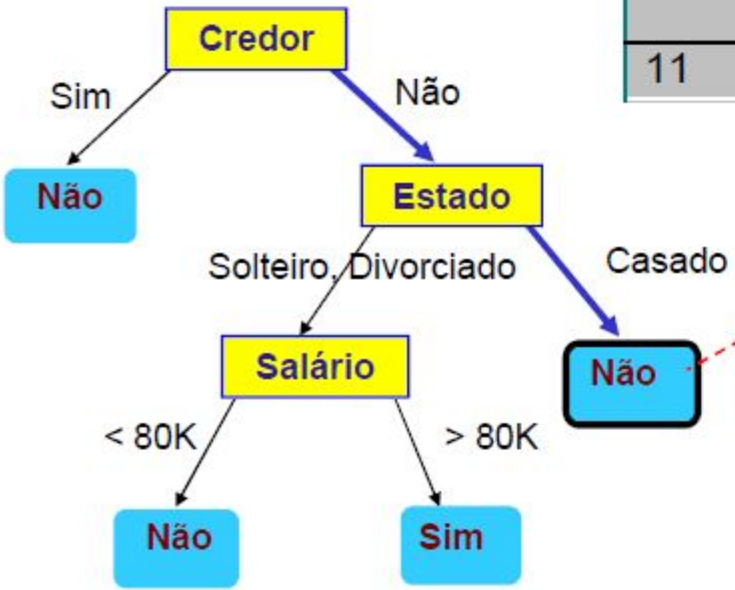
Inference



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

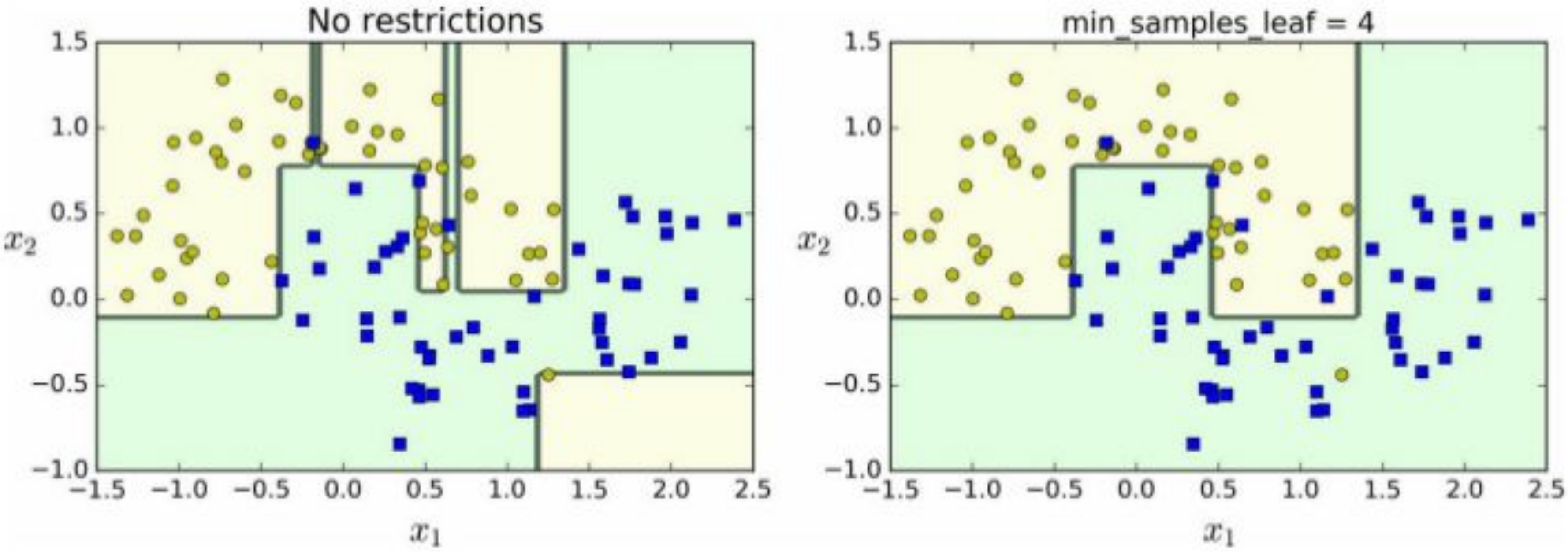


Figure 6-3. Regularization using min_samples_leaf

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

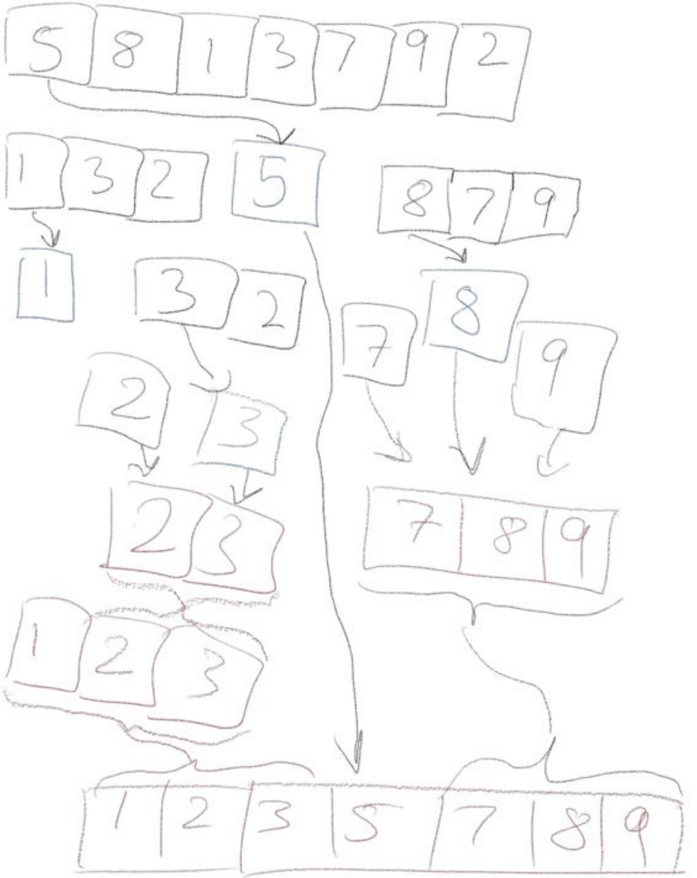
What does this function do?

```
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




```python
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	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	Θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	Θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)

Decision Tree in Pseudocode

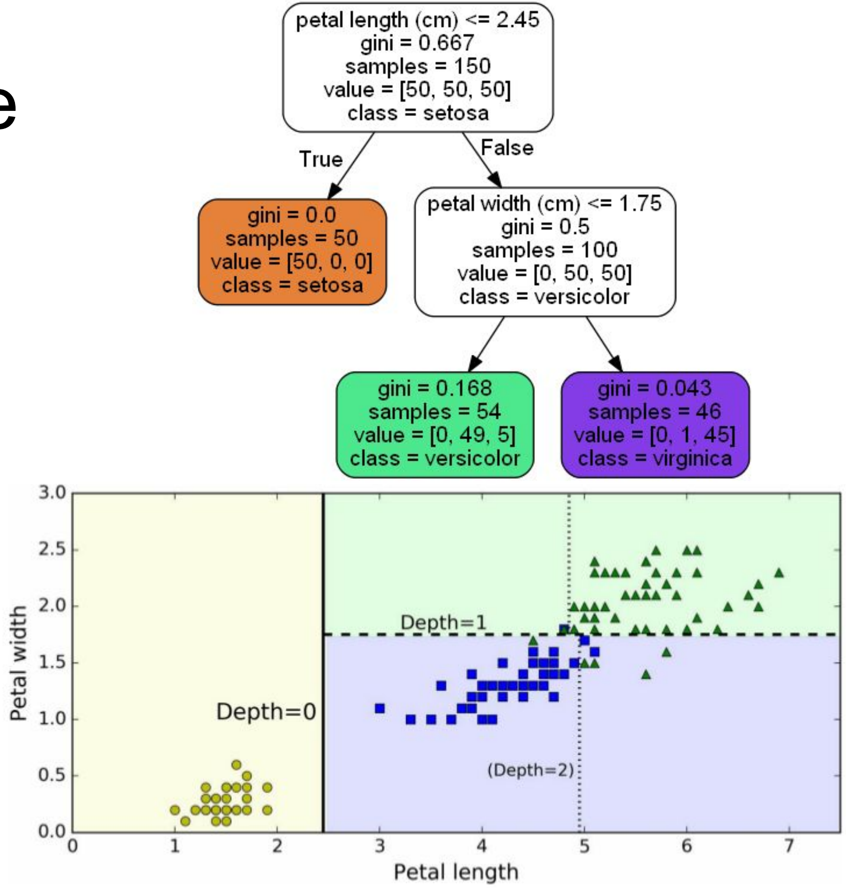


Figure 6-2. Decision Tree decision boundaries

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($\mathcal{D}_1$))

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 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