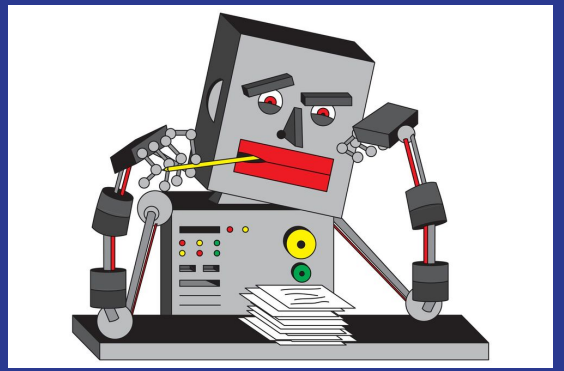
APRENDIZADO DE MÁQUINA

Prof. Eduardo Bezerra
ebezerra@cefet-rj.br
CEFET/RJ - PPCIC



ÁRVORES DE DECISÃO

Introdução

Aprendizagem em árvores de decisão

- **Indução de árvores de decisão:** uma das formas mais simples (e mais bem sucedidas) de aprendizagem automática para a tarefa de classificação.
- **Árvore de decisão**: toma como entrada um objeto ou situação descritos por um conjunto de **atributos** (e de seus valores) e retorna uma decisão -- valor de saída previsto.

Árvores de decisão

- Um agente que usa uma árvore de decisão chega a uma decisão executando uma **sequência** de testes.
  - Cada **nó interno** da árvore corresponde a um teste sobre o valor de um dos atributos.
  - As ramificações a partir de cada nó interno são identificadas (rotuladas) com os valores possíveis do teste.
  - Cada **nó folha** especifica o **valor** a ser retornado se aquela folha for alcançada.

Árvores de decisão para classificação

Árvores de decisão - exemplo

Exemplo: seja construir um classificador para decidir se um cliente em potencial deve esperar por uma mesa em um restaurante.

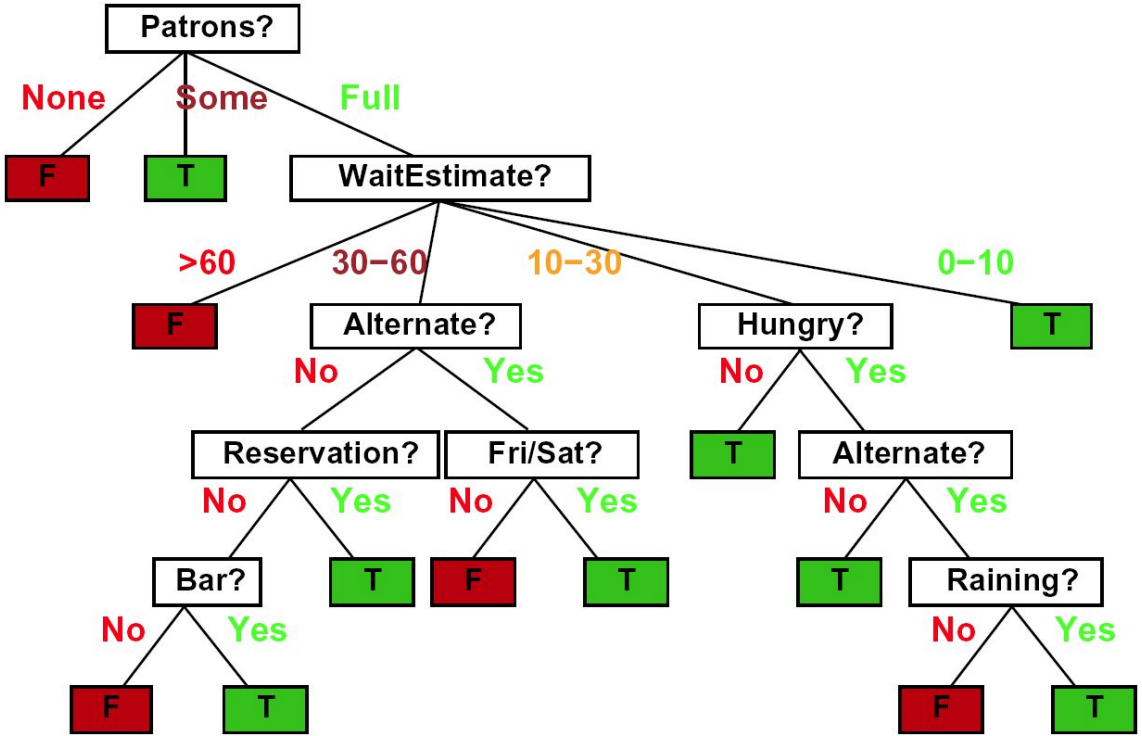
Conjunto de dados com os seguintes atributos (características):

1. ***Alternate***: há um restaurante alternativo na redondeza?
2. ***Bar***: existe um bar confortável onde se pode esperar a mesa?
3. ***Fri/Sat***: hoje é sexta ou sábado ?
4. ***Hungry***: estou com fome?
5. ***Patrons***: número de pessoas no restaurante (***None***, ***Some***, ***Full***)
6. ***Price***: faixa de preços ($, $$, $$$)
7. ***Raining***: está a chover?
8. ***Reservation***: temos reserva?
9. ***Type***: tipo do restaurante (***French***, ***Italian***, ***Thai***, ***Burger***)
10. ***WaitEstimate***: tempo de espera estimado (0-10, 10-30, 30-60, >60)

Árvores de decisão - exemplo

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*T*	*F*	*F*	*T*	*Some*	*$$$*	*F*	*T*	*French*	*0–10*	*T*
X_2	*T*	*F*	*F*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*30–60*	*F*
X_3	*F*	*T*	*F*	*F*	*Some*	*$*	*F*	*F*	*Burger*	*0–10*	*T*
X_4	*T*	*F*	*T*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*10–30*	*T*
X_5	*T*	*F*	*T*	*F*	*Full*	*$$$*	*F*	*T*	*French*	*>60*	*F*
X_6	*F*	*T*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Italian*	*0–10*	*T*
X_7	*F*	*T*	*F*	*F*	*None*	*$*	*T*	*F*	*Burger*	*0–10*	*F*
X_8	*F*	*F*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Thai*	*0–10*	*T*
X_9	*F*	*T*	*T*	*F*	*Full*	*$*	*T*	*F*	*Burger*	*>60*	*F*
X_{10}	*T*	*T*	*T*	*T*	*Full*	*$$$*	*F*	*T*	*Italian*	*10–30*	*F*
X_{11}	*F*	*F*	*F*	*F*	*None*	*$*	*F*	*F*	*Thai*	*0–10*	*F*
X_{12}	*T*	*T*	*T*	*T*	*Full*	*$*	*F*	*F*	*Burger*	*30–60*	*T*

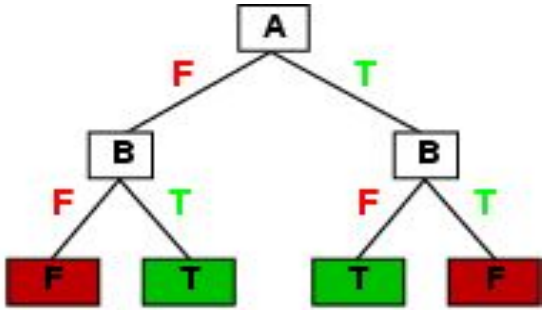
Árvore de decisão - exemplo



Expressividade

- Qualquer função booleana pode ser escrita como uma árvore de decisão

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F

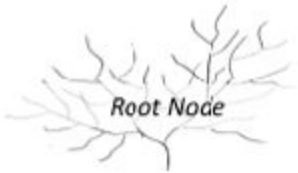


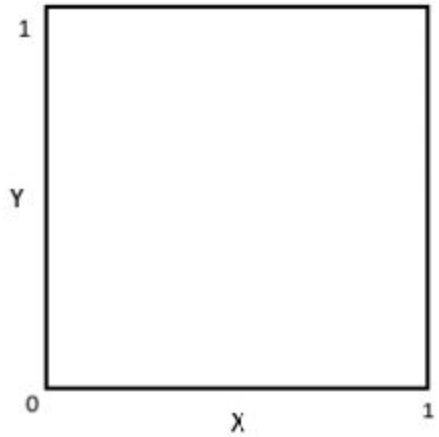
- Trivialmente, há uma árvore de decisão consistente para qualquer conjunto de treinamento: um caminho para cada exemplo.
  - Mas, isso geraria uma representação exponencialmente grande!
  - Devemos procurar por árvores de decisão mais compactas.

Algoritmo DTL (*Decision Tree Learning*)

```
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root test best
        for each value v_i of best do
            examples_i ← {elements of examples with best = v_i}
            subtree ← DTL(examples_i, attributes − best, MODE(examples))
            add a branch to tree with label v_i and subtree subtree
        return tree
```

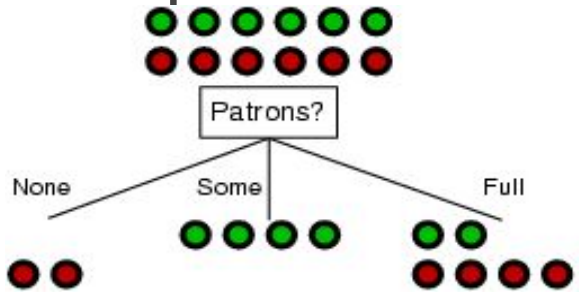
Algoritmo DTL (*Decision Tree Learning*)

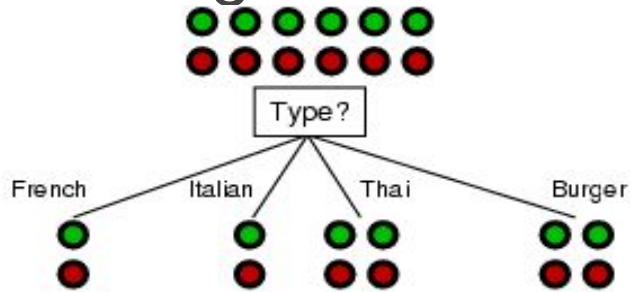




Escolha de atributos (CHOOSE-ATTRIBUTE)

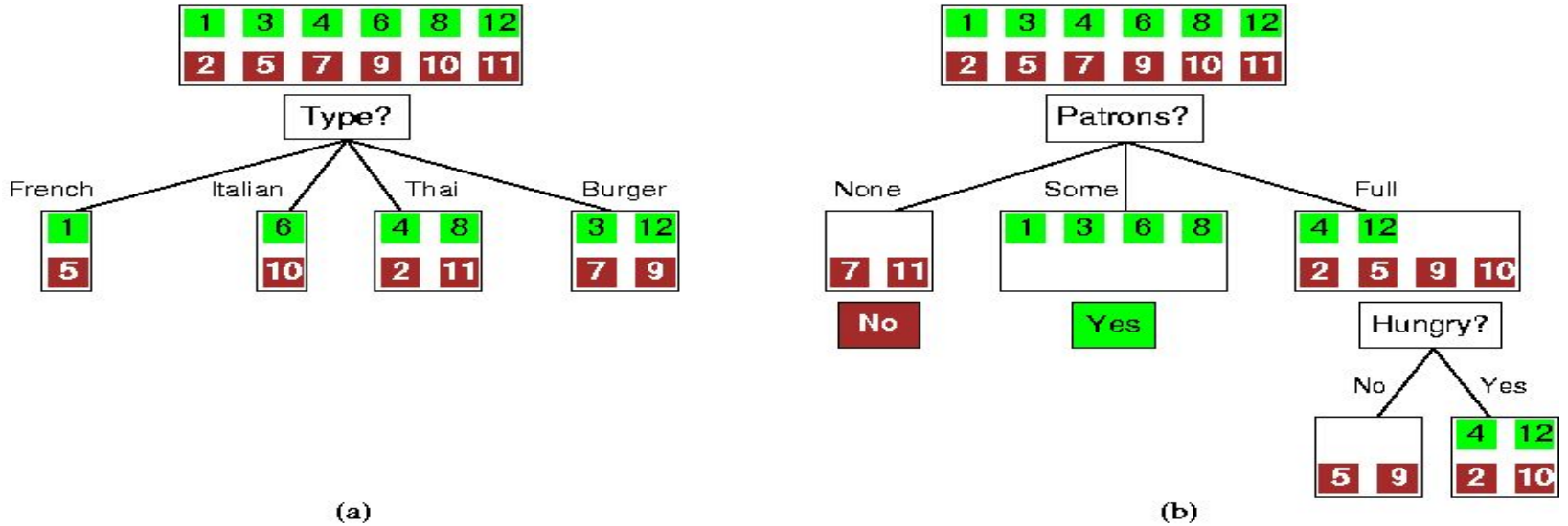
- Ideia: um bom atributo é aquele que divide os exemplos em subconjuntos que (preferivelmente) são “todos positivos" ou ”todos negativos"





- *e.g., Patrons* é melhor do que *Type*.

Escolha de atributos (CHOOSE-ATTRIBUTE)



Como definir o que é um atributo melhor?

- A escolha de atributos deve **minimizar** a profundidade da árvore de decisão.
  - Devemos escolher prioritariamente o atributo que vá o mais longe possível na classificação exata de exemplos;
  - Um atributo perfeito dividiria os exemplos em subconjuntos de mesma classe (i.e., todos positivos ou todos negativos).
- Solução: medir os atributos a partir da quantidade esperada de informações fornecida por ele.

Como definir o que é um atributo melhor?

- O atributo “patrons” não é perfeito, mas é bastante bom; o atributo “type” é completamente inútil.
- Precisamos definir uma medida formal para quantificar as noções de “bastante bom” e de “completamente inútil”.
- A medida deve ter seu valor máximo quando o atributo for perfeito e seu valor mínimo quando o atributo for inútil.
- Essa formalização é possível por meio do uso de um conceito proveniente da Teoria da Informação...

Teoria da Informação

- A Teoria da Informação estuda de que forma uma determinada informação pode ser codificada em bits.
  - um bit é o suficiente para responder a uma pergunta sim/não (como o lançamento de uma moeda)

Entropia

- Dada uma distribuição de probabilidades para uma variável aleatória discreta V com n valores, se cada valor possível v_i de V tem probabilidade p_i, então a entropia ***H*** dessa distribuição é dada por:

$$H(V) = -\sum_{v_i \in V} p_i \log_2 p_i$$

- Por exemplo, no lançamento de uma moeda imparcial:

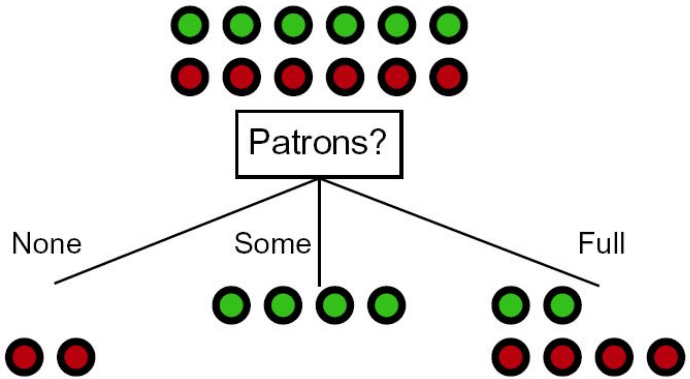
$$H\left(\frac{1}{2}, \frac{1}{2}\right) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1\ bit$$

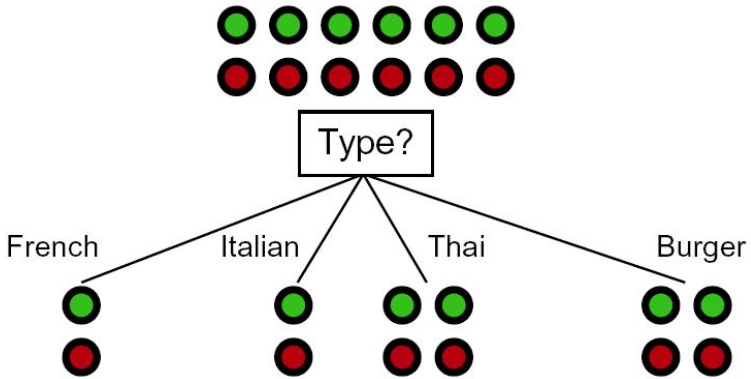
Ganho de Informação *(Information Gain)*

- De volta às árvores de decisão...
  - Qualquer atributo A divide o conjunto de treinamento E em subconjuntos E_1, … , E_v de acordo com seus valores $\mathcal{v}$(A), onde A pode ter |$\mathcal{v}$(A)| valores distintos.
  - Dada uma divisão feita por um atributo A, podemos medir a entropia antes e depois dessa divisão.
    - A diferença é denominada ganho de informação (*information gain*, IG)
    - Problema: há mais de uma distribuição após a divisão!
    - Solução: usar a entropia esperada, usando as quantidades de exemplos em cada classe como pesos.
  - Heurística: escolher o atributo com o maior IG.

Ganho de Informação – exemplo

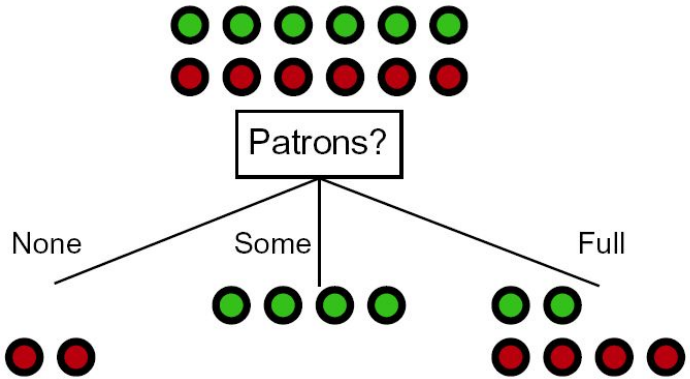
- Quais as entropias, antes e depois das divisões?

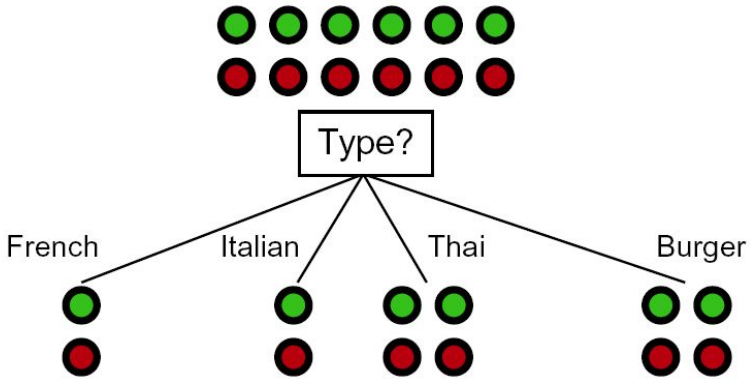




Ganho de Informação – exemplo

- Quais as entropias, antes e depois das divisões?





Ganho de Informação – exemplo

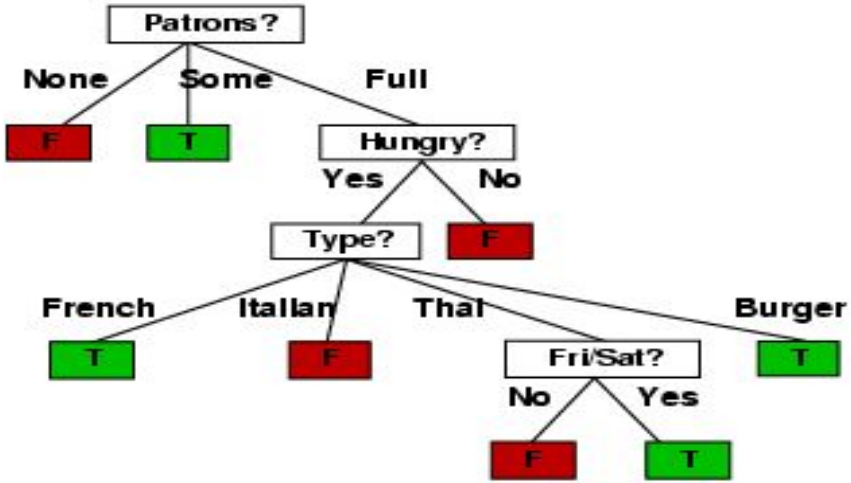
- Para o conjunto de treinamento completo, temos que $H(\frac{6}{12}, \frac{6}{12}) = 1$ bit
- Considerando os atributos *Patrons* e *Type*:

$$IG(Patrons) = 1 - [\frac{2}{12} H(0,1) + \frac{4}{12} H(1,0) + \frac{6}{12} H(\frac{2}{6}, \frac{4}{6})] = .0541 \text{ bits}$$

$$IG(Type) = 1 - [\frac{2}{12} H(\frac{1}{2}, \frac{1}{2}) + \frac{2}{12} H(\frac{1}{2}, \frac{1}{2}) + \frac{4}{12} H(\frac{2}{4}, \frac{2}{4}) + \frac{4}{12} H(\frac{2}{4}, \frac{2}{4})] = 0 \text{ bits}$$

- *De fato, Patrons* possui o maior IG dentre todos os atributos e, portanto, é o primeiro atributo escolhido pelo algoritmo DTL.

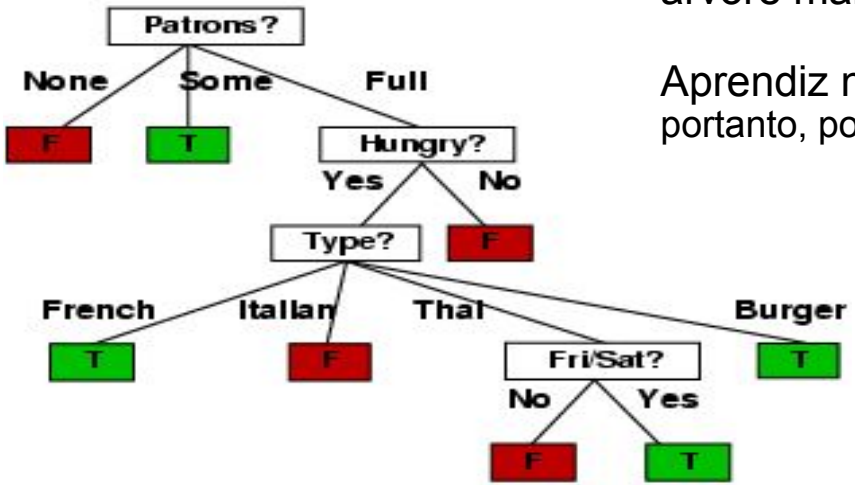
- Árvore de decisão aprendida a partir dos 12 exemplos:



Substancialmente mais simples do que a árvore “completa”;

Não há nenhuma razão para uma solução mais complexa (e.g incluindo os atributos *Rain* e *Res*), pois todos os exemplos já foram classificados.

- Árvore de decisão aprendida a partir dos 12 exemplos:



Com mais exemplos, seria possível induzir uma árvore mais semelhante à árvore original;

Aprendiz nunca viu um exemplo de espera 0-10 portanto, pode cometer um engano...

Árvores de decisão para regressão

Decision Trees for Regression

- In regression tasks, we need an impurity metric that is suitable for continuous variables.
- One possible solution: mean squared errors (MSE).

$$I(t) = \text{MSE}(t) = \frac{1}{m_t} \sum_{i \in D_t} (y^{(i)} - \bar{y}_t)^2$$

$$\bar{y}_t = \frac{1}{m_t} \sum_{i \in D_t} y^{(i)}$$

Python Machine Learning, 2nd ed, pp 340

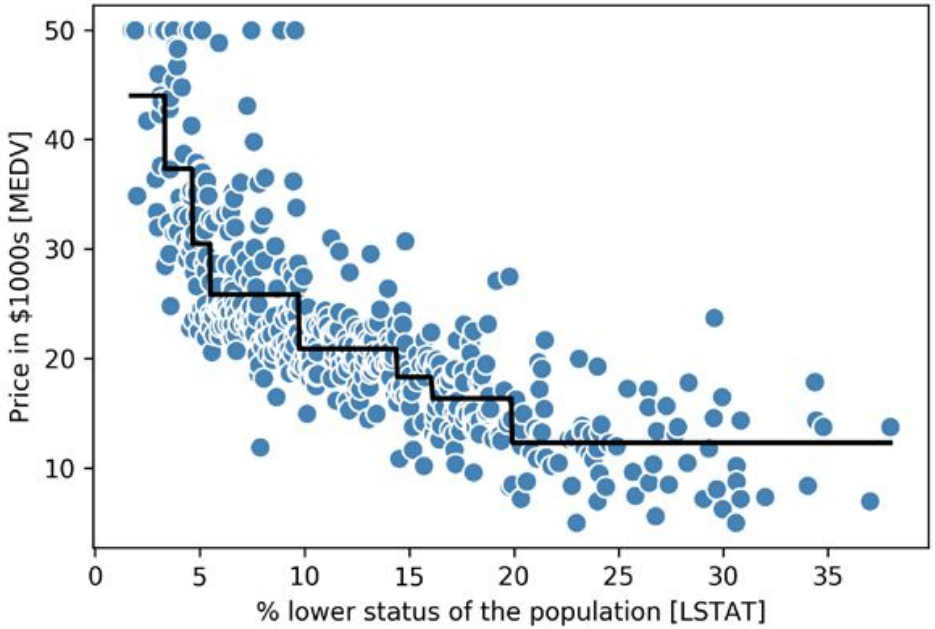
Decision Trees for Regression

$$I(t) = \text{MSE}(t) = \frac{1}{m_t} \sum_{i \in D_t} (y^{(i)} - \bar{y}_t)^2$$

- D_t is the training subset at node t.
- m_t is the size of D_t (i.e., the number of training examples at node t).
- $y^{(i)}$ is the true value of the target feature for the i-th training example in D_t.
- $\bar{y}_t$ is the average of the predicted values for examples in D_t.

Python Machine Learning, 2nd ed, pp 340

Decision Trees for Regression



Python Machine Learning, 2nd ed, pp 341