



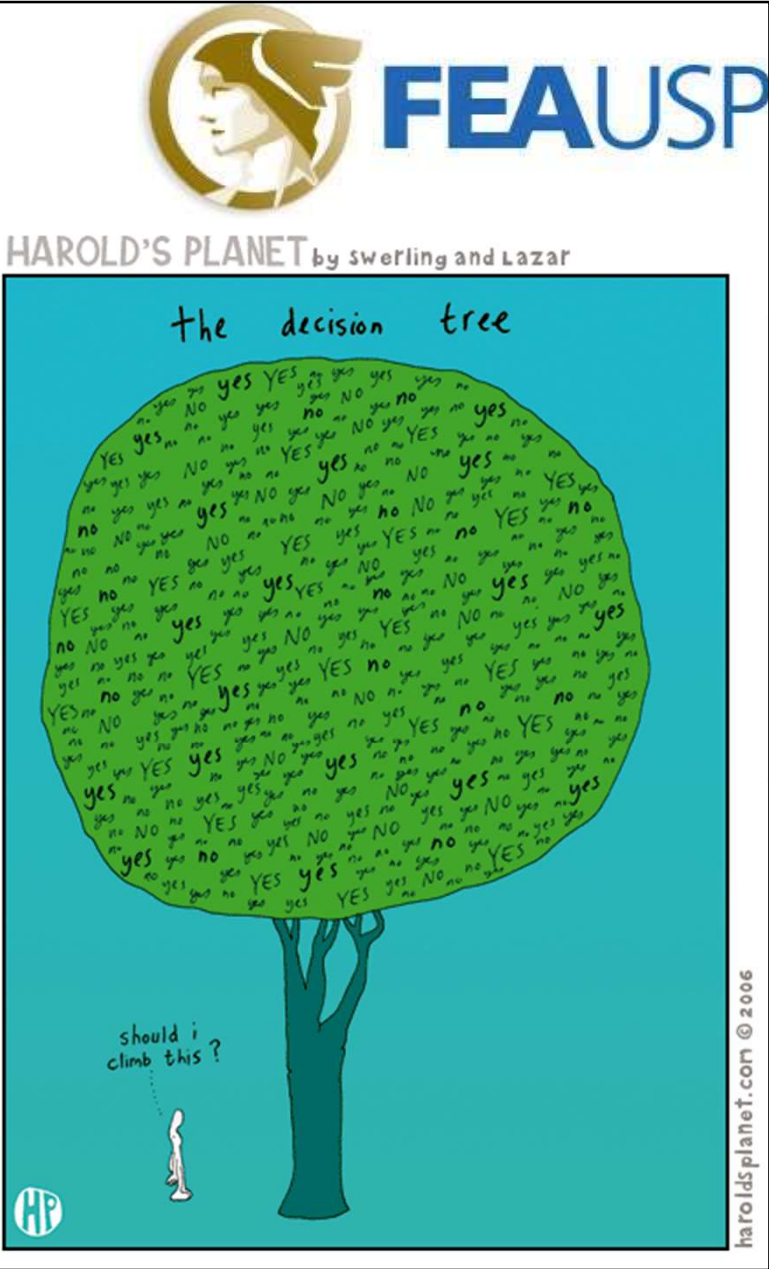
Machine Learning EAD0759

Prof. Antonio Geraldo **Vidal**

vidal@usp.br



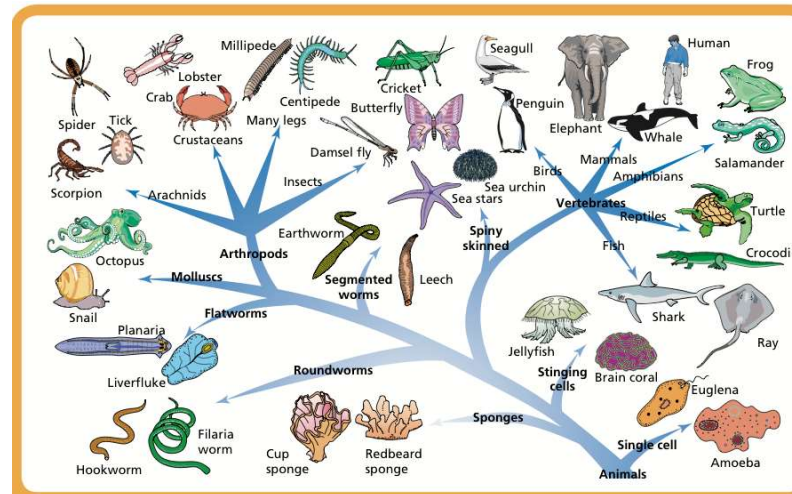
Árvores de Decisão





Conceitos Básicos

- Classificação, a tarefa de organizar objetos em uma entre diversas categorias pré-definidas, é um problema universal que engloba muitas aplicações diferentes.
- Classificação é a tarefa de definir uma **função alvo** que mapeie cada conjunto de atributos X para um dos rótulos de classes Y pré-determinados.
- A função alvo é denominada modelo de classificação.



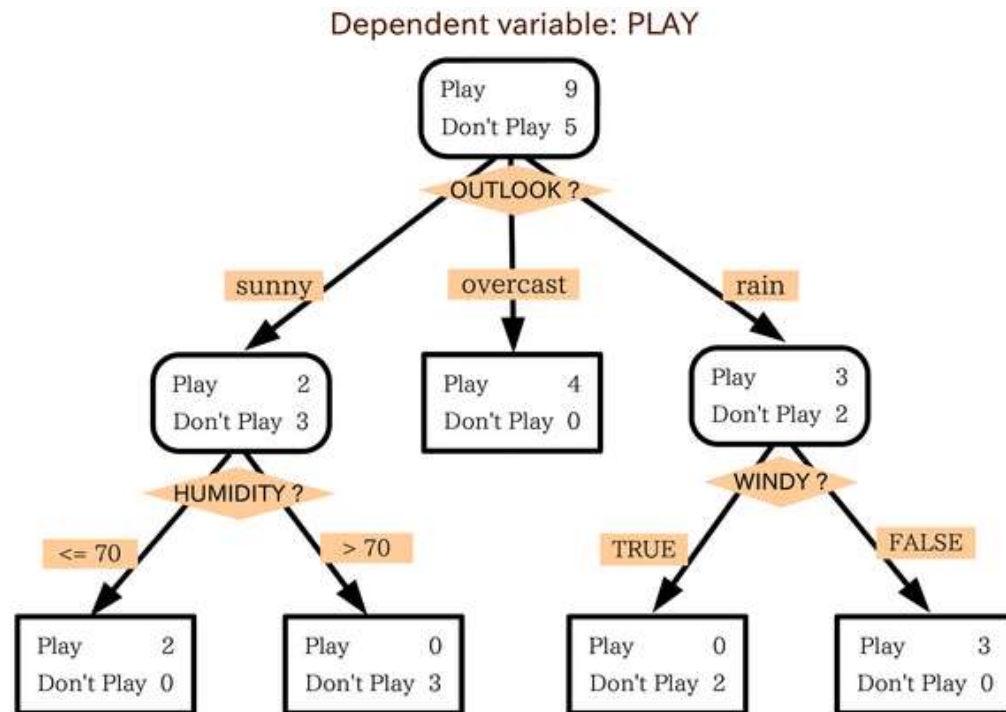


Modelo de Classificação

- Um modelo de classificação é útil para os seguintes propósitos:
 - **Modelagem Descritiva:** enquadrar objetos diferentes em classes conhecidas;
 - **Modelagem Preditiva:** prever classes não conhecidas de objetos diferentes.
- Técnicas de classificação são mais apropriadas para prever ou descrever conjuntos de dados com categorias nominais ou binárias.
- Cada técnica emprega um **algoritmo de aprendizagem** para identificar um modelo apropriado para o relacionamento entre o conjunto de atributos do objeto e o rótulo da classe.



Árvore de Decisão ou Regressão



- **Nó Raiz:** não possui arestas chegando e possui zero ou mais saindo.
- **Nó Galho:** possui uma aresta chegando e duas ou mais saindo.
- **Nó Folha:** possui uma aresta chegando e nenhuma saindo.
- Nós raiz e galho possuem condições de testes de atributos para separar objetos de características diferentes.
- Cada nó folha recebe um rótulo de classificação.



Árvore de Decisão

- Há exponencialmente inúmeras árvores de decisão que podem ser construídas a partir de um determinado conjunto de atributos.
- Encontrar a árvore ótima é inviável, porém algoritmos eficientes têm sido desenvolvidos para induzir a uma árvore de decisão razoavelmente precisa, embora não perfeita.
- Um destes é o algoritmo de Hunt, que é base para muitos outros incluindo CART – *Classification And Regression Tree*.



Árvore de Decisão - Hunt

- Suponha que D_t seja o conjunto de registros de treino que estão associados ao nó t e $Y = \{Y_1, Y_2, \dots, Y_c\}$ sejam rótulos das classes:
- **Passo 1**
 - Se todos os registros em D_t pertencem à mesma classe Y_t , então t é um nó folha rotulado como Y_t .
- **Passo 2**
 - Se D_t contiver registros que pertençam a mais de uma classe, uma **condição de teste de atributo** é selecionada para particionar os registros em subconjuntos menores. Um nó filho é criado para cada resultado da condição de teste e os registros de D_t são distribuídos para os filhos baseados nos resultados.
- **Passo 3**
 - Reaplicar recursivamente os passos anteriores para cada nó filho.

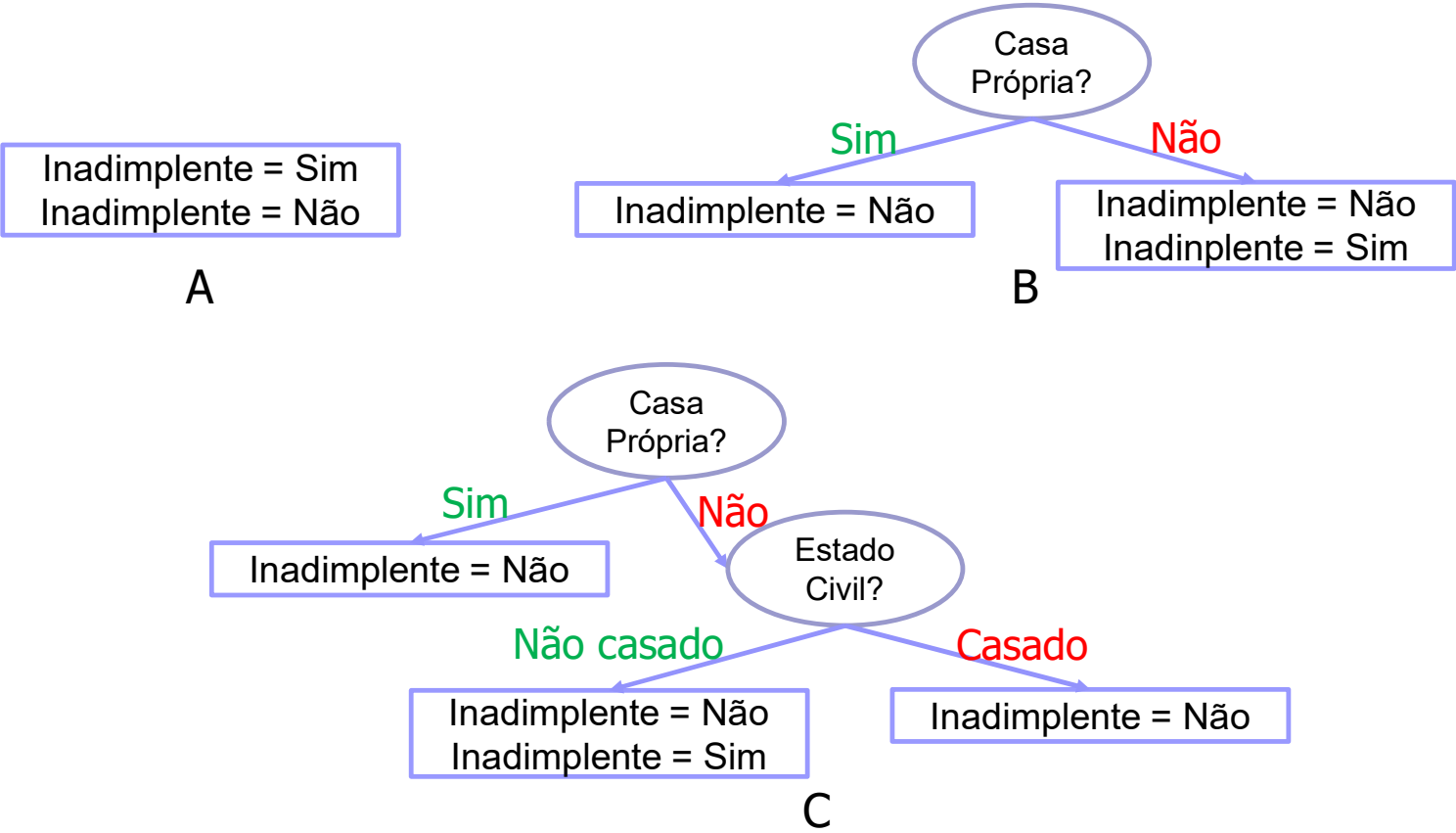



Árvore de Decisão - Exemplo

ID	Casa Própria	Estado Civil	Renda Anual	Inadimplente
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

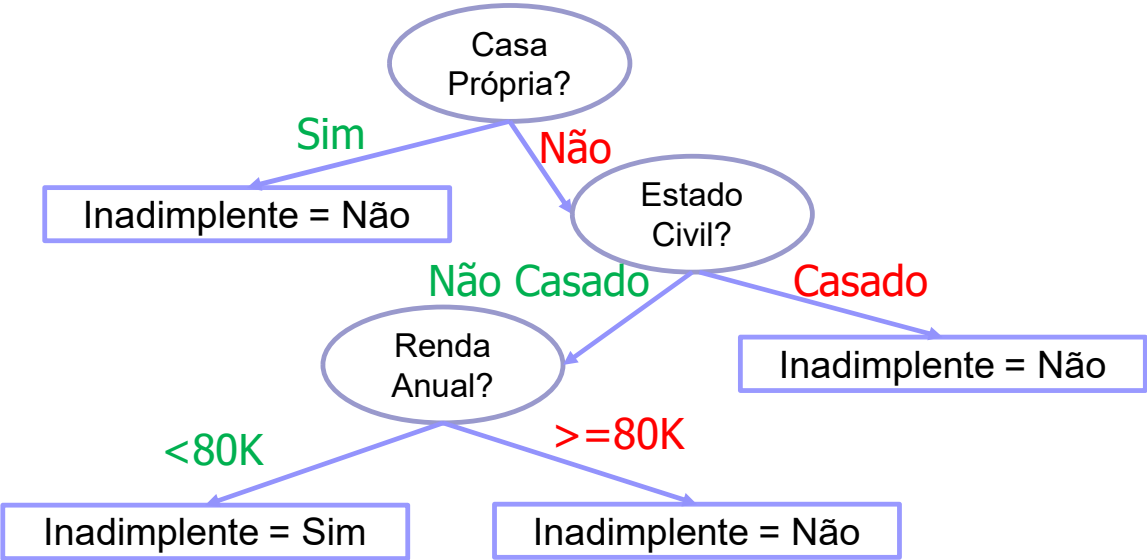


Árvore de Decisão - Exemplo





Árvore de Decisão - Exemplo



```
graph TD; A([Casa Própria?]) -- Sim --> B[Inadimplente = Não]; A -- Não --> C([Estado Civil?]); C -- Casado --> D[Inadimplente = Não]; C -- Não Casado --> E([Renda Anual?]); E -- <80K --> F[Inadimplente = Sim]; E -- >=80K --> G[Inadimplente = Não];
```

D

Questões

- Como considerar atributos de decisão binários, ordinais, nominais e contínuos?
- Como selecionar condições de teste de atributo para dividir registros?
- Quando o procedimento de divisão da árvore deve parar?
- Como obter a melhor árvore para um dado problema?



Condição de Teste de Atributos

- Atributos Binários
 - Geram dois resultados possíveis.
- Atributos Nominais
 - Geram múltiplos resultados possíveis ou $2^k - 1$ formas de partição binária de K valores de atributos.
- Atributos Ordinais
 - Geram múltiplos resultados possíveis ou múltiplas partições binárias.
- Atributos Contínuos
 - A condição de teste pode ser expressa como um teste de comparação ($A < v$ ou $A \geq v$) com resultados binários, ou uma faixa de múltiplos resultados $v_i \leq A \leq v_{i+1}$ para i de 1 a k. Para divisão múltipla, o algoritmo deve considerar todas as faixas possíveis de valores contínuos.



Métricas para Selecionar a Melhor Divisão ou Atributo

- Existem muitas métricas para determinar a melhor forma de dividir os registros.
- São definidas em termos da distribuição da classe dos registros antes e depois da divisão.
- São muitas vezes baseadas no grau de impureza dos nós filhos, buscando-se o menor grau de impureza.
- Exemplos de métricas de impureza:
 - Entropia (t) = $-\sum_{i=0}^{c-1} p(i|t) \log_2 p(i|t)$
 - Índice Gini (t) = $1 - \sum_{i=0}^{c-1} [p(i|t)]^2$
 - Erro de Classificação (t) = $1 - \max_i [p(i|t)]$



Algoritmo de Indução de Árvore de Decisão

Condição de Parada

```
1: Se cond_parada(E,F) = verdadeiro Então  
2:   folha = criarNo();  
3:   folha.rótulo = Classificar(E)  
4:   retorna folha  
5: Senão  
6:   raiz = criarNo()  
7:   raiz.cond_teste = encontrar_melhor_divisão(E,F)  
8:   atribuir  $V = \{v \mid v \text{ é um resultado possível de } raiz.cond\_teste\}$   
9:   Para cada  $v \in V$  faça  
10:     $E_v = \{e \mid raiz.cond\_teste(e) = v \text{ e } e \in E\}$   
11:    filho = CrescimentoDaÁrvore( $E_v$ ,F)  
12:    adicionar filho como descendente de raiz e rotular o limite  
    (raiz -> filho) como  $v$ .  
13:   Fim do faça  
14: Fim do Se  
15: retornar raiz
```



Características de Árvores de Decisão

1. Não requer quaisquer suposições quanto ao tipo de distribuição de probabilidades satisfeitas pela classe ou atributos;
2. Como encontrar a árvore ótima é um problema complexo, os algoritmos empregam uma abordagem heurística (aprendizagem de máquina) para guiar sua pesquisa no vasto espaço de hipóteses possíveis.
3. Técnicas para construção de árvores de decisão são computacionalmente pouco custosas, podendo gerar árvores grandes.
4. Árvores de decisão, especialmente as pequenas, são fáceis de ser interpretadas e ter a precisão avaliada.
5. O tamanho da árvore pode ser administrado com recursos de “poda”, permitindo a geração de árvores bem dimensionadas.
6. Os algoritmos utilizados são robustos quanto à presença de ruído.
7. Atributos redundantes não diminuem a precisão da árvore.
8. Para evitar folhas ou classes pouco significativas (fragmentação de dados) pode-se limitar a divisão abaixo de um certo limite de dados.
9. Pode ocorrer o problema de replicação de sub árvores.
10. As condições de teste envolvem apenas um atributo por vez, o que pode limitar a decisão para modelos complexos de atributos contínuos.



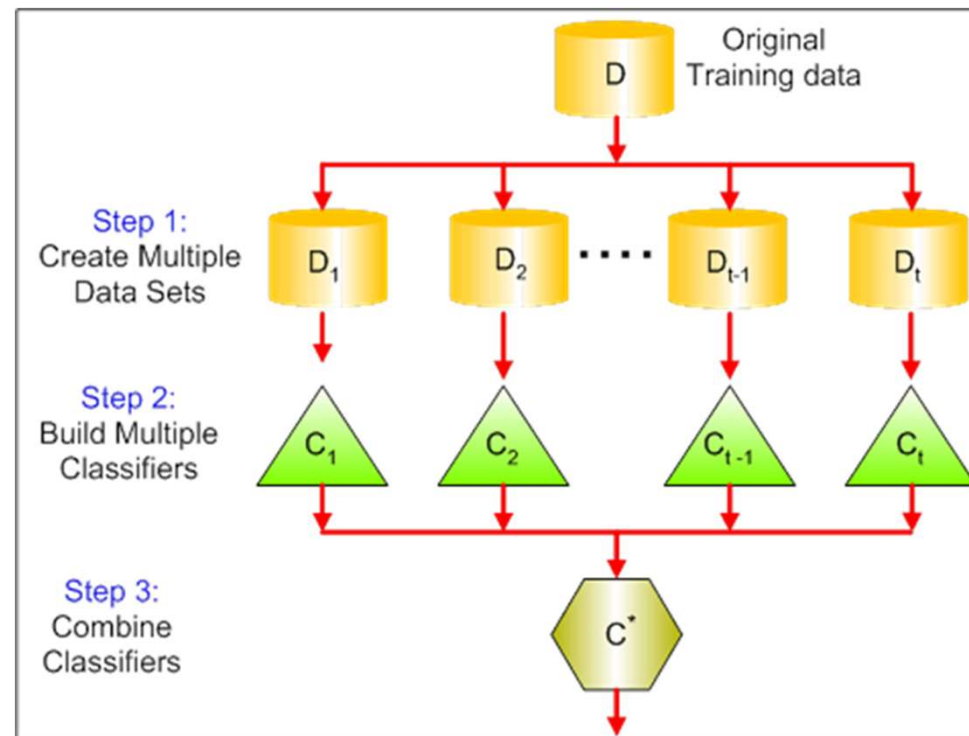
Avaliando o Desempenho de um Classificador

- A avaliação do erro auxilia o algoritmo de aprendizagem a executar a **seleção do modelo**, isto é, um modelo com a complexidade certa não sujeito a *overfitting* (superajuste).
- Pode-se medir o desempenho do modelo no conjunto de testes.
- A precisão calculada a partir do conjunto de teste também pode ser usada para comparar o desempenho relativo de diferentes classificadores no mesmo domínio.
- Para isso os rótulos das classes dos registros de teste devem ser conhecidos.
- Há vários métodos: *Holdout*, subamostra aleatória, validação cruzada, *bootstrap*, etc.



Bagging (ensacamento)

- O *bagging* ou *agregação de bootstrap*, é uma técnica usada para reduzir a variância de suas previsões, combinando o resultado de vários classificadores (árvores) modelados com diferentes sub amostras do mesmo conjunto de dados. A figura a seguir é apresentada as etapas para o *bagging*:





Bagging

- O bagging pode melhorar as previsões para muitos métodos de regressão e classificação, mas é particularmente útil para árvores de decisão.
- Para aplicar bagging ou ensacamento a árvores de regressão/classificação, você simplesmente constrói **B** árvores de regressão/classificação usando **B** conjuntos de treinamento, e calcula a média das previsões resultantes.
- Estas árvores são cultivadas profundamente, e não são podadas. Desta forma, cada árvore individual tem uma variação elevada, mas baixo viés. A média destas **B** árvores reduzem a variância.
- Em termos gerais, o procedimento de bagging tem demonstrado obter melhorias impressionantes na exatidão do resultado, combinando centenas ou mesmo milhares de árvores em um único procedimento.



Random Forests (florestas aleatórias)

- Random forests fornece uma melhoria sobre árvores com bagging por um pequeno ajuste *sobre* as árvores.
- Como no bagging, você constrói um número de árvores de decisão com amostras extraídas (*bootstrapped*) do treinamento.
- Mas ao construir essas árvores de decisão, cada vez que uma divisão em uma árvore é considerada, uma *amostra aleatória de preditores m* é escolhida como candidatos divididos do conjunto completo de *p* preditores.
- A divisão deve usar apenas um desses *m* preditores. Esta é a principal diferença entre random forests e bagging; porque como no bagging, a escolha do preditor *$m=p$* .



Boosting

- **Boosting** é outra abordagem para melhorar as previsões resultantes de uma árvore de decisão. Como o bagging e as random forests, é uma aproximação geral pode ser aplicada a muitos métodos de aprendizagem estatísticos para regressão ou classificação. Lembre-se de que o bagging envolve a criação de várias cópias do conjunto de dados de treinamento original usando um bootstrap, ajustando uma árvore de decisão separada para cada cópia e, em seguida, combinando todas as árvores para criar um único modelo preditivo. Notavelmente, cada árvore é construída em um conjunto de dados bootstrapped, independente das outras árvores.
- Boosting opera de forma semelhante, exceto que as árvores são cultivadas *sequencialmente*: cada árvore é cultivada usando informações de árvores cultivadas anteriormente. Boosting não envolve amostragem de bootstrap; em vez disso, cada árvore é ajustada em uma versão modificada do conjunto de dados original.



Random Forests

- Para cultivar uma Random Forest, você deve:
 1. Primeiro suponha que o número de casos no conjunto de treinamento é K . Em seguida, pegue uma amostra aleatória desses K casos e use esta amostra como o conjunto de treinamento para o cultivo da árvore.
 2. Se lá são p variáveis de entrada, especifique um número $m < p$ que em cada nó, você pode selecionar m variáveis aleatórias fora do p . A melhor divisão destes m é usada para dividir o nó.
 3. Cada árvore é subsequentemente gerada à extensão máxima possível e nenhuma poda é necessária.
 4. Por fim, agregue as previsões das árvores de destino para prever novos dados.