
Tera

Módulo 4:
Aprendizado de Máquina Supervisionado

Aula 19: Árvores de Decisão e Ensembles

I wan’t to play a game…

Regras do Jogo

- Vamos dividir a turma em grupos
- Cada grupo recebe 13 cartas de um baralho comum
- Cada grupo deve construir uma árvore de decisão para classificar suas cartas
- Cada nó na árvore é simplesmente uma comparação com um valor entre Ás e Rei

Regras do Jogo

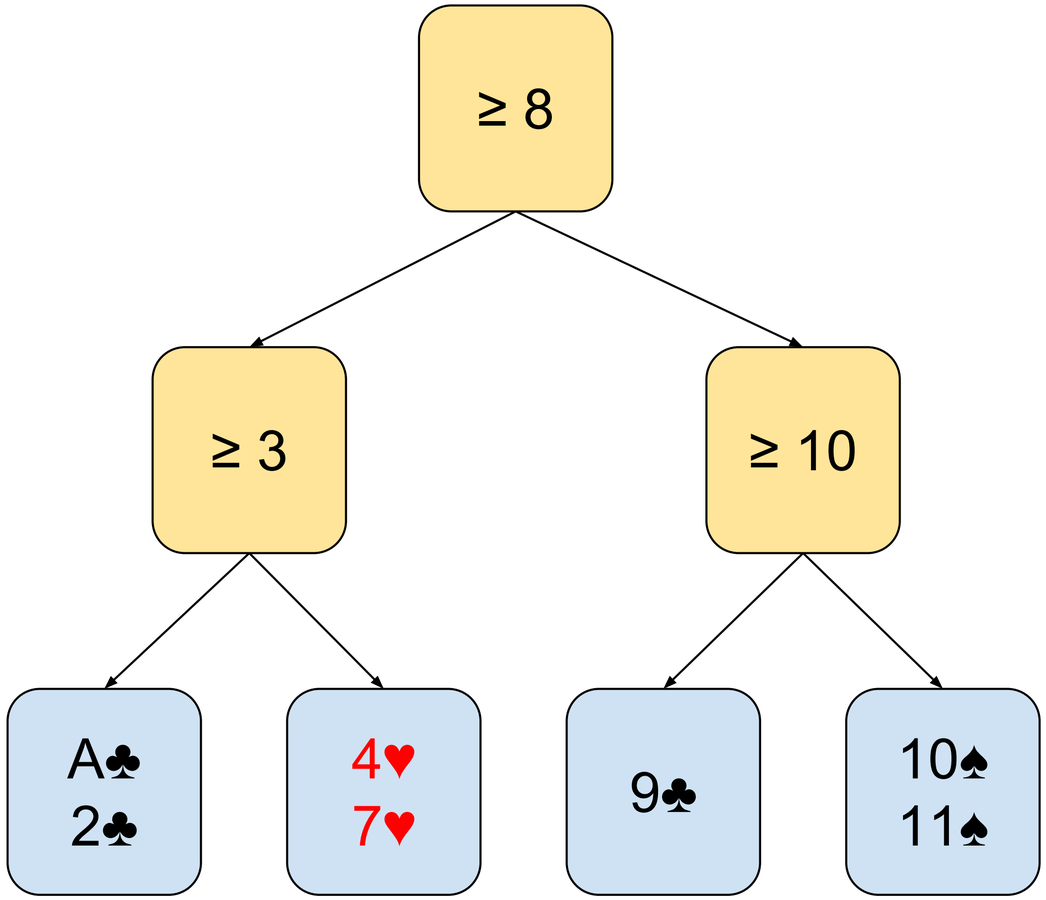
- Se a comparação é verdade para uma certa carta, ela vai para a direita do nó, caso contrário, ela vai para esquerda
- O objetivo é construir uma árvore tal que, ao passar todas cartas por ela, **cartas de um mesmo naipe devem estar separadas**
- Use a menor quantidade de nós possível!

Exemplo

- Recebi as cartas:

  2♣, 7♥, 11♠, 10♠, A♣, 4♥, 9♣

- Nossa árvore poderia ser:



Let’s do it!

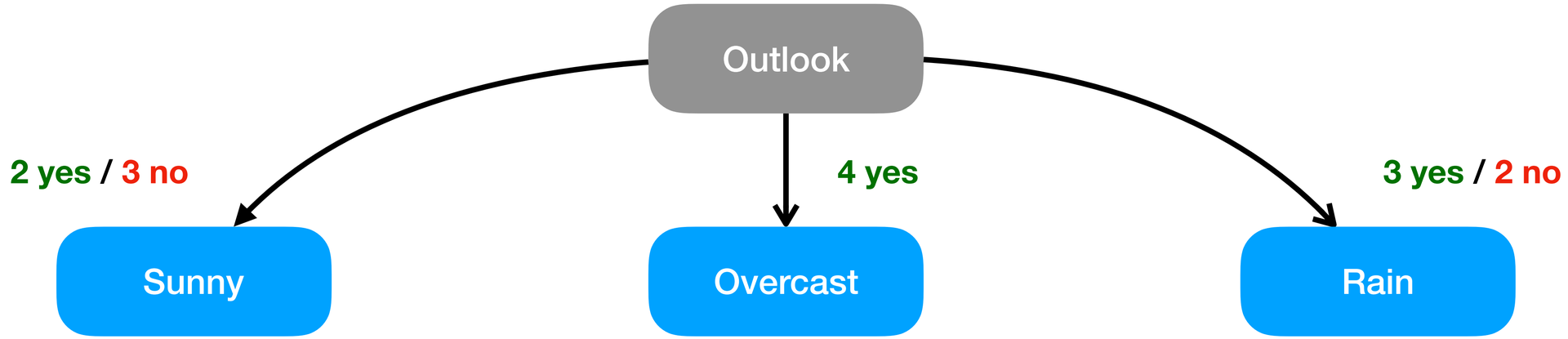
Árvores de Decisão

- Usadas amplamente em aplicações reais e competições
- Resolvem problemas não lineares excepcionalmente bem
- Extremamente flexíveis e interpretáveis
- Base para métodos muito poderosos (Gradient Boosting, Random Forests)

Exemplo

Day	Outlook	Humidity	Wind	Playing?
D1	sunny	high	strong	no
D2	sunny	high	weak	no
D3	overcast	high	weak	yes
D4	rain	high	weak	yes
D5	rain	normal	weak	yes
D6	rain	normal	strong	no
D7	overcast	normal	strong	yes
D8	sunny	high	weak	no
D9	sunny	normal	weak	yes
D10	rain	normal	weak	yes
D11	sunny	normal	strong	yes
D12	overcast	high	strong	yes
D13	overcast	normal	weak	yes
D14	rain	high	strong	no

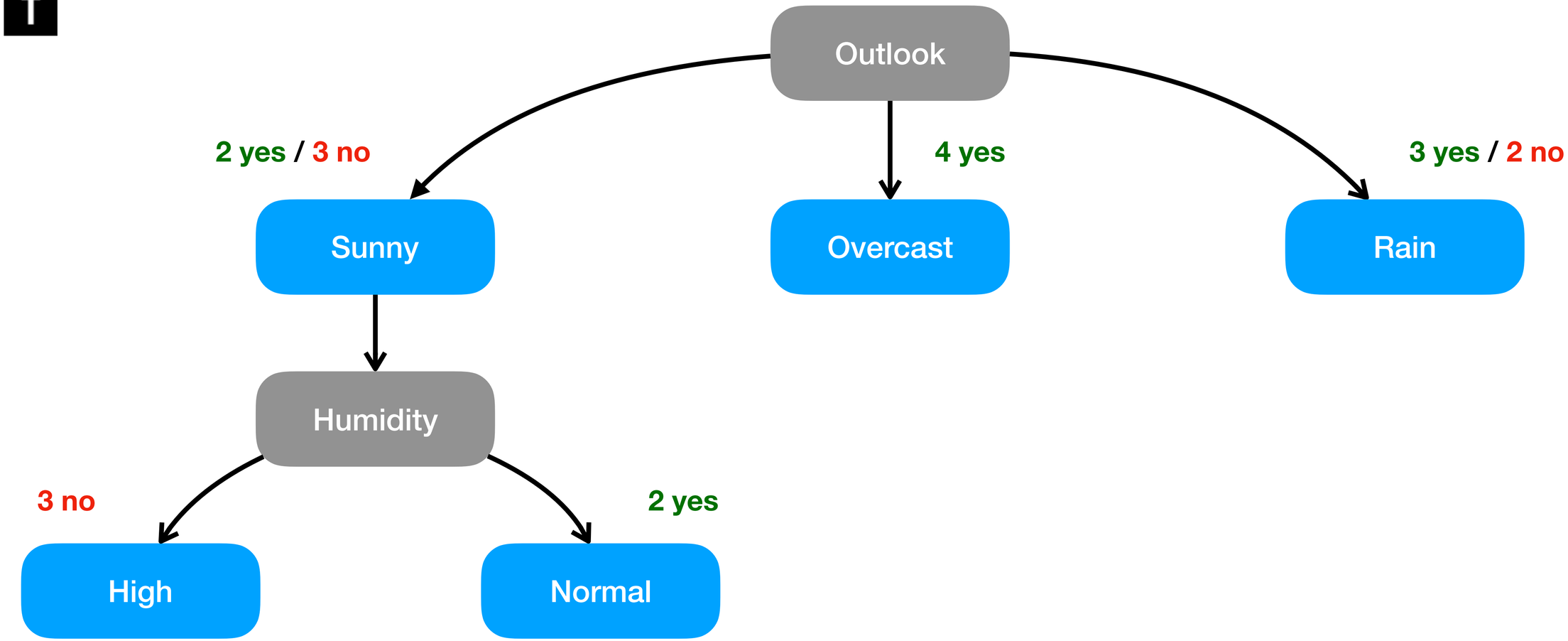
- Queremos prever (e entender) quando alguém vai estar jogando na quadra.
- Temos informações de 14 dias:
  - Sobre o tempo
  - Alguém estava jogando (target)
- Que tal começamos pelo aspecto do tempo (outlook)?



Outlook	Humidity	Wind
sunny	high	strong
sunny	high	weak
sunny	high	weak
sunny	normal	weak
sunny	normal	strong

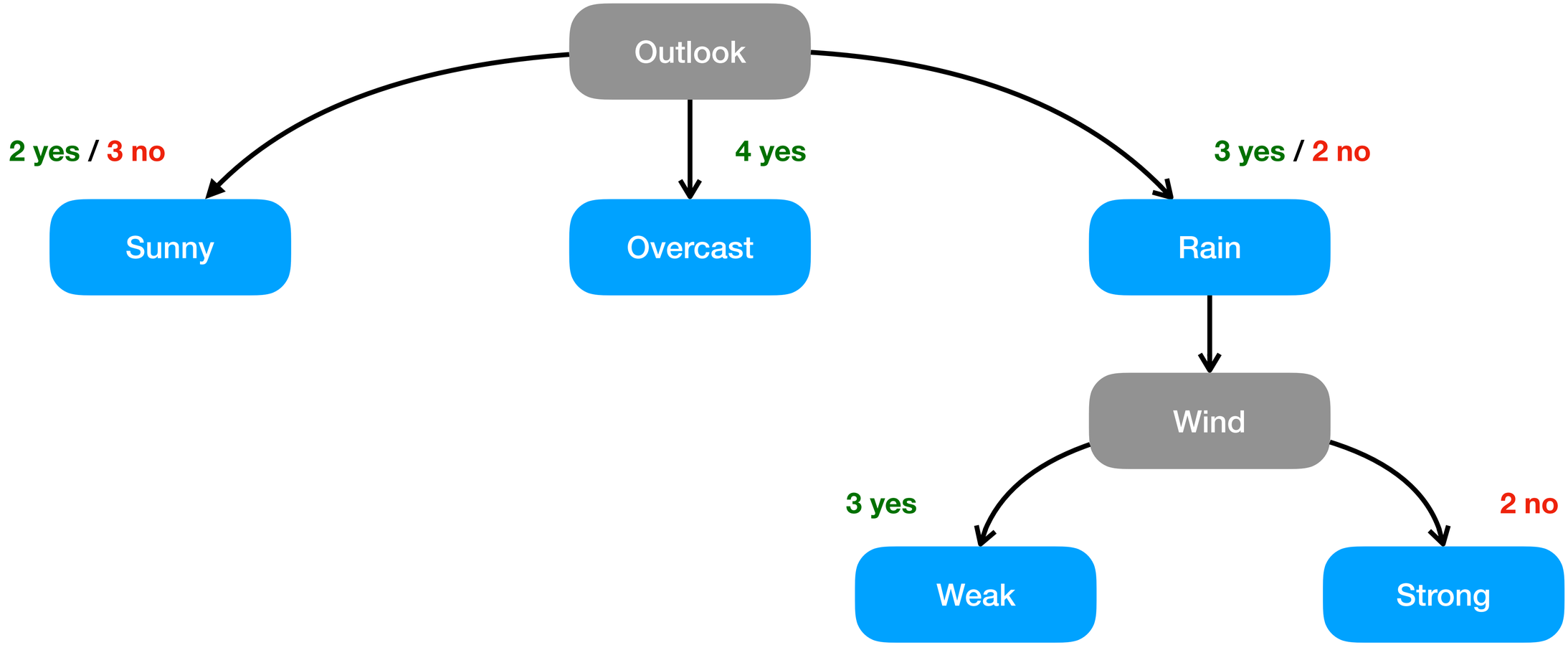
Outlook	Humidity	Wind
overcast	high	weak
overcast	normal	strong
overcast	high	strong
overcast	normal	weak

Outlook	Humidity	Wind
rain	high	weak
rain	normal	weak
rain	normal	weak
rain	normal	strong
rain	high	strong



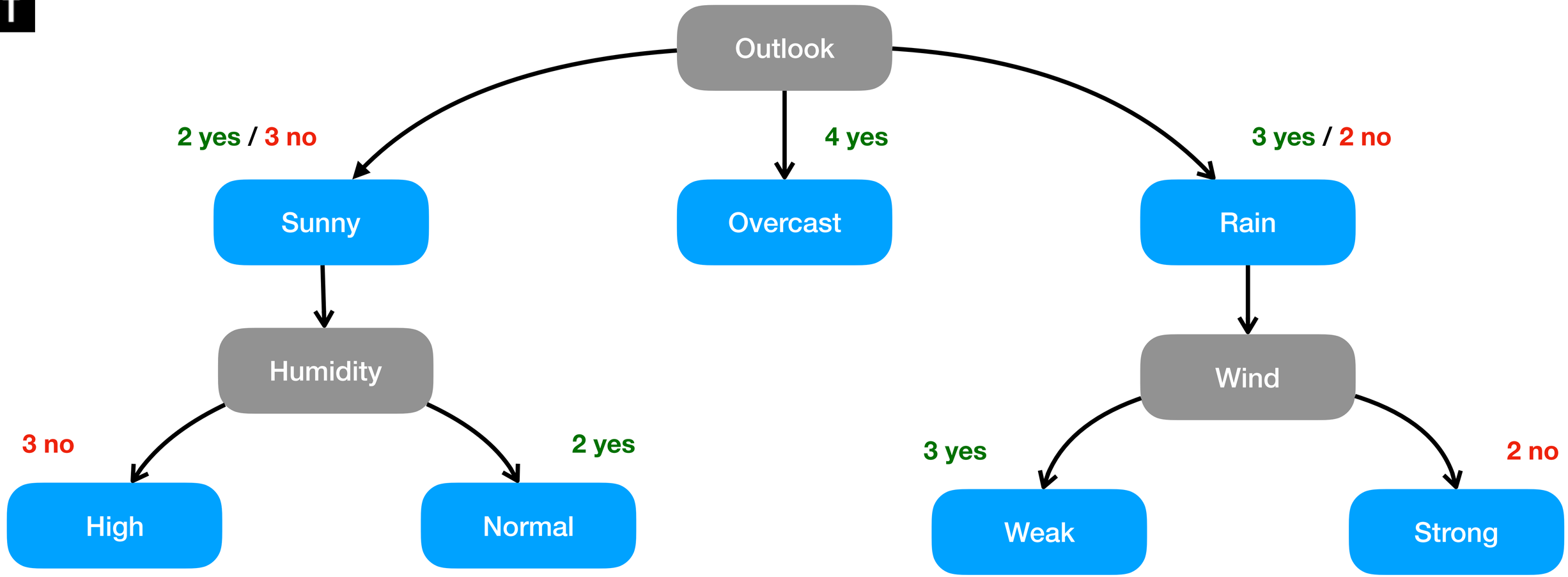
Outlook	Humidity	Wind
sunny	high	strong
sunny	high	weak
sunny	high	weak

Outlook	Humidity	Wind
sunny	normal	weak
sunny	normal	strong



Outlook	Humidity	Wind
rain	high	weak
rain	normal	weak
rain	normal	weak

Outlook	Humidity	Wind
rain	normal	strong
rain	high	strong

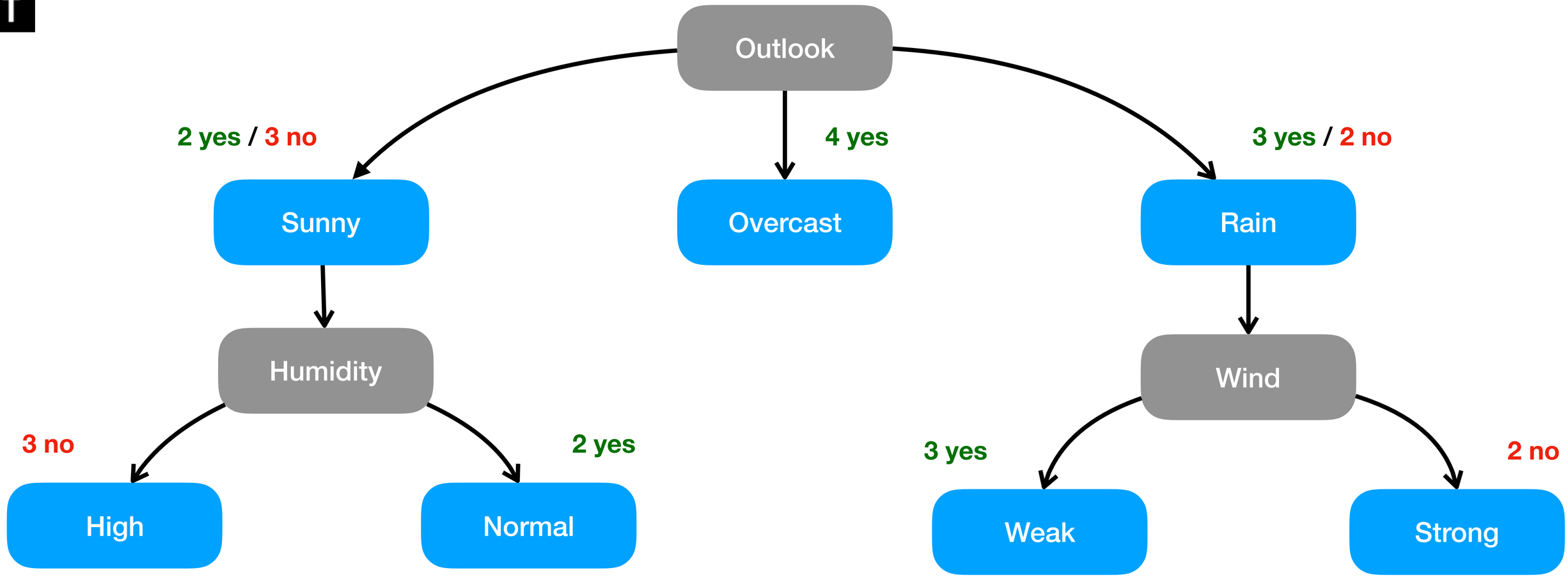


Outlook	Humidity	Wind
sunny	high	strong
sunny	high	weak
sunny	high	weak

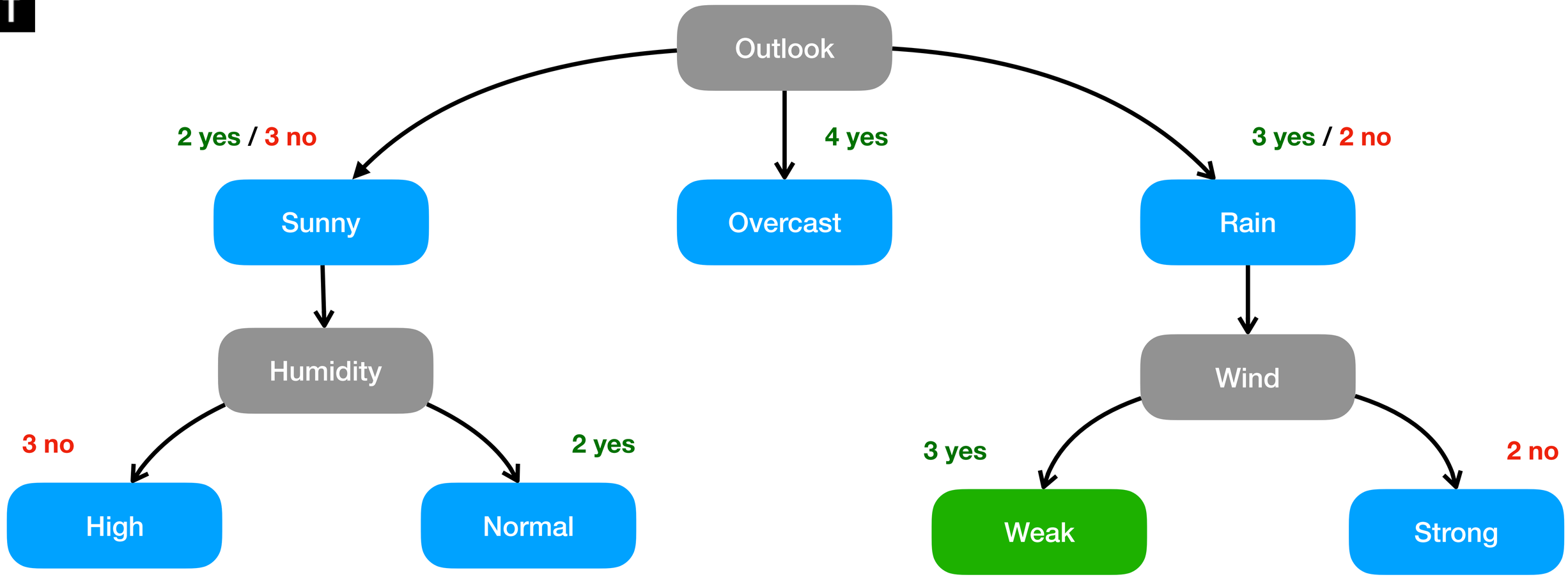
Outlook	Humidity	Wind
sunny	normal	weak
sunny	normal	strong

Outlook	Humidity	Wind
rain	high	weak
rain	normal	weak
rain	normal	weak

Outlook	Humidity	Wind
rain	normal	strong
rain	high	strong



Day	Outlook	Humidity	Wind	Playing?
D15	rain	high	weak	???



Day	Outlook	Humidity	Wind	Playing?
D15	rain	high	weak	yes

Árvore de Decisão

*Um modelo estatístico **supervisionado** que busca aprender uma **seqüência de regras** estruturadas em uma árvore de modo a maximizar a **separação** entre instâncias de diferentes classes*

Árvore de Decisão

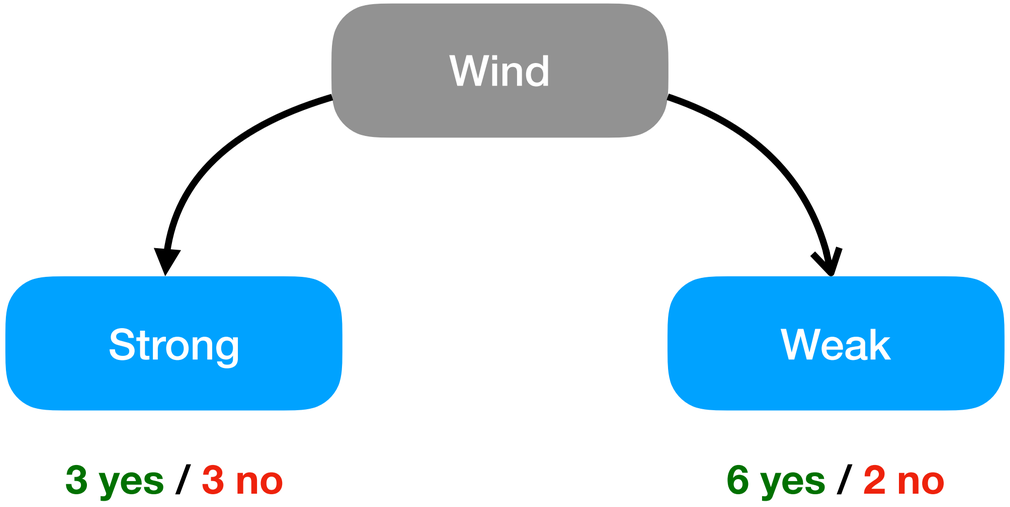
- Os nós finais são chamados de **folhas** e contém a decisão final em relação a classe (ou probabilidade das classes)
- Os nós com instâncias de uma só classe são chamados de **puro**
- Podem ser usadas para **classificação** e **regressão**
- Operam em atributos **numéricos** e **categóricos***

<code> … </code>

Escolhendo Atributos

- Escolhemos começar nossa árvore por Outlook
- E se tivéssemos começado por Wind?

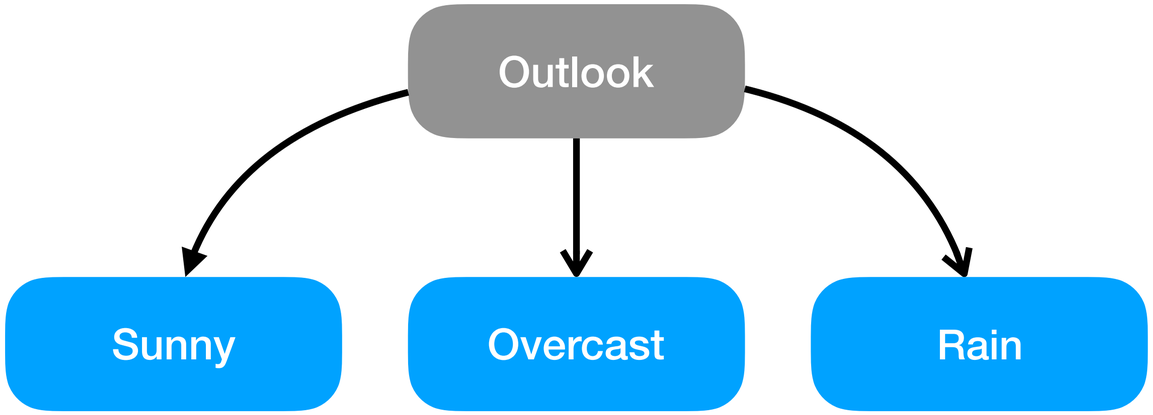
Outlook	Humidity	Wind
sunny	high	strong
rain	normal	strong
overcast	normal	strong
sunny	normal	strong
overcast	high	strong
rain	high	strong



Outlook	Humidity	Wind
sunny	high	weak
overcast	high	weak
rain	high	weak
rain	normal	weak
sunny	high	weak
sunny	normal	weak
rain	normal	weak
overcast	normal	weak

Escolhendo Atributos

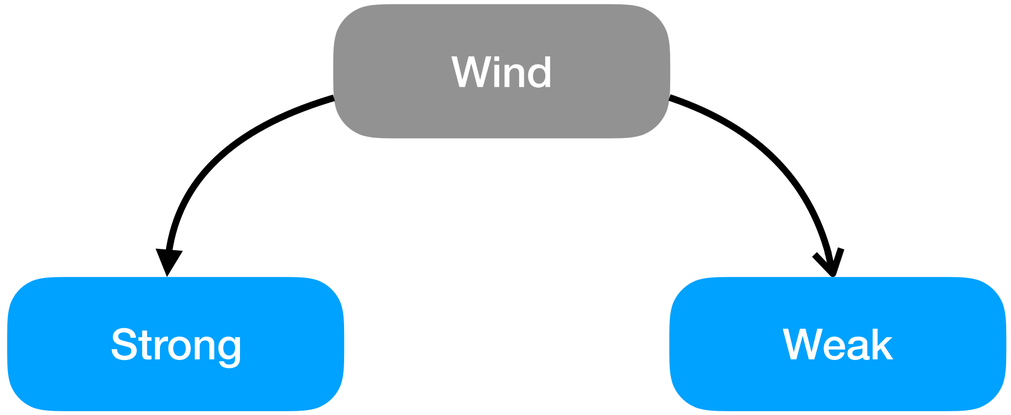
- Qual escolha de *split* foi melhor?
- Suponhamos que temos que chutar o resultado logo depois do primeiro split



2 yes / 3 no **4 yes** **3 yes / 2 no**

P(no|sunny) = 3/5 **P(yes|overcast) = 4/4** **P(yes|rain) = 3/5**

P(correct|humidity) = (0.6 + 1 + 0.6)/3 = 0,7333



3 yes / 3 no **6 yes / 2 no**

P(yes|sunny) = 3/6 **P(yes|weak) = 6/8**

P(correct|wind) = (0.5 + 0.75)/2 = 0,625

Escolhendo Atributos

- A probabilidade de acertamos é maior no primeiro split, pois ele discrimina melhor a nossa variável resposta (playing)
- O primeiro split representa um **ganho de informação** maior

Entropia

- Como escolher os melhores splits sistematicamente?
- Vamos usar a entropia dos subconjuntos como métrica de **impureza**:

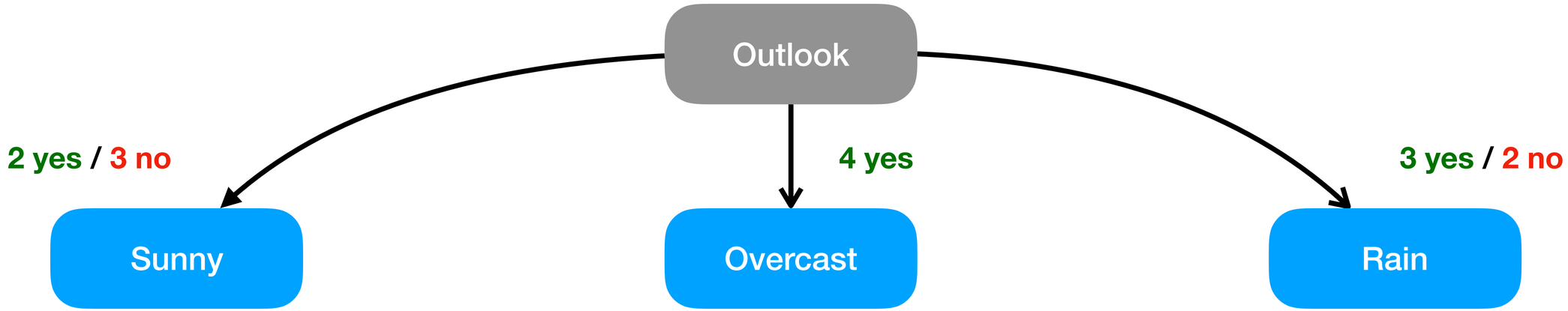
$$H(E) = -P_{yes} \log_2 P_{yes} - P_{no} \log_2 P_{no}$$

$$P_{yes} = \frac{n_{yes}}{n_{yes} + \mathrm{n}_{no}}$$

$$P_{no} = \frac{n_{no}}{n_{yes} + \mathrm{n}_{no}}$$

Exemplo

$$H(E) = -P_{yes} \log_2 P_{yes} - P_{no} \log_2 P_{no}$$



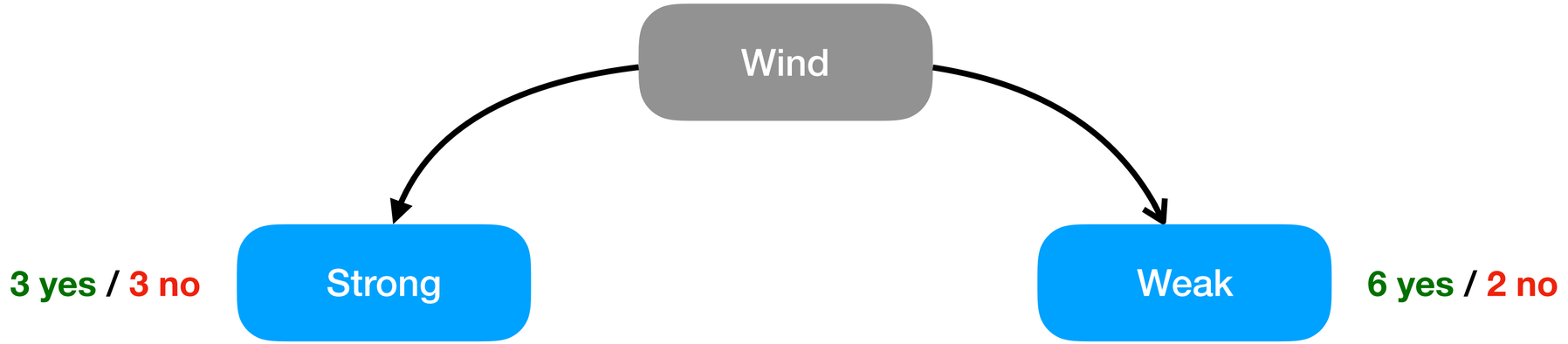
$$H(E_S) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}$$
$$= -0.4\log_2 0.4 - 0.6\log_2 0.6$$
$$= 0.9705$$

$$H(E_O) = -\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4}$$
$$= -1\log_2 1 - 0\log_2 0$$
$$= 0.0$$

$$H(E_R) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}$$
$$= -0.6\log_2 0.6 - 0.4\log_2 0.4$$
$$= 0.9705$$

Exemplo

$$H(E) = -P_{yes} \log_2 P_{yes} - P_{no} \log_2 P_{no}$$



$$H(E_S) = -\frac{3}{6}\log_2\frac{3}{6} - \frac{3}{6}\log_2\frac{3}{6}$$
$$= -0.5\log_2 0.5 - 0.5\log_2 0.5$$
$$= 1$$

$$H(E_W) = -\frac{6}{8}\log_2\frac{6}{8} - \frac{2}{8}\log_2\frac{2}{8}$$
$$= -0.75\log_2 0.75 - 0.25\log_2 0.25$$
$$= 0.8112$$

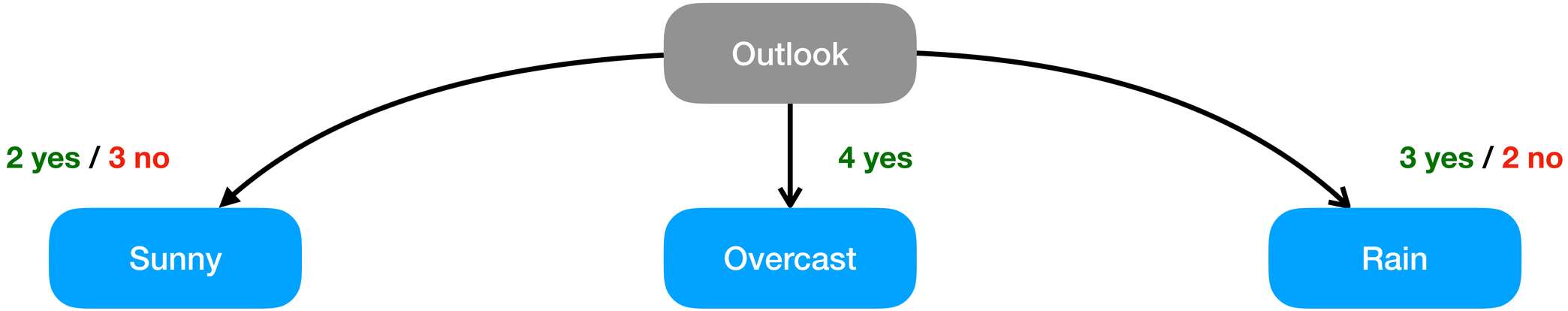
Entropia Esperada

- Ainda resta a pergunta, qual desses splits é objetivamente melhor?
- Considere um split **S** em um atributo com **K** valores diferentes, que divide o conjunto **E,** de tamanho **N**, em subconjuntos $\mathbf{E_1}$, $\mathbf{E_2}$, …, $\mathbf{E_K}$ de tamanhos $\mathbf{N_1}$, $\mathbf{N_2}$, …, $\mathbf{N_K}$, respectivamente
- Vamos definir a **entropia esperada** após esse split como:

$$EH(S) = \sum_{i=1}^{K} \frac{N_i}{N} H(E_i)$$

Exemplo

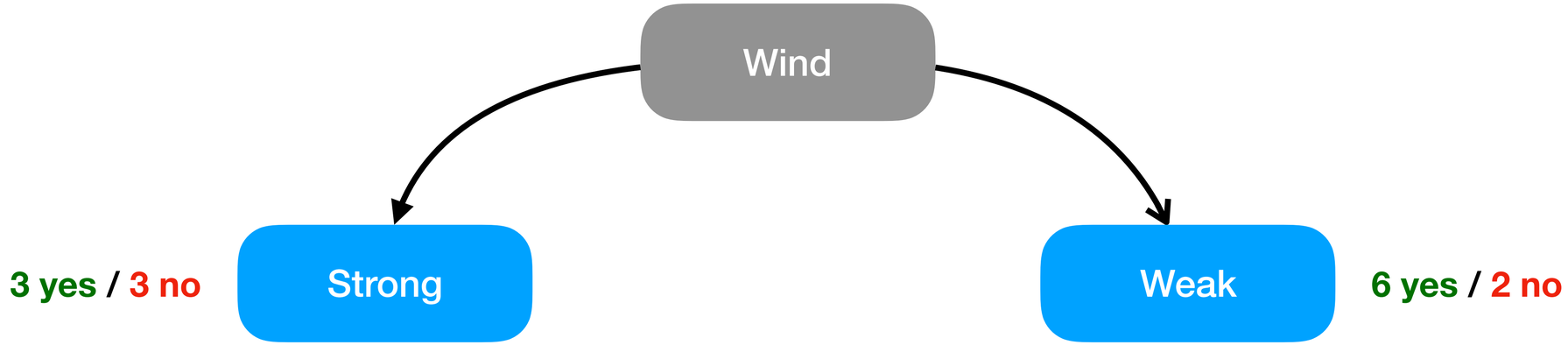
$$EH(S) = \sum_{i=1}^{K} \frac{N_i}{N} H(E_i)$$



$$EH(S_{Outlook}) = \frac{5}{14}0.9705 + \frac{4}{14}0 + \frac{5}{14}0.9705$$
$$= 0.6932$$

Exemplo

$$EH(S) = \sum_{i=1}^{K} \frac{N_i}{N} H(E_i)$$



$$EH(S_{Outlook}) = \frac{6}{14}1 + \frac{8}{14}0.8112$$
$$= 0.8921$$

Ganho de Informação

- Não basta minimizar a entropia esperada, pois o importante é que haja **redução de entropia** para haver ganho de informação
- Por fim, definimos o **ganho de informação** de um split **S** como:

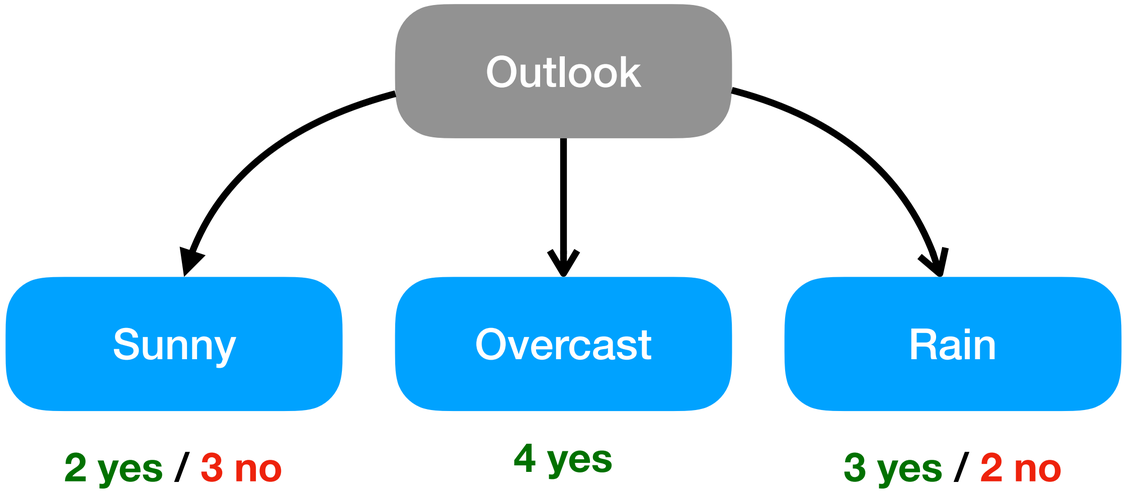
$$I(S) = H(E) - EH(S)$$

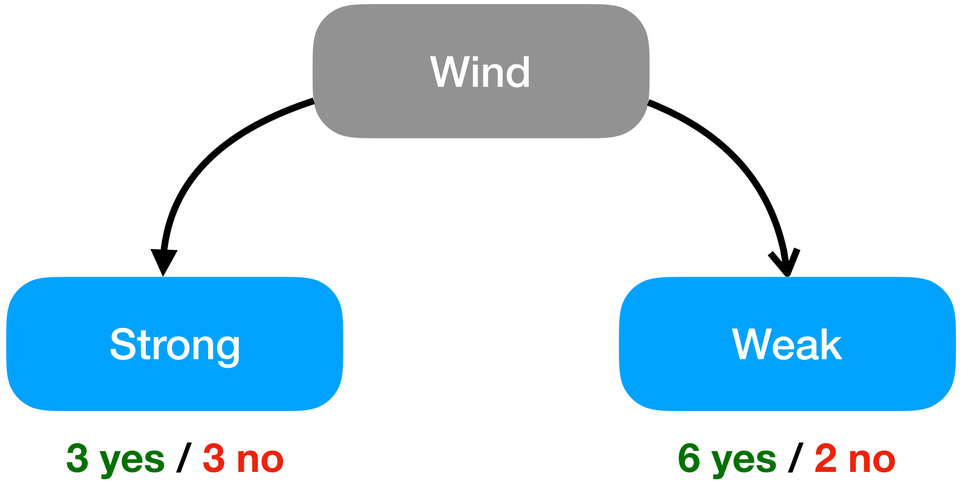
$$I(S) = H(E) - \sum_{i=1}^{K} \frac{N_i}{N} H(E_i)$$

Exemplo

$$I(S) = H(E) - EH(S)$$

$$\begin{aligned} H(E) &= -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} \\ &= -0.643\log_2 0.643 - 0.357\log_2 0.357 \\ &= 0.9402 \end{aligned}$$





$$\begin{aligned} I(S_{Outlook}) &= 0.9402 - 0.6932 \\ &= 0.2470 \end{aligned}$$

$$\begin{aligned} I(S_{Wind}) &= 0.9402 - 0.8921 \\ &= 0.0482 \end{aligned}$$

<code> … </code>

Algoritmo

TreinoDT(Xy)

- Se (len(Xy) < MIN_AMOSTRAS) e (Impureza(Xy) < MIN_IMPUREZA)
  - Retorne NoFolha(Xy)
- Para cada atributo Ai de Xy, calcule o ganho de informação I(Ai)
- Seja Am o atributo com o maior valor I(Am), crie um nó Nm para Am
- Divida Xy em [Xy1, Xy2, …, Xyk], de acordo com os valores de Am
- Faça [TreinoDT(Xy1), TreinoDT(Xy2), … TreinoDT(Xyk)] filhos de Nm

Algoritmos

- Existem muitos algoritmos diferentes ID3, CART, C4.5, etc
- Na prática, as implementações disponíveis geralmente são **híbridas** e funcionam muito bem
- Existem outros critérios de impureza, como **Gini**
- Tratamos o caso de classificação binária, mas métricas de impureza generalizam perfeitamente para **múltiplas classes**

Atributos contínuos

- Como podemos trabalhar com **atributos contínuos** também?
- Usando operadores de comparação: >, <, !=
- Não só buscamos os melhores atributos, mas os melhores splits em cada atributo
- Dado um atributo A, os splits possíveis podem ser: $A \leq V_j$, onde V_j são valores definidos pela sua implementação
- Geralmente isso é feito de forma a não perder informação

<code> … </code>

Vantagens

- Extremamente interpretáveis e intuitivas
- Pouco tratamento dos dados, pois (em teoria) não exige normalização, remoção de outliers, imputação e one-hot encoding
- Lida com atributos categóricos e contínuos
- Predição relativamente rápida em O(log(n))
- Possuem uma seleção de features natural

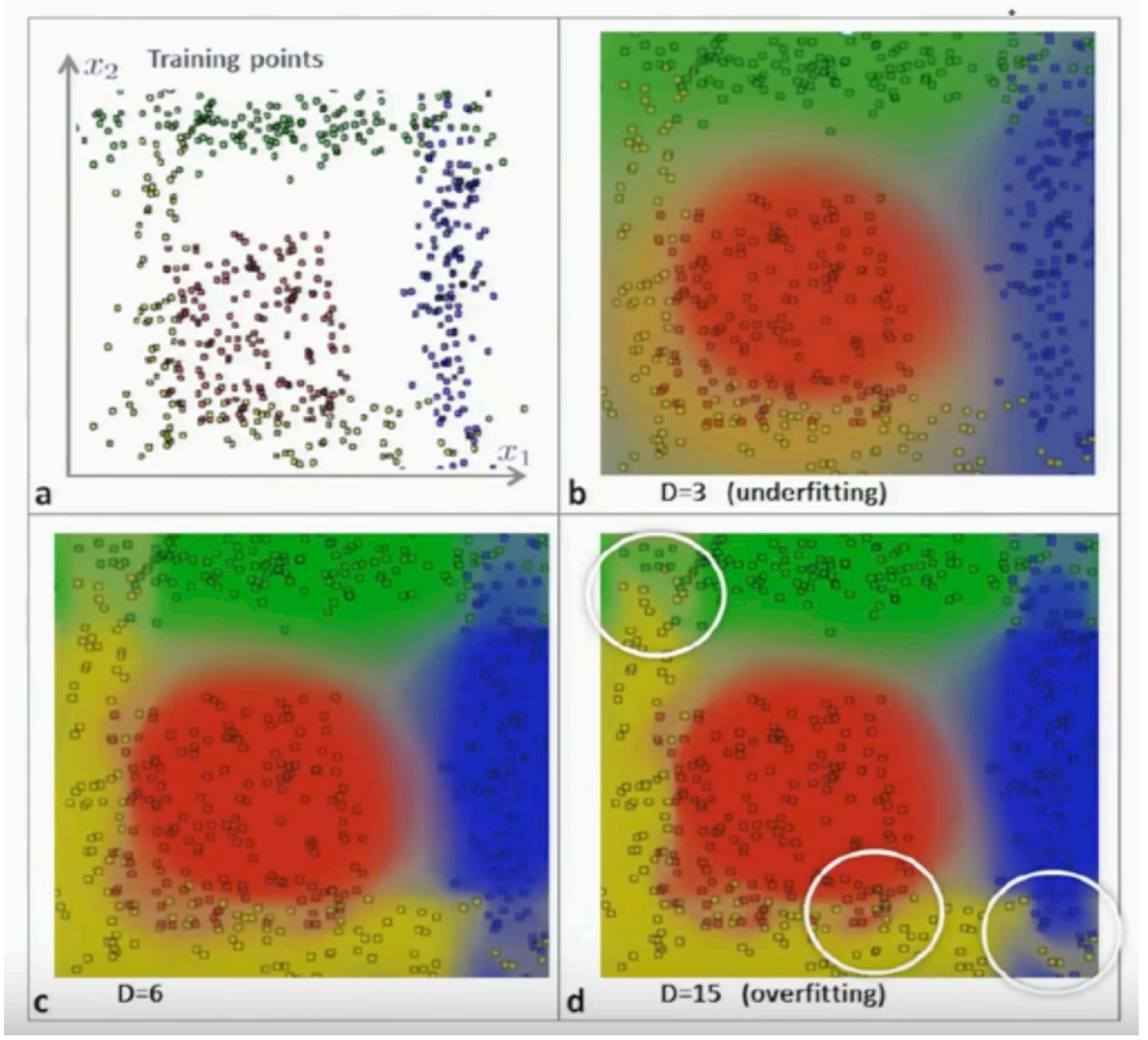
Desvantagens

- Instáveis: pequenas variações nas instâncias podem gerar árvores completamente diferentes
- Propensas a criar modelos extremamente complexos que não generalizam bem (overfitting)
- O problema de criar uma árvore ótima é NP-completo, logo os algoritmos gulosos usados frequentemente caem ótimos locais retornar árvores sub-ótimas

Overfitting

- Descrevem bem o conjunto de treino, porém não necessariamente generalizam
- Para controlar a complexidade dos modelos, os hiperparâmetros devem ser devidamente otimizados
- Como isso pode ser feito?
- Principais hiperparâmetros que controlam complexidade:
  - *max_depth*: profundidade máxima da árvore
  - *min_samples_split*: mínimo de instâncias no nó para que haja um split
  - *min_impurity_decrease*: mínimo de redução de impureza para que haja um split

<code> … </code>


Training points
x_2
x_1
a
b
D=3 (underfitting)
c
D=6
d
D=15 (overfitting)

<code> … </code>

Análise e Exploração

- Vimos como árvores de decisão podem ser transparente e informativas em relação ao que elas aprendem com os dados
- São muito poderosas para oferecer *insights* para seu negócio, mesmo quando um modelo preditivo não é necessário ou possível de se implementar

<code> … </code>

Limitações

- Apesar das inúmeras vantagens, árvores de decisão são modelos relativamente fracos em termos de poder preditivo
- A alta variância e a propensidade a overfitting são problemas significativos
- Felizmente esses problemas podem ser mitigados usando florestas aleatórias

Cenário

- Imagine que você possui capital e precisa decidir se deve **investir um startup promissora S**
- A Startup opera em uma área pouco familiar para você, então você **conversa com 5 especialistas** para auxiliar na sua decisão

Especialistas

Especialista em media social

Ex-CEO de uma startup adquirida

Operador da bolsa

Ex-funcionário da empresa

Investidor da empresa

Especialistas

60% de acerto

70% de acerto

75% de acerto

65% de acerto

70% de acerto

Especialistas

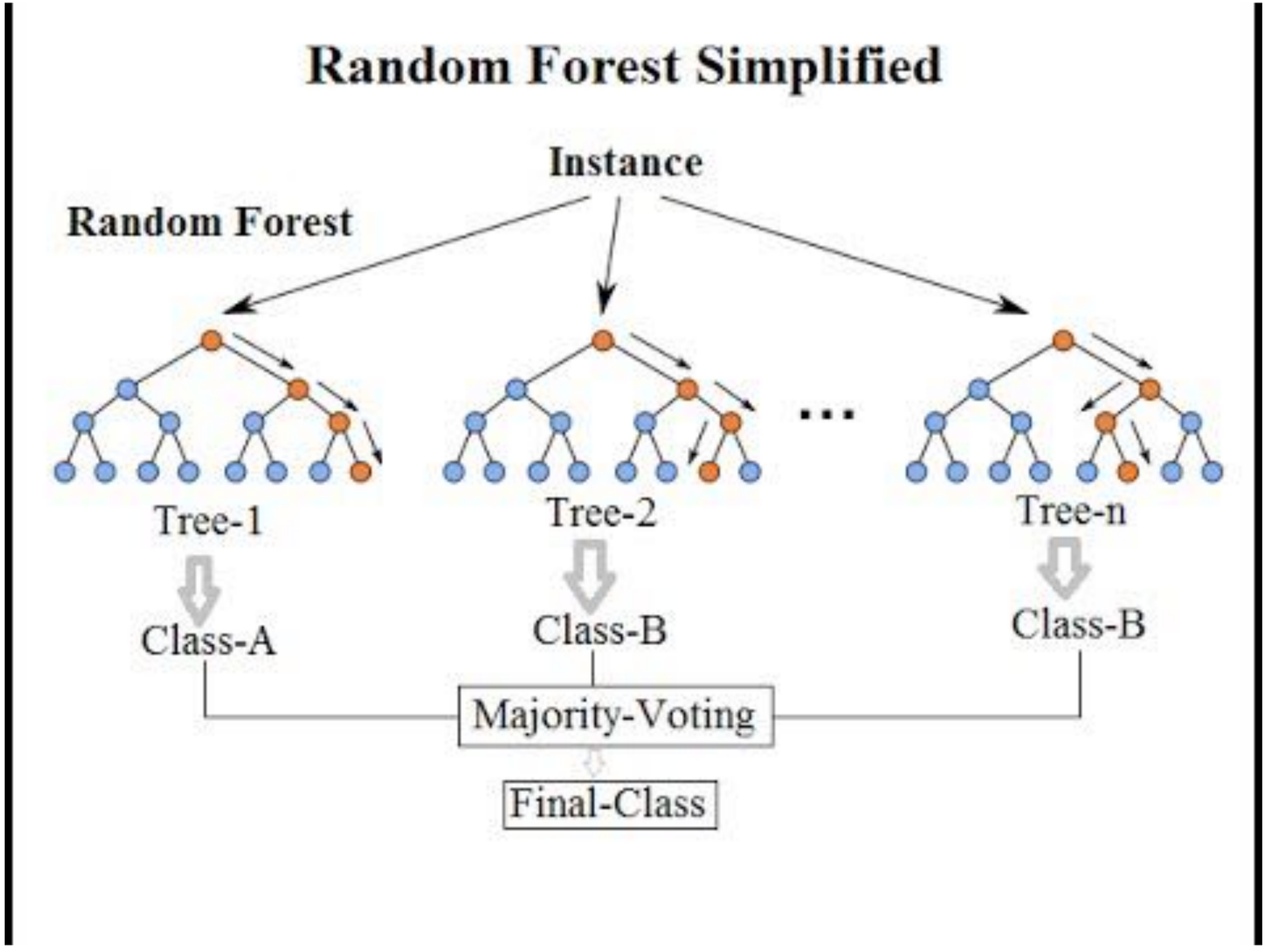
- Todos têm seus pontos fortes e seus viés
- Você se sentiria mais confortável com a opinião de um deles ou de todos eles?
- Se você somente decidir por unanimidade, sua chance de acerto é maior que 99.9% !!
- Se você decidir por maioria, sua chance de acerto ainda é de 90%!

Especialistas

- Esses números assumem que as opiniões dos especialistas **são independentes**, o que raramente é verdade
- Intuitivamente, se todos especialistas fossem ex-funcionários da empresa S, sua confiança seria mesma que no primeiro cenário?
- Diversidade é bom!
- Agora pense que ao invés de especialistas, temos cinco modelos treinados

Florestas Aleatórias

- Podemos usar as decisões de várias árvores para fazer uma predição
- A diversidade entre as árvores é promovida usando dados e atributos diferentes pra treinar cada árvore
- Por fim, devemos combinar as decisões das árvores de alguma forma


Random Forest Simplified
Instance
Random Forest
Tree-1
Tree-2
...
Tree-n
Class-A
Class-B
Class-B
Majority-Voting
Final-Class

<code> … </code>

Ensembles

- Mas se podemos combinar resultados de árvores de decisão diferentes, por que não **combinar quaisquer modelos diferentes**?

Ensembles *são métodos que utilizam múltiplos modelos para obter resultados preditivos que os modelos constituintes não seriam capaz de obter*

Ensembles

Ensembles *são métodos que utilizam* ***múltiplos modelos***

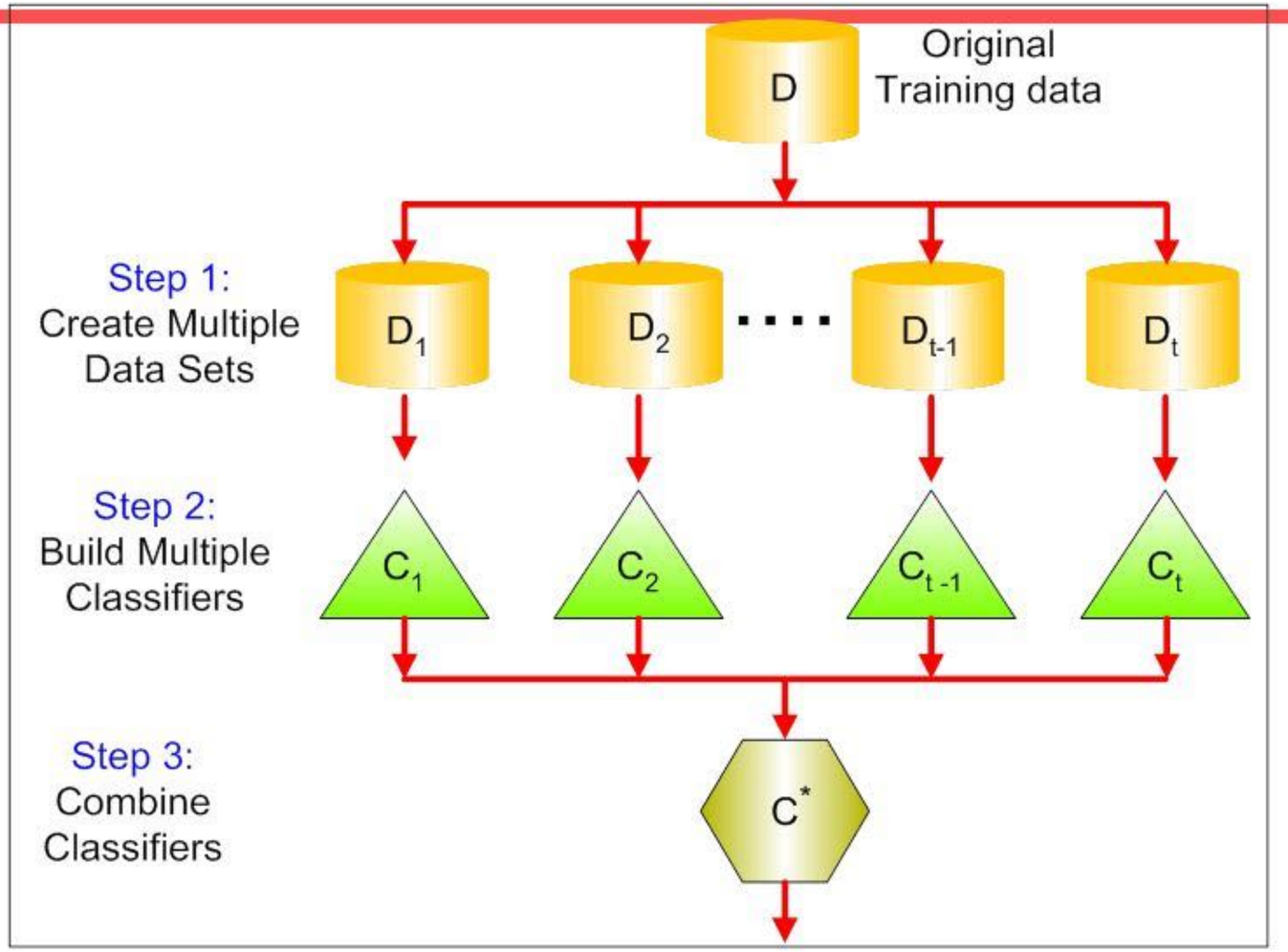
para obter resultados preditivos que os modelos

constituintes não seriam capaz de obter

Bagging

- Vimos que uma Floresta Aleatória é tipo de estratégia de ensemble
- Em especial, pode-se dizer que é uma forma de **Bagging**
- Bagging consiste em combinar o resultado de *n* classificadores *[C_1, C_2, …, C_n]*, treinados respectivamente em *n* datasets *[D_1, D_2, …, D_n]*
- Cada dataset D_i é uma amostra, com reposição, de *m* elementos do dataset original D

General Idea



Bagging

- Cada modelo C_i é considerado um **weak learner**
- O resultado dos modelos C_i podem ser combinados de diversas formas:
  - **Regressão**: Média das predições
  - **Classificação**:
    - Classe majoritária dentre as predições (voting)
    - Média das probabilidades das predições

<code> … </code>

O que mais?

- Em bagging, treinamos classificadores com amostras diferentes do dataset, porém com o mesmo algoritmo
- Por que não usar **diferentes algoritmos**?
- Podemos obter diversidade entre os *weak learners* utilizando diferentes:
  - Populações
  - Algoritmos
  - Parametrizações
  - Sementes aleatórias

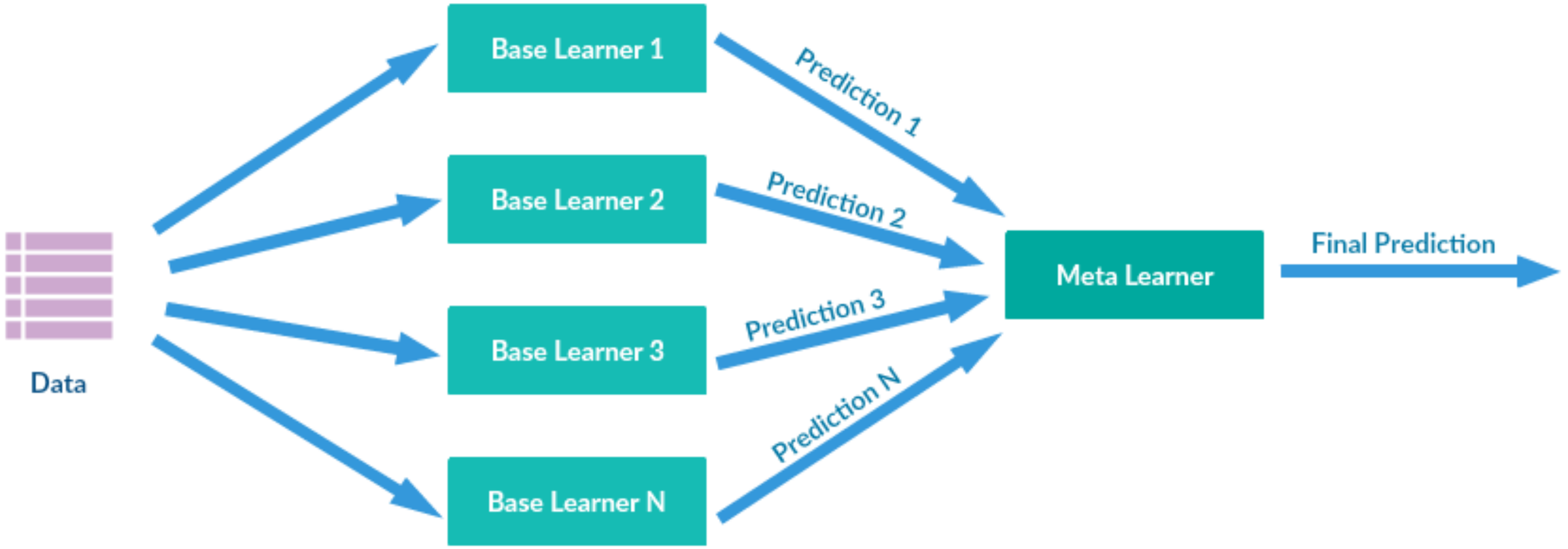
Voting

- Quaisquer modelos podem ser combinados através de qualquer estratégia
- **VotingClassifier** do SKLearn pode ser usado para isso
- E essa combinação pode ser também usada em um novo *ensemble*!
- *The sky is the limit*…

Stacking

- E se quisermos dar **pesos diferentes** a cada weak learner?
- Para cada exemplo no treino, podemos criar uma nova instância de treinamento *[P_1, P_2, …, P_n]* com as predições de cada um dos modelos *[C_1, C_2, …, C_n]*
- Esse novo modelo é chamado um *meta-learner,* e ele vai aprender a melhor forma de combinar as predições *[P_1, P_2, …, P_n]*
- Isso é chamado de Stacking, e é extremamente poderoso!

Stacking


Base Learner 1
Base Learner 2
Base Learner 3
Base Learner N
Prediction 1
Prediction 2
Prediction 3
Prediction N
Meta Learner
Final Prediction
Data

Ensembles

- Então por que não criar milhares de modelos para todos os problemas?
- Restrição computacional (treino longo e latência na predição), complexidade de implementação, dificuldade de manutenção e ganhos marginais a partir de um certo ponto
- Porém existem sim ensembles extremamente complexos e poderosos, especialmente em competições

Conclusões

- Árvores de Decisão são modelos extremamente interessantes por sua versatilidade e interpretabilidade
- Muito cuidado deve ser tomado ao parametrizar o modelo devido ao risco de *overfitting*
- São muito úteis no processo de investigação e exploração, porém não tão poderosas em termos preditivos

Conclusões

- Em contraste, sua utilização como *weak learners* em técnicas de *ensembling* produz modelos muito poderosos
- Ensembles são extremamente poderosos e abrem infinitas possibilidades de modelagem
- Na prática, porém, são difíceis de colocar em produção e manter

Estudos Adicionais

- Post-pruning: removendo nós após a construção da árvore
- Extremely randomized trees
- Stacking
- XGBoost (Gradient Boosting Machine)
- Multi-output classification

DÚVIDAS?!

Regressão

- Árvores de decisão podem também ser usadas para regressão
- Uma ideia possível é categorizar a saída em intervalos discretos e tratar o problema como classificação
- Dois aspectos precisam ser ajustados:
  - Qual métrica de impureza podemos usar para fazer os splits?
  - Que resposta retornamos nos nós folhas?

Regressão

- Ao invés de calcular a redução de entropia de cada subconjunto em um split, podemos calcular a **redução do desvio padrão:**

$$I'(S) = V(E) - EV(S)$$

- Onde V(E) é o desvio padrão de E e EV(S) é o desvio padrão esperado após o split S

$$I([[0,0],[1,1,1]]) > I([[0,1],[0,1,1]])$$

$$I'([[20,19],[7,9,6]]) > I'([[20,7],[19,9,6]])$$