Aprendizado de Máquinas
Florestas Aleatórias

Douglas Rodrigues

Universidade Federal Fluminense

Floresta Aleatória (Random Forest)

- *"Trees have one aspect thet prevents them from being the ideal tool for predictive learning, namely inaccuracy."* The Elements of Statistical Learning.
- Árvores funcionam bem com os dados que a geraram, mas não tem boa flexilidade para lidar com novas amostras.
- As **Florestas Aleatórias** combinam a simplicidade das árvores com flexibilidade, implicando em um aumento significativo de acurácia.

Floresta Aleatória (Random Forest)

Algoritmo:

1. Cria uma nova amostra, utilizando *bootstrap*, do mesmo tamanho do banco de dados original.
2. Utilizando *bootstrap*, selecionamos algumas variáveis explicativas e criamos uma árvore de decisão.
3. Repetimos os passos 1 e 2 acima quantas k vezes, gerando k árvores de decisão.
4. Para classificar uma amostra, aplicamos todas as árvores de decisão da nossa floresta, e escolhemos como classificação final o candidato que teve mais votos.

Passo 1: Cria uma nova amostra, utilizando *bootstrap*, do mesmo tamanho do banco de dados original.

- A reamostragem terá o mesmo tamanho da amostra original, e será utilizado como amostra TREINO.

Passo 1: Cria uma nova amostra, utilizando *bootstrap*, do mesmo tamanho do banco de dados original.

- A reamostragem terá o mesmo tamanho da amostra original, e será utilizado como amostra TREINO.
- Cerca de $1/3$ dos indivíduos não serão sorteados. Eles são chamados OOB (*Out-Of-Bag*).

Passo 1: Cria uma nova amostra, utilizando *bootstrap*, do mesmo tamanho do banco de dados original.

- A reamostragem terá o mesmo tamanho da amostra original, e será utilizado como amostra TREINO.
- Cerca de $1/3$ dos indivíduos não serão sorteados. Eles são chamados OOB (*Out-Of-Bag*).
- Os indivíduos OOB serão utilizados como amostra TESTE. Explicaremos mais a seguir.

Passo 2: Utilizando *bootstrap*, sorteamos algumas variáveis explicativas e criamos uma árvore de decisão.

Passo 2: Utilizando *bootstrap*, sorteamos algumas variáveis explicativas e criamos uma árvore de decisão.

Passo 3: Repetimos os passos 1 e 2 quantas k vezes, gerando k árvores de decisão.

Passo 2: Utilizando *bootstrap*, sorteamos algumas variáveis explicativas e criamos uma árvore de decisão.

Passo 3: Repetimos os passos 1 e 2 quantas k vezes, gerando k árvores de decisão.

- Para criar cada árvore de decisão, é sorteado (via *bootstrap*) uma quantidade de variáveis.
- Isso torna a árvore "menos precisa", mas evita *overfiting*.
- O padrão é sortear, para cada árvore, $\sqrt{v}$ variáveis, onde v é o número total de variáveis.

Passo 4: Classificação.

- Para realizar a classificação de novas amostras, aplicamos os indivíduos em cada uma das k árvores de decisão criadas, e confrontamos o resultado, escolhendo a classificação que obteve a maior votação.

Passo 4: Classificação.

- Para realizar a classificação de novas amostras, aplicamos os indivíduos em cada uma das k árvores de decisão criadas, e confrontamos o resultado, escolhendo a classificação que obteve a maior votação.
- Por exemplo, se k=100, e obtivemos que 73 apontaram a classificação A, enquanto que 27 apontaram como sendo do tipo B, a classificação final será tipo A, já que obteve a maioria dos votos.

Tipo A	Tipo B
73	27

Sobre as amostras OOB

- Uma amostra pode ser `Out-of-Bagging` em algumas árvores e em outras não. Quando vamos TESTAR o erro OOB da Floresta gerada, cada amostra só sera avaliada pelas árvores em que ela foi uma amostra OOB.

Sobre as amostras OOB

- Uma amostra pode ser `Out-of-Bagging` em algumas árvores e em outras não. Quando vamos TESTAR o erro OOB da Floresta gerada, cada amostra só sera avaliada pelas árvores em que ela foi uma amostra OOB.

- Exemplo: Supomos que construímos uma floresta aleatória com 10 árvores, e a amostra #3 foi OOB nas árvores 2, 5 e 7.

  Então, para avaliar o erro OOB, passamos a amostra #3 pelas árvores 2, 5 e 7, e ela será classificada pela votação majoritária que receber.

Árvore	Tipo A	Tipo B
Árvore 2	x	
Árvore 5	x	
Árvore 7		x
Vencedor:	x	

Sobre as amostras OOB

- Fazemos isso para cada amostra que foi OOB em alguma árvore.
- Nota que, se tivermos um grande número de árvores na floresta, há grande chance de todos as amostras serem OOB para alguma árvore.

Sobre as amostras OOB

- Fazemos isso para cada amostra que foi OOB em alguma árvore.
- Nota que, se tivermos um grande número de árvores na floresta, há grande chance de todos as amostras serem OOB para alguma árvore.
- Observe também que, construindo 1000 árvores, por exemplo, dificilmente uma amostra será avaliada pelas 1000 árvores, pois ela é avaliada apenas pelas árvores onde foi OOB, o que explica a necessidade de reservarmos uma amostra para VALIDAÇÃO.

Sobre o proximity

- Ao criarmos uma Floresta Aleatória, podemos pedir para ser criada também uma **matriz de proximidade** $n \times n$, onde $n_{i \times j}$ é o índice de proximidade entre os elementos i e j (entre 0 e 1). O método de construção dessa matriz é a seguinte:
  1. Iniciamos com a matriz identidade.
  2. Em cada árvore, se i e j terminam na mesma folha, somamos $+1$ a $n_{i \times j}$
  3. Renormalizamos os valores dividindo pelo número de árvores (para obter um valor entre 0 e 1).
  4. Ao final, teremos uma matriz triangular, com 1 na diagonal e valores entre 0 e 1 nas demais posições.

Sobre o proximity

- Ao criarmos uma Floresta Aleatória, podemos pedir para ser criada também uma **matriz de proximidade** $n \times n$, onde $n_{i \times j}$ é o índice de proximidade entre os elementos i e j (entre 0 e 1). O método de construção dessa matriz é a seguinte:
  1. Iniciamos com a matriz identidade.
  2. Em cada árvore, se i e j terminam na mesma folha, somamos $+1$ a $n_{i \times j}$
  3. Renormalizamos os valores dividindo pelo número de árvores (para obter um valor entre 0 e 1).
  4. Ao final, teremos uma matriz triangular, com 1 na diagonal e valores entre 0 e 1 nas demais posições.

- A matriz de proximidade é muito útil para o modelo processar automaticamente novas amostras com valores `NA` e para identificar *outliers*.

randomForest()

- Vamos carregar o banco de dados.

```
> load("heart_disease.rda")
```

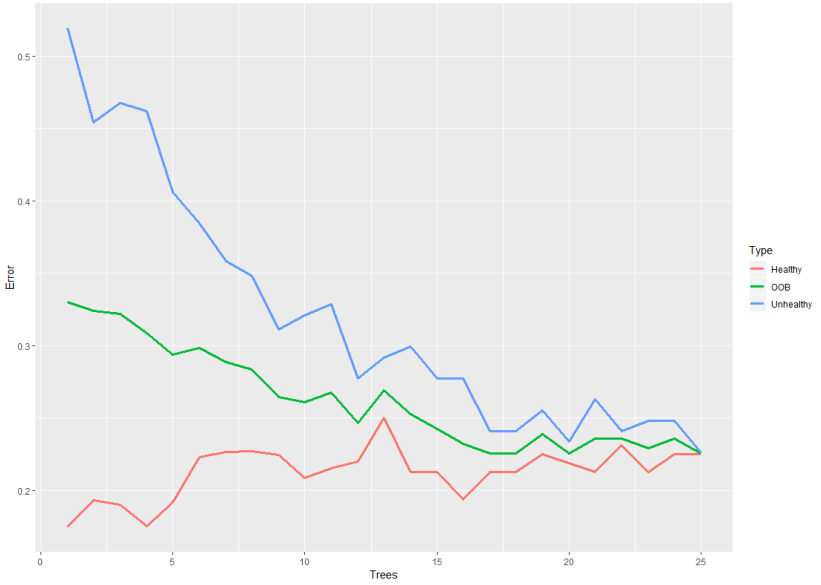
- Carregar a biblioteca, e criar a floresta aleatória.

```
> library(randomForest)
#25 árvores
> set.seed(42)
> model<-randomForest(hd ~ ., data=data, ntree=25)
```

- Avaliando o modelo.

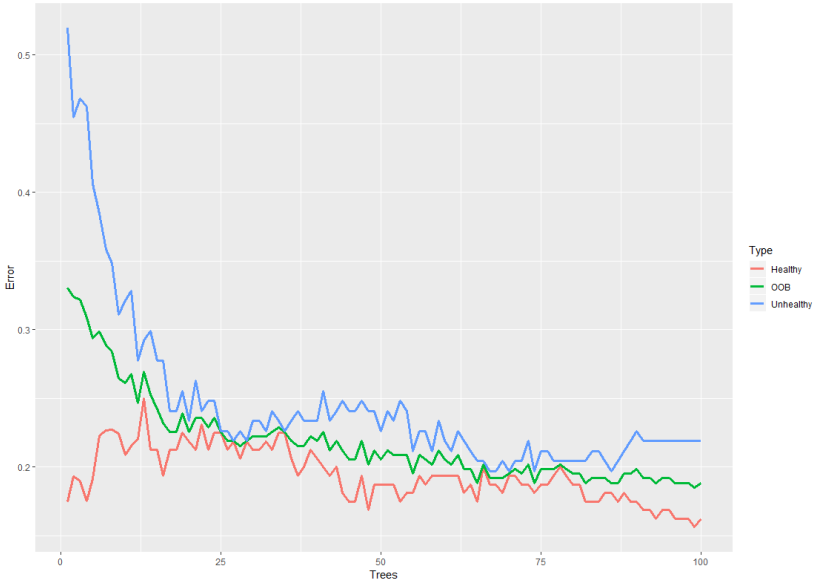
```
#Confusion matrix das amostras OOB
> model$confusion

#Votos das árvores
> model$votes
```

Evolução do Erro OOB - 25 árvores

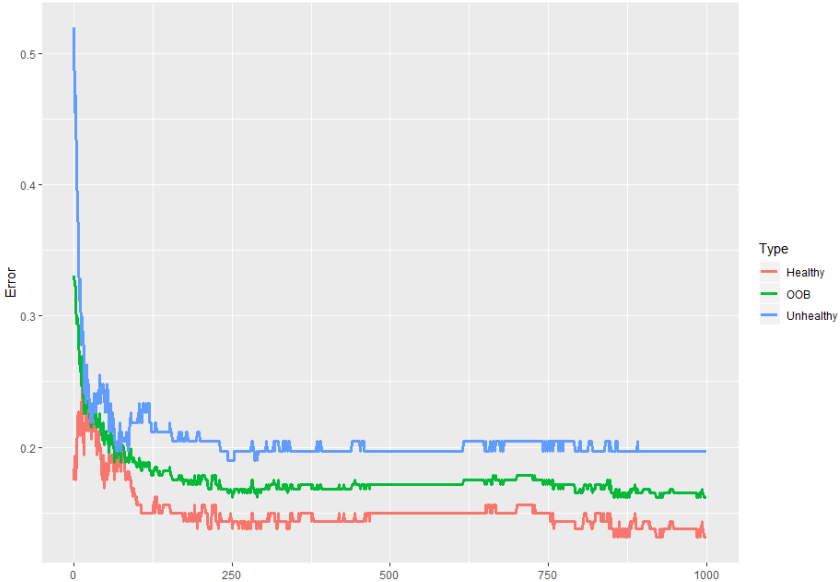


Evolução do Erro OOB - 50 árvores

Evolução do Erro OOB - 100 árvores



Evolução do Erro OOB - 1000 árvores



Evolução do Erro OOB - 10000 árvores