# Analise Multivariada II

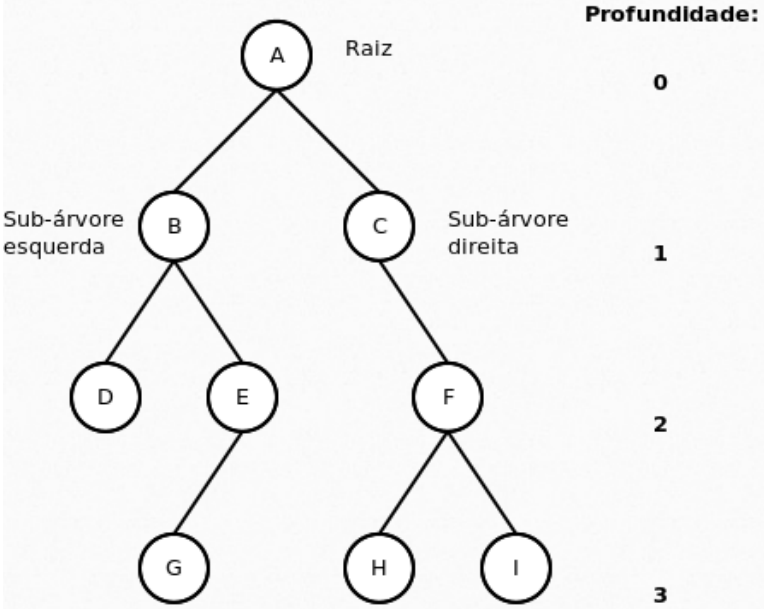
Luciane Alcoforado

setembro de 2018

Universidade Federal Fluminense
Instituto de Matemática e Estatística
Departamento de Estatística

# 1 Conceitos Iniciais

## 1.1 Árvore




Uma árvore é formada por um conjunto de elementos que armazenam informações chamados Nós. Toda a árvore possui o elemento chamado raiz, que possui ligações para outros elementos denominados ramos ou filhos. Estes ramos podem estar ligados a outros elementos que também podem possuir outros ramos. O elemento que não possui ramos é conhecido como nó folha, nó terminal ou nó externo.

## 1.2 Definição formal de árvore

Formalmente, definimos uma árvore $T$ como um conjunto finito de zero ou mais nós tal que:

- se o número de nós = 0, temos uma árvore vazia, ou
- se o número de nós > 0 existe um nó especialmente denominado raiz de $T$ os nós restantes formam $m \geq 0$ conjuntos disjuntos $p_1, p_2, \ldots, p_m$, cada um desses conjuntos é uma árvore em si, chamada subárvore da raiz de $T$, ou simplesmente subárvore.

## 1.3 Elementos da árvore

- **Nós** - são todos os itens guardados na árvore
- **Raiz** - é o nó do topo da árvore (no caso da figura acima, a raiz é o nó A)
- **Filhos** - são os nós que vem depois dos outros nós (no caso da figura acima, o nó B é filho do A)
- **Pais** - são os nós que vem antes dos outros nós (no caso da figura acima, o nó A é pai do B; no nó B é pai do D e E)
- **Folhas** - são os nós que não têm filhos; são os últimos nós da árvore (no caso da figura acima, as folhas são G, H e I)

## 1.4 Ordem da árvore

O número máximo de filhos em um nó é chamado ordem da árvore. Uma árvore binária é aquela de ordem 2, i.e., em que cada nó possui no máximo 2 filhos.

## 1.5 Caminho da árvore

Uma sequência de nós distintos $v_1, v_2, \ldots, v_k$, tal que existe sempre entre nós consecutivos (isto é, entre $v_1$ e $v_2$, entre $v_2$ e $v_3$, ... , $v_{(k-1)}$ e $v_k$) a relação “é filho de“ ou”é pai de" é denominada um caminho na árvore.

# 2 Arvore de Decisão, Classificação e Regressão

Árvores de decisão, árvores de classificação e regressão por árvore, formam uma classe de técnicas de reconhecimento de padrões em mineração de dados que têm suas origens em:

- método científico e utilizam técnicas estatísticas como testes de hipóteses e modelos de regressão e;
- redes neurais, que nasceu em pesquisas de ciência cognitiva, ao descrever o funcionamento do cérebro humano.

As árvores de decisão referem-se a representação de uma sequencia de tomada de decisão que podem ser usadas para classificação, por exemplo, para determinar a categoria de uma observação ou para previsão, para estimar um valor numérico. Usar uma árvore de decisão para classificação é uma metodologia alternativa para a regressão logística. Usar uma árvore de decisão para previsão é um método alternativo para a regressão linear. Árvore de classificação diz respeito ao método de classificação utilizado para se obter a árvore em si.

## 2.1 Vantagens de utilizar árvores de classificação

- São fáceis de interpretar
- Podem ser aplicadas em variável numérica ou categórica

- Podem lidar com dados perdidos
- São robustas para outliers
- Modelam a não linearidade dos dados
- Não necessita de nenhum pré-processamento dos dados

## 2.2 Desvantagens

- Grandes árvores podem ser difíceis de interpretar
- As árvores têm alta variação, o que faz com que o desempenho do modelo seja pobre (muitas subdivisões ou seu tamanho pode dificultar na visualização)
- Pode ocorrer overfitting (sobreajuste) o que não condiz com a realidade do fenômeno investigado

# 3 Aplicações

Exemplos da indústria:

- **Identifique os grupos-alvo**. Se você está procurando os melhores clientes em potencial para um produto, é possível identificar os nós terminais na árvore que têm a maior porcentagem de vendas e concentrar seu esforço de vendas em indivíduos descritos por esses nós.
- **Prever resultados**. Você pode querer prever resultados específicos, por exemplo, a ocorrência de crime, fraude, resultado clínico e assim por diante. Em todos esses casos, o resultado tem um valor binário, isto é, ocorreu ou não. A árvore de decisão identificará as regras que determinam se o evento ou resultado ocorrerá ou não.
- **Exploração de Dados e Detecção de Padrões**. Semelhante aos gráficos, as árvores de decisão para análise exploratória de grandes conjuntos de dados podem ajudar a detectar e visualizar relacionamentos e padrões em conjuntos muito maiores de variáveis. Por exemplo, se você estiver analisando reclamações de seguro, você pode descobrir que as reclamações de roubo são mais prováveis em casas com códigos postais de renda mais alta. Os diagramas de árvore mostrarão claramente como as reclamações são segmentadas em diferentes variáveis. Quanto maior o número de variáveis, mais valiosa é a exploração usando árvores de decisão.

## 3.1 Criando uma árvore de decisão

Na construção da árvore de decisão, procura-se associar a cada nó de decisão o atributo “mais informativo” entre aqueles ainda não utilizados no caminho desde a raiz da árvore.

Vamos obter um modelo de árvore de decisão para entender quais características dos passageiros do titanic estavam em maior risco de morte, usando um conjunto de dados denominado ptitanic. A variável resposta, chamada “survived”, indica se o passageiro sobreviveu ou não, este é um problema de classificação binária (existem apenas duas classes).

O pacote **rpart** é utilizado para gerar a árvore de decisão e **rpart.plot** para visualizar a árvore.

Neste exemplo usaremos os dados *ptitanic* do pacote **rpart.plot**:

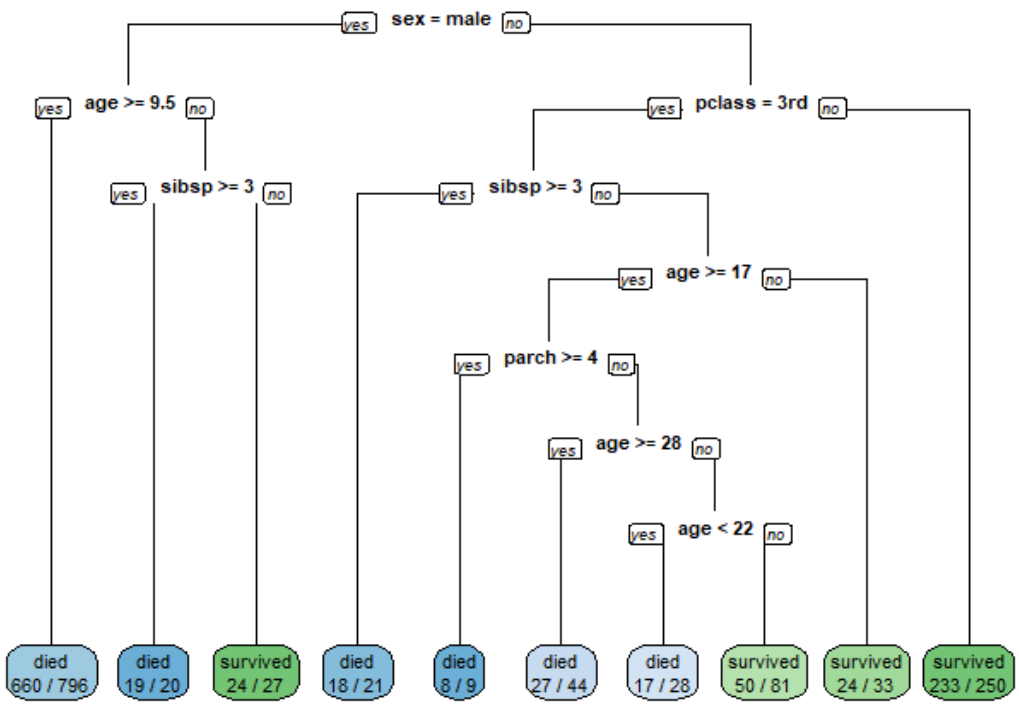
Um data frame com 1309 observações em 6 variáveis.

- **pclass** classe do passageiro, factor: 1st 2nd 3rd
- **survived**, factor: died ou survived
- **sex**, factor: male female
- **age** idade em anos, min 0.167 max 80.0
- **sibsp** número de irmãos ou cônjuge a bordo, integer: 0...8
- **parch** número de pais ou filhos a bordo, integer: 0...6

```
# Dados
library(rpart.plot)
data("ptitanic")

library(dplyr)
glimpse(ptitanic)
```

```
## Observations: 1,309
## Variables: 6
## $ pclass   <fct> 1st, 1st, 1st, 1st, 1st, 1st, 1st, 1st, 1st, 1st, 1st...
## $ survived <fct> survived, survived, died, died, died, survived, survi...
## $ sex      <fct> female, male, female, male, female, male, female, mal...
## $ age      <dbl> 29.0000, 0.9167, 2.0000, 30.0000, 25.0000, 48.0000, 6...
## $ sibsp    <int> 0, 1, 1, 1, 1, 0, 1, 0, 2, 0, 1, 1, 0, 0, 0, 0, 0, 0,...
## $ parch    <int> 0, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,...
```

```
# Modelo
library(rpart)
titanic_model <- rpart(formula = survived ~ .,
                       data = ptitanic,
                       method = "class")

# Resultado
library(rpart.plot)
rpart.plot(x = titanic_model, yesno = 2,
           type = 0, extra = 2)
```




A árvore é construída pelo seguinte processo: primeiro é encontrada uma variável (dentre as possíveis) que melhor divide os dados em dois grupos. Os dados são separados e então este processo é aplicado separadamente para cada subgrupo, e assim por diante recursivamente até que os subgrupos alcançam um tamanho mínimo ou até que nenhuma melhora seja possível.

### 3.1.1 Exercício

Responda com base no modelo:

Considere um passageiro do titanic do sexo feminino de 2a. classe, acompanhada do marido e dois filhos e com idade de 28 anos. O que o modelo prevê?

Observe que não temos ninguém com essa característica no conjunto de dados.

```
ptitanic %>%
  filter(sex=="female") %>%
  filter(pclass == "2nd") %>%
  filter(age==28) %>%
  filter(sibsp == 1) %>%
  filter(parch == 2)
```

```
## [1] pclass   survived sex      age      sibsp    parch
## <0 rows> (or 0-length row.names)
```

O modelo de árvore de classificação prevê que uma suposta pessoa com estas características teria sobrevivido com probabilidade de 233/250 (0.93) considerando apenas duas variáveis: o sexo não ser masculino e não ser de 3a. classe.

```
#Quantas mulheres da 1a. ou 2a. classe que sobreviveram/não
tab1=ptitanic %>%
  filter(sex=="female") %>%
  filter(pclass != "3rd") %>%
  group_by(survived) %>%
  summarise(freq=n()) %>%
  mutate(freqr=freq/sum(freq))

total=data.frame(survived="total", freq=sum(tab1$freq), freqr=1)

tab1 = bind_rows(tab1,total)

knitr::kable(tab1, format = "pandoc",
             caption = "Mulheres da 1a. ou 2a. classe",
             digits = 2)
```

Mulheres da 1a. ou 2a. classe

| survived | freq | freqr |
|---|---|---|
| died | 17 | 0.07 |
| survived | 233 | 0.93 |
| total | 250 | 1.00 |

## 3.2 Conjunto de treino e conjunto de teste

É importante definir um conjunto de treino para posteriormente validar o modelo através do conjunto de teste, obtendo desse modo a acurácia do modelo.

```
# Total de linhas do dataset
n <- nrow(ptitanic)

# Numero de linhas do conjunto de treino (80% do dataset)
n_train <- round(0.80 * n)

# Crie um vetor de índices de 80% do dataset
set.seed(123)
train_indices <- sample(1:n, n_train)

# Subconjunto de treinamento
titanic_train <- ptitanic[train_indices, ]
```

```
# Exclua os indices de treinamento para criar o subconjunto de teste
titanic_test <- ptitanic[-train_indices, ]
```

Agora vamos modelar:

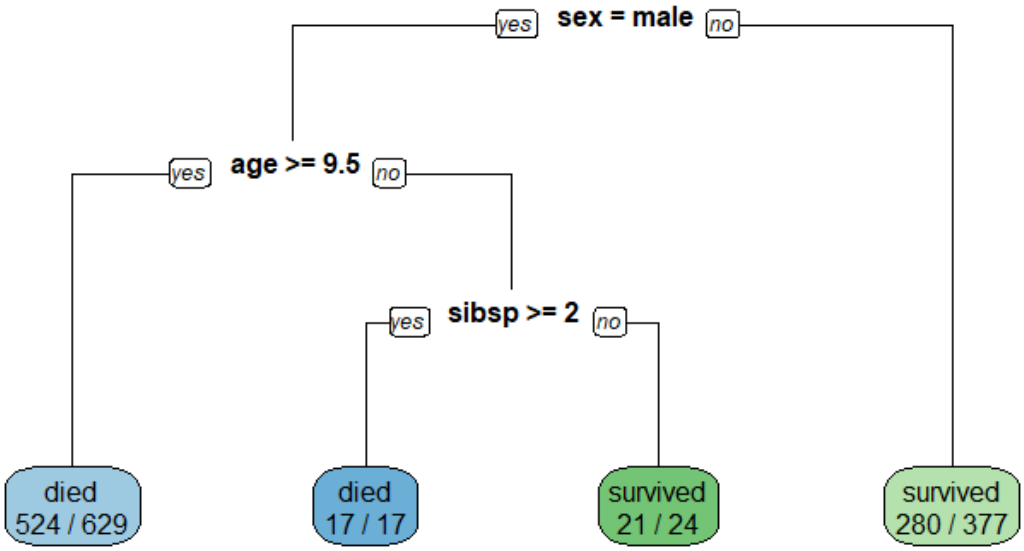
```
# Modelo de treino
titanic_model <- rpart(formula = survived~.,
                       data = titanic_train,
                       method = "class")

# Observe os detalhes do modelo
print(titanic_model)
```

```
## n= 1047
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 1047 406 died (0.6122 0.3878)
##    2) sex=male 670 126 died (0.8119 0.1881)
##      4) age>=9.5 629 105 died (0.8331 0.1669) *
##      5) age< 9.5 41  20 survived (0.4878 0.5122)
##       10) sibsp>=2 17   0 died (1.0000 0.0000) *
##       11) sibsp< 2 24   3 survived (0.1250 0.8750) *
##    3) sex=female 377  97 survived (0.2573 0.7427) *
```

Podemos verificar que no nó raiz houve 406 mortos entre 1047 passageiros ou 61% morreram e 39% sobreviveram. A variável sexo foi selecionada como o fator mais importante para determinar o desfecho (viver/morrer) com 81% dos homems tendo desfecho de morte contra 19% dos homens que sobreviveram.

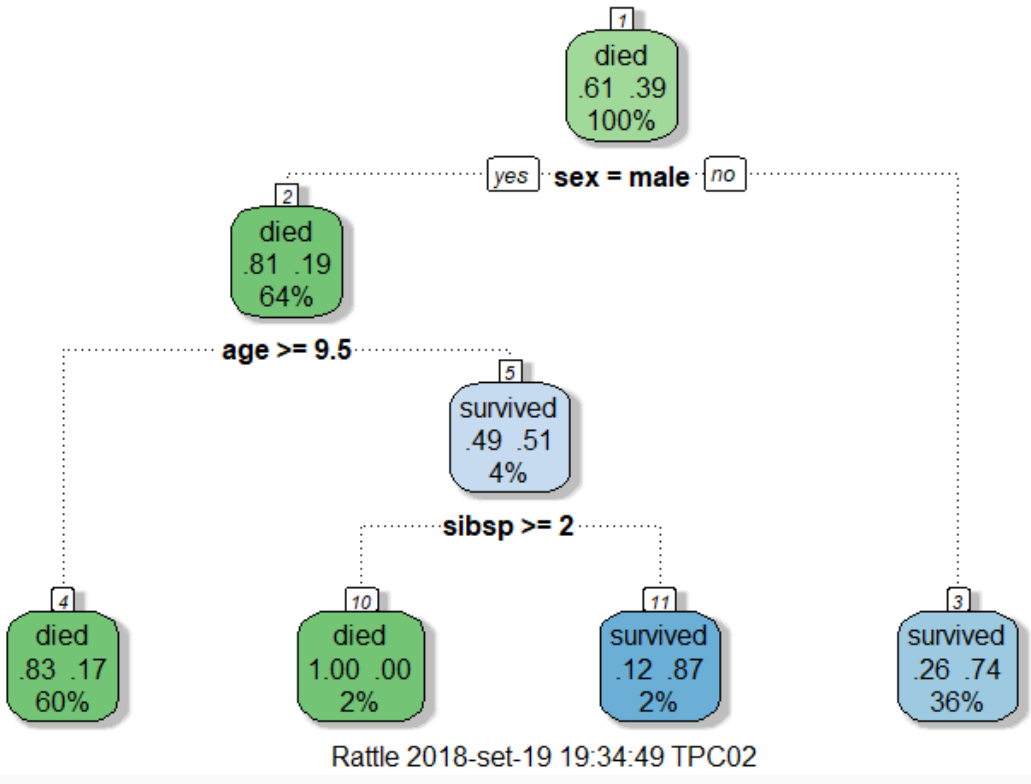
### 3.2.1 Visualizando a árvore

```
library(rpart.plot)
rpart.plot(x = titanic_model,
           yesno = 2, type = 0, extra = 2) #consulte o help da função para verificar as possib
```




Outra visualização

```
library(rattle)

# Visualizando
fancyRpartPlot(titanic_model)
```



Comparando com a regressão logística

```
modelolog=glm(survived~. ,
    data = titanic_train,
    family = binomial)
anova(modelolog, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: survived
##
## Terms added sequentially (first to last)
##
##
##        Df Deviance Resid. Df Resid. Dev             Pr(>Chi)
## NULL                     837       1138
## pclass  2     97.5       835       1041 < 0.0000000000000002 ***
## sex     1    246.2       834        795 < 0.0000000000000002 ***
## age     1     32.2       833        762          0.000000014 ***
## sibsp   1      8.7       832        754               0.0032 **
## parch   1      1.4       831        752               0.2427
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Razão de chance

```
exp(coef(modelolog))
```

```
## (Intercept)   pclass2nd   pclass3rd     sexmale         age       sibsp
##    67.41210     0.23455     0.07277     0.07032     0.95692     0.68972
##       parch
##     1.13504
```

Verificando os dados

```
library(dplyr)
glimpse(titanic_train)
```

```
## Observations: 1,047
## Variables: 6
## $ pclass   <fct> 2nd, 3rd, 2nd, 3rd, 3rd, 1st, 3rd, 3rd, 3rd, 2nd, 3rd...
## $ survived <fct> survived, died, survived, died, died, survived, died,...
## $ sex      <fct> male, male, female, female, female, female, male, mal...
## $ age      <dbl> 24, 34, 19, 47, 22, NA, 29, 16, 18, 33, NA, 4, 25, 45...
## $ sibsp    <int> 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1,...
## $ parch    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 1, 0, 0, 0, 0, 0, 2,...
```

Resumo da variável survived

```
titanic_train %>%
  group_by(survived) %>%
  summarise(freq=n()) %>%
  mutate(p=freq/sum(freq))
```

```
## # A tibble: 2 x 3
##   survived  freq     p
##   <fct>    <int> <dbl>
## 1 died       641 0.612
## 2 survived   406 0.388
```

Resumo da variável sex

```
titanic_train %>%
  group_by(sex) %>%
  summarise(freq=n()) %>%
  mutate(p=freq/sum(freq))
```

```
## # A tibble: 2 x 3
##   sex     freq     p
##   <fct>  <int> <dbl>
## 1 female   377 0.360
## 2 male     670 0.640
```

Sobrevivência quanto ao sexo masculino:

```
titanic_train %>%
  filter(sex=="male") %>%
  group_by(survived) %>%
  summarise(freq=n()) %>%
  mutate(por=freq/sum(freq))
```

```
## # A tibble: 2 x 3
##   survived  freq   por
##   <fct>    <int> <dbl>
## 1 died       544 0.812
## 2 survived   126 0.188
```

Sobrevivência quanto a idade de pelo menos 9.5 anos:

```
titanic_train %>%
  filter(age>=9.5) %>%
  group_by(survived) %>%
  summarise(freq=n()) %>%
  mutate(por=freq/sum(freq))
```

```
## # A tibble: 2 x 3
##   survived  freq   por
##   <fct>    <int> <dbl>
## 1 died       468 0.605
## 2 survived   306 0.395
```

Sobrevivência quanto a numero de irmãos ou cônjuges:

```
titanic_train %>%
  filter(sibsp==0) %>%
  group_by(survived) %>%
  summarise(freq=n()) %>%
  mutate(por=freq/sum(freq))
```

```
## # A tibble: 2 x 3
##   survived  freq   por
##   <fct>    <int> <dbl>
```

```
## 1 died       465 0.650
## 2 survived   250 0.350
```

Sobrevivência quanto ao número de pais ou filhos a bordo:

```
titanic_train %>%
  filter(parch==0) %>%
  group_by(survived) %>%
  summarise(freq=n()) %>%
  mutate(por=freq/sum(freq))
```

```
## # A tibble: 2 x 3
##   survived  freq   por
##   <fct>    <int> <dbl>
## 1 died       533 0.667
## 2 survived   266 0.333
```

### 3.2.2 Previsão e Acurácia

Agora vamos calcular a previsão do modelo para o subconjunto de teste.

```
head(predict(titanic_model, titanic_test))
```

```
##      died survived
## 4  0.8331   0.1669
## 8  0.8331   0.1669
## 11 0.8331   0.1669
## 12 0.2573   0.7427
## 18 0.2573   0.7427
## 27 0.8331   0.1669
```

```
head(predict(titanic_model, titanic_test, type="class"))
```

```
##        4        8       11       12       18       27
##     died     died     died survived survived     died
## Levels: died survived
```

Agora vamos calcular a acurácia do modelo, isto é, o número de respostas corretas/total de respostas.

```
library(caret)
class_prediction=predict(titanic_model, titanic_test, type="class")
confusionMatrix(data = class_prediction,
                reference = titanic_test$survived)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction died survived
##   died      138       33
##   survived   30       61
##
##                Accuracy : 0.76
##                  95% CI : (0.703, 0.81)
##     No Information Rate : 0.641
##     P-Value [Acc > NIR] : 0.0000263
##
##                   Kappa : 0.474
##  Mcnemar's Test P-Value : 0.801
##
##             Sensitivity : 0.821
##             Specificity : 0.649
##          Pos Pred Value : 0.807
##          Neg Pred Value : 0.670
##              Prevalence : 0.641
##          Detection Rate : 0.527
##    Detection Prevalence : 0.653
##       Balanced Accuracy : 0.735
##
##        'Positive' Class : died
##
```

## 3.3 Como funcionam as árvores de decisão?

Existem várias etapas envolvidas na construção de uma árvore de decisão.

### 3.3.1 Divisão

O processo de particionar o conjunto de dados em subconjuntos. Divisões são formadas em uma variável específica e em um local específico. Para cada divisão, duas determinações são feitas: a variável preditora usada para a divisão, chamada de variável de divisão, e o conjunto de valores para a variável preditora (que são divididos entre o nó filho à esquerda e o nó filho à direita), chamado de divisão pontual. A divisão é baseada em um critério específico, por exemplo, Gini (para classificação)

ou somas de quadrados (para regressão) do conjunto de dados inteiro. O nó da folha, também chamado de nó terminal, contém um pequeno subconjunto das observações. A divisão continua até que um nó de folha seja construído.

#### 3.3.1.1 O índice Gini

O **índice Gini** diz que se selecionarmos aleatoriamente dois itens de uma população, então ambos devem ser da mesma classe e a probabilidade disto é 1 se a população for pura. Quanto maior o valor de Gini, maior a homogeneidade; CART (Árvore de Classificação e Regressão) usa o método Gini para criar divisões binárias.

Passos para calcular o Gini de uma divisão:

Primeiro obtém-se o Cálculo do Gini para sub-nós e calcula-se a soma dos quadrados da probabilidade de sucesso e da de fracasso (p^2 + q^2)

```
#Divisão por sexo

#Gini para o sub-nó feminino
G=titanic_train %>%
  filter(sex=="female") %>%
  group_by(survived) %>%
  summarise(freq=n()) %>%
  mutate(p=freq/sum(freq))
G
```

```
## # A tibble: 2 x 3
##   survived  freq     p
##   <fct>    <int> <dbl>
## 1 died        97 0.257
## 2 survived   280 0.743
```

```
G = G %>%
  select(p)
Gf=G[1,]^2+G[2,]^2; Gf #feminino
```

```
##           p
## [1,] 0.6178
```

```r
#Gini para o sub-nó masculino
G=titanic_train %>%
  filter(sex=="male") %>%
  group_by(survived) %>%
  summarise(freq=n()) %>%
  mutate(p=freq/sum(freq))
G
```

```
## # A tibble: 2 x 3
##   survived  freq     p
##   <fct>    <int> <dbl>
## 1 died       544 0.812
## 2 survived   126 0.188
```

```r
G = G %>%
  select(p)
Gm=G[1,]^2+G[2,]^2; Gm #masculino
```

```
##           p
## [1,] 0.6946
```

Após, calcula-se o Gini ponderado para a divisão:

```r
#Gini ponderado da divisão por sexo

P=titanic_train %>%
  group_by(sex) %>%
  summarise(freq=n()) %>%
  mutate(p=freq/sum(freq))
P #peso de cada categoria
```

```
## # A tibble: 2 x 3
##   sex     freq     p
##   <fct>  <int> <dbl>
## 1 female   377 0.360
## 2 male     670 0.640
```

```
GP_sexo= Gf*P[1,3]+Gm*P[2,3]; GP_sexo
```

```
##       p
## 1 0.667
```

Vamos calcular para a divisão por classe

```
#Divisão por classe

#Gini para o sub-nó pclass=3rd
G=titanic_train %>%
  filter(pclass=="3rd") %>%
  group_by(survived) %>%
  summarise(freq=n()) %>%
  mutate(p=freq/sum(freq))
G
```

```
## # A tibble: 2 x 3
##   survived  freq     p
##   <fct>    <int> <dbl>
## 1 died       411 0.750
## 2 survived   137 0.250
```

```
G = G %>%
  select(p)
Gf=G[1,]^2+G[2,]^2; Gf #3a. classe
```

```
##          p
## [1,] 0.625
```

```
#Gini para o sub-nó 1a. ou 2a. classe
G=titanic_train %>%
  filter(pclass!="3rd") %>%
  group_by(survived) %>%
  summarise(freq=n()) %>%
```

```
  mutate(p=freq/sum(freq))
G
```

```
## # A tibble: 2 x 3
##   survived  freq     p
##   <fct>    <int> <dbl>
## 1 died       230 0.461
## 2 survived   269 0.539
```

```
G = G %>%
  select(p)
Gm=G[1,]^2+G[2,]^2; Gm #1a ou 2a classe
```

```
##           p
## [1,] 0.5031
```

```
#Gini ponderado da divisão por classe

P=titanic_train %>%
  group_by(pclass) %>%
  summarise(freq=n()) %>%
  mutate(p=freq/sum(freq))
P #peso de cada categoria
```

```
## # A tibble: 3 x 3
##   pclass  freq     p
##   <fct>  <int> <dbl>
## 1 1st      270 0.258
## 2 2nd      229 0.219
## 3 3rd      548 0.523
```

```
GP_classe= Gf*P[3,3]+Gm*(1-P[3,3]); GP_classe
```

```
##        p
## 1 0.5669
```

Vemos que o índice de Gini para divisão por sexo é 0.667 enquanto que para divisão por classe é 0.5669, quanto maior o valor melhor será a divisão, portanto neste caso o algoritmo seleciona a divisão por sexo.

#### 3.3.1.2 Exercício

1- Obtenha o índice Gini para outras variáveis. Verifique que a variável sexo é a que possui o maior índice Gini.

#### 3.3.1.3 A entropia

Entropia é uma medida utilizada na teoria da informação para avaliar o grau de desinformação de um sistema. A entropia caracteriza a (im)pureza dos dados: é uma medida da falta de homogeneidade dos dados de entrada em relação a sua classificação. Seu valor máximo é 1 e quanto menor mais homogêneo.

Seu valor é obtido por $\sum -p_i * log_2(p_i)$.

O ganho de informação é $1 - entropia$.

```
#Entropia do nó pai
E=titanic_train %>%
  group_by(survived) %>%
  summarise(freq=n()) %>%
  mutate(p=freq/sum(freq), entropia=-p*log2(p))
E
```

```
## # A tibble: 2 x 4
##   survived  freq     p entropia
##   <fct>    <int> <dbl>    <dbl>
## 1 died       641 0.612    0.433
## 2 survived   406 0.388    0.530
```

```
Ep = E %>%
  select(entropia) %>%
  summarise(sum(entropia)); Ep
```

```
## # A tibble: 1 x 1
##   `sum(entropia)`
##             <dbl>
## 1           0.963
```

```
#Entropia da divisão por sexo

#Entropia para o sub-nó feminino
E=titanic_train %>%
  filter(sex=="female") %>%
  group_by(survived) %>%
  summarise(freq=n()) %>%
  mutate(p=freq/sum(freq), entropia=-p*log2(p))


Ef = E %>%
  select(entropia) %>%
  summarise(sum(entropia))
Ef
```

```
## # A tibble: 1 x 1
##   `sum(entropia)`
##             <dbl>
## 1           0.823
```

```
#Entropia para o sub-nó masculino
E=titanic_train %>%
  filter(sex=="male") %>%
  group_by(survived) %>%
  summarise(freq=n()) %>%
  mutate(p=freq/sum(freq), entropia=-p*log2(p))


Em = E %>%
  select(entropia) %>%
  summarise(sum(entropia))
Em
```

```
## # A tibble: 1 x 1
##   `sum(entropia)`
```

```
##             <dbl>
## 1           0.697
```

```
#Entropia para Divisão por Sexo = entropia ponderada de sub-nós

P=titanic_train %>%
  group_by(sex) %>%
  summarise(freq=n()) %>%
  mutate(p=freq/sum(freq))
P #peso de cada categoria
```

```
## # A tibble: 2 x 3
##   sex     freq     p
##   <fct>  <int> <dbl>
## 1 female   377 0.360
## 2 male     670 0.640
```

```
EP_sexo= Ef*P[1,3]+Em*(1-P[1,3]); EP_sexo
```

```
##   sum(entropia)
## 1        0.7425
```

#### 3.3.1.4 Exercício

1- Obtenha a entropia de outras variáveis. Verifique que sexo é a variável que possui a menor entropia.

### 3.3.2 Comparando modelo baseando em Gini e em ganho de informação (entropia)

```
# Modelo de Treino baseado em gini
model1 <- rpart(formula = survived ~ .,
                        data = titanic_train,
                        method = "class",
                        parms = list(split = "gini"))

# Modelo de Treino baseado em informação
model2 <- rpart(formula = survived ~ .,
                        data = titanic_train,
```

```
                      method = "class",
                      parms = list(split = "information"))

# Gerando predições no conjunto de teste usando o modelo de Gini
pred1 <- predict(object = model1,
             newdata = titanic_test,
             type = "class")

# Gerando predições no conjunto de teste usando o modelo de Informação
pred2 <- predict(object = model2,
             newdata = titanic_test,
             type = "class")

# Comparando os erros de classificação
Metrics::ce(actual = titanic_test$survived,
   predicted = pred1)
```

```
## [1] 0.2405
```

```
Metrics::ce(actual = titanic_test$survived,
   predicted = pred2)
```

```
## [1] 0.2672
```

```
#Comparando a acurácia dos modelos
Metrics::accuracy(actual = titanic_test$survived,
   predicted = pred1)
```

```
## [1] 0.7595
```

```
Metrics::accuracy(actual = titanic_test$survived,
   predicted = pred2)
```

```
## [1] 0.7328
```

### 3.3.3 Poda

**Poda**: é o processo de reduzir o tamanho da árvore, transformando alguns nós de ramificação em nós de folhas e removendo os nós de folha sob a ramificação original. A poda é útil porque as árvores de classificação podem se encaixar bem nos dados de treinamento, mas podem fazer um trabalho ruim de classificar novos valores. Ramos inferiores podem ser fortemente afetados por outliers. A poda permite que você encontre a próxima árvore maior e minimize o problema. Uma árvore mais simples evita muitas vezes o ajuste excessivo.

### 3.3.4 Exercício

Considere as variáveis Y: compra ou não um imóvel, X1: ter ou não emprego, X2: ter ou não filhos, X3: estar ou não casado. Temos então p=3 covariáveis. Na primeira variável teremos dois nós (Tem emprego ou não tem emprego); na segunda variável teremos 2 nós (ter ou não filhos) e na variável 3 teremos 2 nós (ser ou não casado).

Vamos considerar por simplicidade, as variáveis categorizadas em resposta binária (do tipo 1-sim ou 0-não).

| | Emprego | Filho | Casado | Compra |
|---|---|---|---|---|
| Cliente1 | 1 | 0 | 0 | Não |
| Cliente2 | 1 | 1 | 1 | Sim |
| Cliente3 | 1 | 1 | 1 | Sim |
| Cliente4 | 1 | 0 | 0 | Não |
| Cliente5 | 0 | 0 | 1 | Não |
| Cliente6 | 0 | 1 | 1 | Sim |
| Cliente7 | 1 | 0 | 0 | Não |
| Cliente8 | 1 | 0 | 0 | Não |
| Cliente9 | 0 | 1 | 1 | Sim |
| Cliente10 | 0 | 1 | 1 | Sim |
| Cliente11 | 0 | 1 | 1 | Sim |
| Cliente12 | 0 | 1 | 1 | Sim |
| Cliente13 | 1 | 0 | 0 | Não |

| | Emprego | Filho | Casado | Compra |
|---|---|---|---|---|
| Cliente14 | 0 | 1 | 1 | Sim |
| Cliente15 | 1 | 1 | 1 | Sim |
| Cliente16 | 1 | 0 | 0 | Não |
| Cliente17 | 1 | 0 | 0 | Não |
| Cliente18 | 0 | 1 | 1 | Sim |
| Cliente19 | 1 | 1 | 1 | Sim |
| Cliente20 | 0 | 1 | 1 | Sim |

1- Obtenha uma árvore de classificação para a variável compra.

2- Realize uma pesquisa sobre **tema a escolha** em que se aplique a técnica de árvore de classificação. O trabalho deve mencionar a fonte de obtenção dos dados. (os dados devem ser reais e não simulados). Você pode optar por realizar uma pesquisa aplicando um questionário. Sua base de dados não deve conter menos do que 200 observações e 5 variáveis. Entrega do trabalho: prazo de 2 semanas.

```
library(rpart)
library(visNetwork)
#Vamos obter uma árvore de classificação com base no modelo Compra~Casado+Filho

tree1 = rpart(Compra~Casado+Filho, data = d, method = "class")
tree1
```

```
## n= 200
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 200 93 Sim (0.4650 0.5350)
##   2) Filho=0 93  0 Não (1.0000 0.0000) *
##   3) Filho=1 107  0 Sim (0.0000 1.0000) *
```

```
summary(tree1)
```