Árvores de Classificação e Regressão

EST0133 - Introdução à Modelagem de Big Data

Marcus Nunes
https://introbigdata.org/
https://marcusnunes.me/

Universidade Federal do Rio Grande do Norte

Introdução

Introdução

- CART é a abreviação de Classification and Regression Trees
- É um modelo que utiliza uma árvore de decisão como um modelo preditivo
- A ideia é mapear observações que levem a conclusões sobre as características do modelo

Ideia Geral

Ideia Geral

- O conjunto de dados `iris` possui 150 medições realizadas em três espécies de plantas
- São quatro medições para planta: comprimento e largura da sépala, comprimento e largura da pétala
- Vamos ajustar um modelo CART a estes dados

```
> library(rpart)
> modelo <- rpart(Species ~ .,
+     method="class",
+     data=iris)
```

Ideia Geral

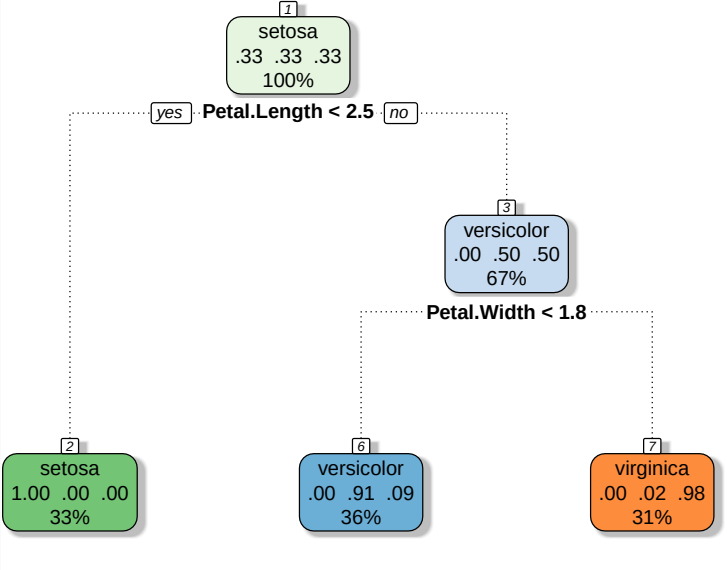
```
> print(modelo)


## n= 150
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 150 100 setosa (0.33333333 0.33333333 0.33333333)
##   2) Petal.Length< 2.45 50   0 setosa (1.00000000 0.00000000 0.00000000) *
##   3) Petal.Length>=2.45 100  50 versicolor (0.00000000 0.50000000 0.50000000)
##     6) Petal.Width< 1.75 54   5 versicolor (0.00000000 0.90740741 0.09259259) *
##     7) Petal.Width>=1.75 46   1 virginica (0.00000000 0.02173913 0.97826087) *
```

Ideia Geral

```
> library(rattle)
> fancyRpartPlot(modelo, caption=NULL)
```

Ideia Geral


1
setosa
.33 .33 .33
100%
yes
Petal.Length < 2.5
no
3
versicolor
.00 .50 .50
67%
Petal.Width < 1.8
2
setosa
1.00 .00 .00
33%
6
versicolor
.00 .91 .09
36%
7
virginica
.00 .02 .98
31%

Definições

Definições

- Seja X o espaço das variáveis aleatórias do problema a ser analisado
- O vetor $\boldsymbol{x} \in X$ contém p características $X_1, X_2, \cdots, X_p$, que podem ou não ser categorizadas
- Os classificadores em formato de árvore são construídos através de divisões sucessivas de X em dois conjuntos, iniciando-se com o próprio X

Definições

- Definições: nó, nó folha, nó pai, nó filho
- A união das regiões ocupadas por dois nós filhos é a região ocupada por seu nó pai
- Cada nó folha é alocado a uma classe

Definições

- Cada nó é designado por t
- Seu filho esquerdo é denotado por t_L e seu filho direito é denotado por t_R
- A coleção de todos os nós é denotada por T
- A coleção de todos os nós folha é denotada por $\tilde{T}$
- Cada divisão é denotada por s e seu conjunto por S

Algoritmo

Algoritmo

- A construção da árvore envolve os seguintes passos:
  1. A seleção das divisões
  2. As decisões de quando declarar um nó como folha ou continuar dividindo-o
  3. A alocação de cada nó folha a uma classe

Algoritmo

- Em particular, precisamos decidir o seguinte:
  1. Um conjunto $\mathcal{Q}$ de decisões binárias do tipo $\{X \in A\}, A \subset X$
  2. Um critério de divisão $\phi(s, t)$ que pode ser calculado para qualquer divisão s de qualquer nó t
  3. Uma regra para finalizar as divisões
  4. Uma regra para alocar cada nó folha a uma classe

Algoritmo

- O vetor $\boldsymbol{x} = (x_1, x_2, \cdots, x_p)$ contém informações sobre os dados
- Cada divisão depende do valor de uma única variável
- Para cada variável x_j, $\mathcal{Q}$ inclui todas as perguntas da forma

$$\{x_j \leq c?\}$$

  para números reais c
- Como o conjunto de treinamento é finito, há apenas um número finito de divisões distintas que podem ser geradas a partir das questões $\{x_j \leq c?\}$

Algoritmo

- Se x é categórica com valores, digamos, $\{1, 2, \cdots, M\}$, então $\mathcal{Q}$ contém todas as questões da forma

$$\{x_j \in A?\},$$

  em que A varia sob todos os subconjuntos de $\{1, 2, \cdots, M\}$
- As divisões para todas as p variáveis constituem o conjunto padrão de questões

Algoritmo

- Devido à natureza dos classificadores de árvore, transformações monótonas nas variáveis preditoras não influenciam no desempenho dos algoritmos
- Ou seja, aplicar o logaritmo em variáveis não-normais não irá alterar o resultado de um modelo de árvore, por exemplo
- Entretanto, transformações não-monótonas (quadrática, PCA) podem influenciar o resultado das análises

Algoritmo

- A qualidade da divisão é medida por uma função de impureza
- Intuitivamente, desejamos que cada nó folha seja “puro”, isto é, uma classe domina

Algoritmo

Definição
Uma função de impureza é uma função ϕ definida no conjunto de todas as K-uplas de números $(p_1, p_2, \cdots, p_k)$ satisfazendo $p_j \geq 0$, $j = 1, 2, \cdots, K$, em que $\sum_j p_j = 1$ com as propriedades

i) ϕ atinge seu máximo apenas no ponto $\left(\frac{1}{K}, \frac{1}{K}, \cdots, \frac{1}{K}\right)$

ii) ϕ atinge seu mínimo apenas nos pontos $(1, 0, \cdots, 0), (0, 1, \cdots, 0), \cdots, (0, 0, \cdots, 1)$

iii) ϕ é uma função simétrica de $p_1, p_2, \cdots, p_k$, isto é, ϕ permanece constante se os p_j forem permutados

Algoritmo

Definição
Dada uma função de impureza ϕ, defina a medida de impureza $i(t)$ de um nó como sendo

$$i(t) = \phi(p(1|t), p(2|t), \cdots, p(K|t))$$

em que $p(j|t)$ é a probabilidade estimada da classe j dentro do nó t

Algoritmo

- A qualidade de uma divisão s para o nó t, denotada por $\phi(s, t)$, é definida por

$$\begin{aligned}\phi(s, t) &= \Delta i(s, t) \\ &= i(t) - p_R i(t_R) - p_L i(t_L)\end{aligned}$$

em que p_L e p_R são as proporções das amostras do nó t que vão para o nó esquerdo t_L e para o nó direito t_R, respectivamente

Algoritmo

- Defina $I(t) = i(t)p(t)$, isto é, a função de impureza do nó t ponderada pela proporção estimada dos dados que vão para o nó t
- A impureza da árvore T, $I(T)$ é definida por

$$\begin{aligned} I(T) &= \sum_{t \in \tilde{T}} I(t) \\ &= \sum_{t \in \tilde{T}} i(t)p(t) \end{aligned}$$

Algoritmo

- Note que para qualquer nó t valem as seguintes equações:

$$
\begin{aligned}
p(t_L) + p(t_R) &= p(t) \\
p_L &= p(t_L)/p(t) \\
p_R &= p(t_R)/p(t) \\
p_L + p_R &= 1
\end{aligned}
$$

Algoritmo

- Defina

$$
\begin{aligned}
\Delta &= I(t) - I(t_L) - I(t_R) \\
&= p(t)i(t) - p(t_L)i(t_L) - p(t_R)i(t_R) \\
&= p(t)\left(i(t) - p_L i(t_L) - p_R i(t_R)\right) \\
&= p(t)\Delta i(s,t)
\end{aligned}
$$

Algoritmo

- Funções de impureza possíveis:
  1. Entropia: $\sum_{j=1}^{K} p_j \log \frac{1}{p_j}$ (se $p_j = 0$, use o limite $\lim_{p_j \to 0} p_j \log p_j = 0$)
  2. Taxa de erro de classificação: $1 - \max_j p_j$
  3. Índice de Gini: $\sum_{j=1}^{K} p_j(1 - p_j) = 1 - \sum_{j=1}^{K} p_j^2$
- O índice de Gini parece funcionar melhor na prática

Algoritmo

- O número total de amostras é N e o número de amostras na classe j é N_j, $1 \leq j \leq K$
- O número de amostras no nó t é $N(t)$
- O número de amostras da classe j indo para o nó t é $N_j(t)$
- Propriedades:
  - $\sum_{j=1}^{K} N_j(t) = N(t)$
  - $N_j(t_L) + N_j(t_R) = N_j(t)$
  - Para uma árvore completa e balanceada, a soma de $N(t)$ sob todos os t no mesmo nível é N

Algoritmo

- Denote a probabilidade *a priori* da classe j por π_j
  - As prioris π_j podem ser estimadas a partir dos dados por N_j/N
  - Algumas prioris são dadas antes da análise começar
- A probabilidade estimada de uma amostra na classe j ir para o nó t é $P(t|j) = N_j(t)/N_j$
  - $P(t_L|j) + P(t_R|j) = P(t|j)$
  - Para uma árvore completa a soma dos $P(t|j)$ sob todos os t no mesmo nível é 1

Algoritmo

- A probabilidade conjunta de uma amostra estar na classe j e ir para o nó t é dada por

$$\begin{aligned} P(j,t) &= \pi_j P(t|j) \\ &= \pi_j N_j(t)/N_j \end{aligned}$$

- A probabilidade de qualquer amostra ir para o nó t é

$$\begin{aligned} P(t) &= \sum_{j=1}^{K} P(j,t) \\ &= \sum_{j=1}^{K} \pi_j N_J(t)/N_j \end{aligned}$$

Note que $P(t_L) + P(t_R) = P(t)$

Algoritmo

- A probabilidade de uma amostra estar na classe j dado que ela vai para o nó t é

$$P(j|t) = P(j,t)/P(t)$$

Para qualquer t, $\sum_{j=1}^{K} P(j|t) = 1$

Algoritmo

- Quando $\pi_j = N_j/N$, temos a seguinte simplificação:
  - $P(j|t) = N_j(t)/N(t)$
  - $P(j) = N(t)/N$
  - $P(j,t) = N_j(t)/N$

Algoritmo

- Um critério de parada simples diz para interromper a separação dos nós quando

$$\max_{s \in S} \Delta I(s, t) < \beta,$$

em que β é um valor escolhido

- Entretanto, este critério não é satisfatório
- Um nó com um pequeno decréscimo de impureza após um passo de divisão pode gerar um decréscimo grande após múltiplos níveis de divisão

Algoritmo

- Uma regra de alocação de classe designa uma classe $j = \{1, 2, \cdots, K\}$ para cada nó folha $t \in \tilde{T}$
- A classe alocada para o nó $t \in \tilde{T}$ é denotada por $\kappa(t)$
- Para uma perda 0-1, a regra de alocação de classe é dada por

$$k(t) = \arg\max_j P(j|t)$$

Exemplo

Exemplo

```
> # pacotes
>
> library(tidymodels)
> theme_set(theme_bw())
> library(vip)
```

Exemplo

```
> # semente aleatoria
>
> set.seed(4236)
>
> # 70% dos dados como treino
>
> iris_split <- initial_split(iris, prop = .70, strata = Species)
>
> # criar os conjuntos de dados de treino e teste
>
> iris_treino <- training(iris_split)
>
> iris_teste  <- testing(iris_split)
```

Exemplo

```
> # pre-processamento
>
> iris_rec <-
+   recipe(Species ~ .,
+          data = iris_treino) %>%
+   # remover observacoes de modo que todos os niveis de Species
+   # fiquem com o mesmo numero de observacoes
+   themis::step_downsample(Species) %>%
+   # center/scale
+   step_center(-Species) %>%
+   step_scale(-Species) %>%
+   # funcao para aplicar a transformacao aos dados
+   prep()
```

Exemplo

```
> # aplicar a transformacao aos dados
>
> iris_treino_t <- juice(iris_rec)
>
> # preparar o conjunto de teste
>
> iris_teste_t <- bake(iris_rec,
+     new_data = iris_teste)
```

Exemplo

```
> # definicao do tuning
>
> iris_rpart_tune <-
+   decision_tree(
+     cost_complexity = tune(),
+     tree_depth = tune(),
+     min_n = tune()) %>%
+   set_engine("rpart") %>%
+   set_mode("classification")
```

Exemplo

```
> # grid de procura
>
> iris_rpart_grid <- grid_regular(
+     cost_complexity(range(-5, -1)),
+     tree_depth(range(1, 5)),
+     min_n(range(1, 11)),
+     levels = c(5, 5, 3))
```

Exemplo

```
> # workflow
>
> iris_rpart_tune_wflow <-
+   workflow() %>%
+   add_model(iris_rpart_tune) %>%
+   add_formula(Species ~ .)
```

Exemplo

```
> # definicao da validacao cruzada
>
> set.seed(1740)
>
> iris_treino_cv <- vfold_cv(iris_treino_t, v = 5)
```

Exemplo

```
> # avaliacao do modelo
>
> iris_rpart_fit_tune <-
+   iris_rpart_tune_wflow %>%
+   tune_grid(
+     resamples = iris_treino_cv,
+     grid = iris_rpart_grid
+   )
```

Exemplo

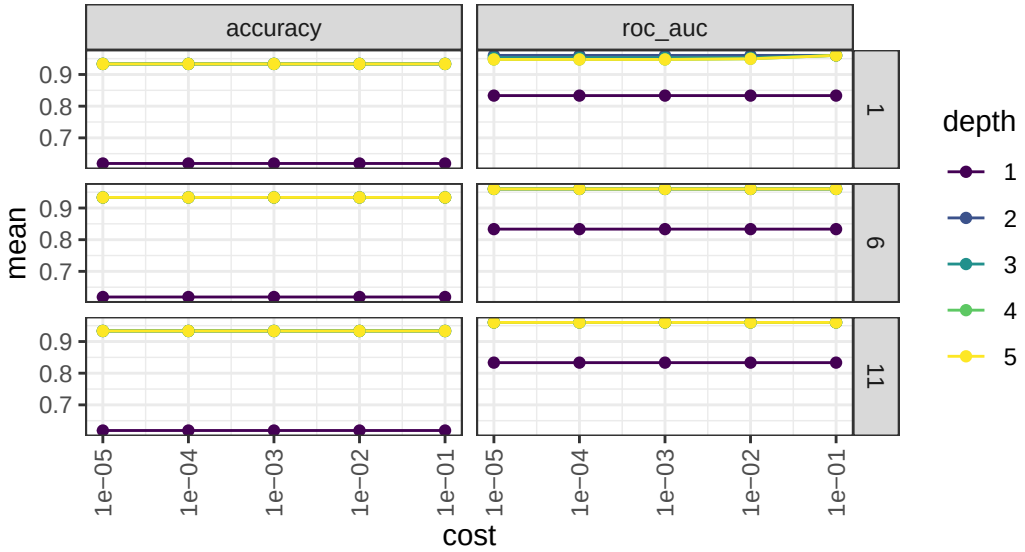
```
> # resultados
>
> collect_metrics(iris_rpart_fit_tune)

## # A tibble: 150 x 9
##    cost_complexity tree_depth min_n .metric .estimator  mean
##              <dbl>      <int> <int> <chr>   <chr>      <dbl>
##  1         0.00001          1     1 accura~ multiclass 0.619
##  2         0.00001          1     1 roc_auc hand_till  0.833
##  3         0.0001           1     1 accura~ multiclass 0.619
##  4         0.0001           1     1 roc_auc hand_till  0.833
##  5         0.001            1     1 accura~ multiclass 0.619
##  6         0.001            1     1 roc_auc hand_till  0.833
##  7         0.01             1     1 accura~ multiclass 0.619
##  8         0.01             1     1 roc_auc hand_till  0.833
```

Exemplo

```
> iris_rpart_fit_tune %>%
+   collect_metrics() %>%
+   mutate(cost = cost_complexity,
+          depth = factor(tree_depth)) %>%
+   ggplot(., aes(x = cost, y = mean, colour = depth, group = depth)) +
+   geom_line() +
+   geom_point() +
+   facet_grid(min_n ~ .metric) +
+   scale_x_continuous(trans = "log10") +
+   theme(axis.text.x = element_text(angle = 90, vjust = 0.5)) +
+   scale_colour_viridis_d()
```

Exemplo


accuracy
roc_auc
1
6
11
mean
0.9
0.8
0.7
1e−05
1e−04
1e−03
1e−02
1e−01
cost
depth
1
2
3
4
5

Exemplo

```
> # melhores modelos
>
> iris_rpart_fit_tune %>%
+   show_best("roc_auc")

## # A tibble: 5 x 9
##   cost_complexity tree_depth min_n .metric .estimator  mean
##             <dbl>      <int> <int> <chr>   <chr>      <dbl>
## 1         0.00001          2     1 roc_auc hand_till  0.960
## 2         0.0001           2     1 roc_auc hand_till  0.960
## 3         0.001            2     1 roc_auc hand_till  0.960
## 4         0.01             2     1 roc_auc hand_till  0.960
## 5         0.1              2     1 roc_auc hand_till  0.960
## # ... with 3 more variables: n <int>, std_err <dbl>,
## #   .config <chr>
```

Exemplo

```
> iris_rpart_fit_tune %>%
+   show_best("accuracy")

## # A tibble: 5 x 9
##   cost_complexity tree_depth min_n .metric  .estimator  mean
##             <dbl>      <int> <int> <chr>    <chr>      <dbl>
## 1         0.00001          2     1 accuracy multiclass 0.933
## 2         0.0001           2     1 accuracy multiclass 0.933
## 3         0.001            2     1 accuracy multiclass 0.933
## 4         0.01             2     1 accuracy multiclass 0.933
## 5         0.1              2     1 accuracy multiclass 0.933
## # ... with 3 more variables: n <int>, std_err <dbl>,
## #   .config <chr>
```

Exemplo

```
> # melhor modelo
>
> iris_rpart_best <-
+   iris_rpart_fit_tune %>%
+   select_best("accuracy")
>
> iris_rpart_final <-
+   iris_rpart_tune_wflow %>%
+   finalize_workflow(iris_rpart_best)
>
> iris_rpart_final <- fit(iris_rpart_final,
+                         iris_treino_t)
```

Exemplo

```
> iris_rpart_final

## == Workflow [trained] ==========================================
## Preprocessor: Formula
## Model: decision_tree()
##
## -- Preprocessor ----------------------------------------------
## Species ~ .
##
## -- Model -----------------------------------------------------
## n= 105
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
```

Exemplo

```
> # resultados no conjunto de teste
>
> resultado_rpart <-
+   iris_teste_t %>%
+   bind_cols(predict(iris_rpart_final, iris_teste_t) %>%
+               rename(predicao_rpart = .pred_class))
```

Exemplo

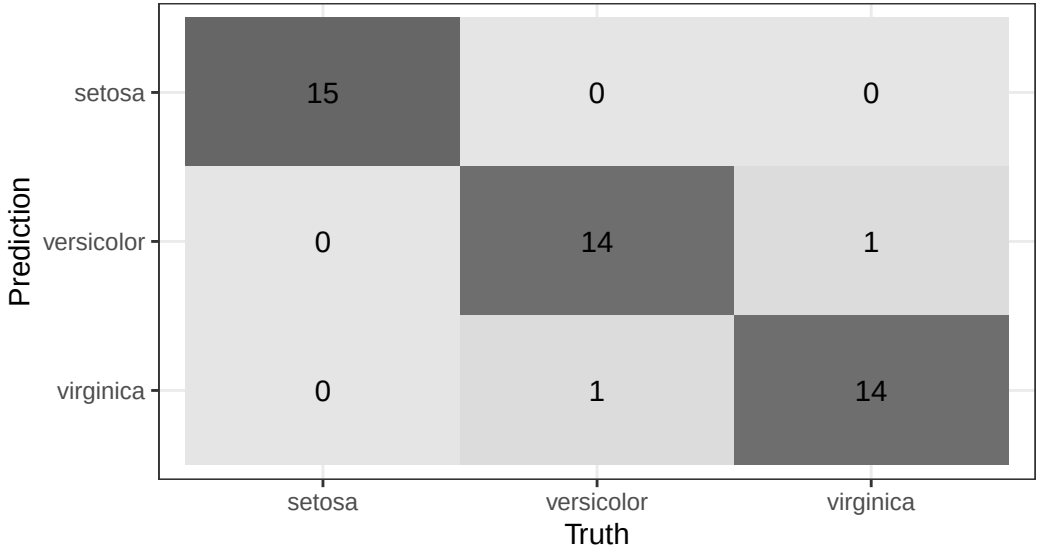
```
> metrics(resultado_rpart,
+         truth = Species,
+         estimate = predicao_rpart,
+         options = "roc")

## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.956
## 2 kap      multiclass     0.933
```

Exemplo

```
> conf_mat(resultado_rpart,
+          truth = Species,
+          estimate = predicao_rpart) %>%
+   autoplot(type = "heatmap")
```

Exemplo


Prediction
setosa
versicolor
virginica
15
0
0
0
14
1
0
1
14
setosa
versicolor
virginica
Truth

Exemplo

```
> sens(resultado_rpart,
+      truth = Species,
+      estimate = predicao_rpart)

## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 sens    macro          0.956
```

Exemplo

```
> spec(resultado_rpart,
+      truth = Species,
+      estimate = predicao_rpart)

## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 spec    macro          0.978
```

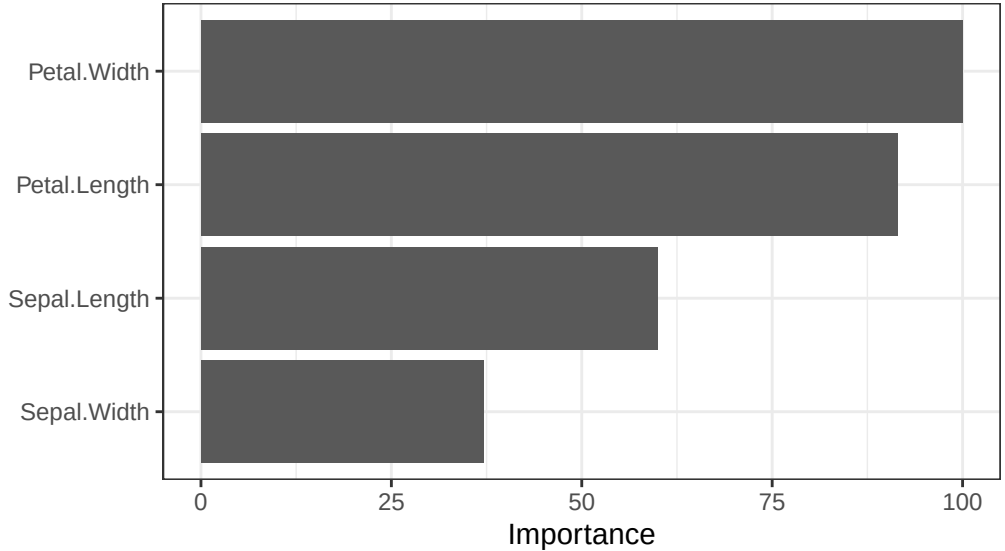
Exemplo

- Algoritmos de árvore são capazes de estimar a importância de cada variável preditora no modelo final
- Para isso, é calculado o aumento na acurácia ou RMSE de cada variável preditora como divisora primária ou substituta
- Os valores de todas estas melhoras são somadas para cada nó e cada variável
- A variável com o maior valor é considerada como 100 e as importâncias das demais são calculadas de maneira proporcional

Exemplo

```
> iris_rpart_final %>%
+   pull_workflow_fit() %>%
+   vip(scale = TRUE)
```

Exemplo


Petal.Width
Petal.Length
Sepal.Length
Sepal.Width
0
25
50
75
100
Importance

Exercícios

Exercícios

O pacote `MASS` possui um conjunto de dados chamado `Pima.tr`. Este conjunto de dados possui informações a respeito de testes sobre diabetes realizados em mulheres da tribo Pima[1], dos Estados Unidos. Este conjunto de dados possui as seguintes variáveis:

- npreg - número de gestações
- glu - concentração de glicose
- bp - pressão diastólica (mm Hg)
- skin - medida da dobra do tríceps (mm)
- bmi - índice de massa corporal
- ped - diabetes pedigree function
- age - idade (anos)
- type - diabética ou não

[1] `https://pt.wikipedia.org/wiki/Pima`

Exercícios

Além disso, este mesmo pacote possui um outro conjunto de dados, com as mesmas colunas, chamado `Pima.te`.

1. Crie um novo conjunto chamado `pima` unindo os dois conjuntos de dados originais. Utilize este novo conjunto de dados para criar os seus próprios conjuntos de treinamento e teste.

Exercícios

2. Faça a análise exploratória desse conjunto de dados, decidindo quais variáveis devem ser transformadas e quais transformações devem ser aplicadas nessas variáveis.
3. Construa a árvore de classificação desse conjunto de dados, referente ao diagnóstico de diabetes das mulheres da tribo Pima.
4. Determine se o ajuste foi bem realizado.
5. Quais são as três variáveis mais importantes para essa tarefa?

Árvores de Classificação e Regressão

EST0133 - Introdução à Modelagem de Big Data

Marcus Nunes
https://introbigdata.org/
https://marcusnunes.me/

Universidade Federal do Rio Grande do Norte