Random Forest

EST0133 - Introdução à Modelagem de Big Data

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

Universidade Federal do Rio Grande do Norte

Introdução

Introdução

- É um algoritmo derivado das árvores de classificação e regressão
- Foi criado por Tin Kam Ho em 1995 e aperfeiçoado por Leo Breiman em 2001
- Surgiu em um artigo discutindo duas culturas para análise de dados: uma derivada da estatística, outra derivada da computação
- Se tornou muito popular popular nos últimos anos, servindo como base para algoritmos mais avançados

Introdução

- É muito utilizado tanto em aplicações de classificação quanto regressão
- Pode lidar com problemas do tipo “small n large p” - problemas em que temos um tamanho amostral n muito pequeno se comparado ao número de parâmetros p do modelo
- Não é utilizada apenas para predição, podendo ser aplicada em problemas de seleção de variáveis

Introdução

- É uma combinação de várias árvores de regressão e classificação
- Parte do princípio que previsões feitas a partir da combinação de modelos são melhores do que de um modelo apenas
- Os erros dos estimadores são combinados e diminuídos, gerando assim um resultado com menor variância
- Além disso, por ser baseado em árvores, transformações monótonas nas variáveis preditoras não influenciam no desempenho dos algoritmos

Bagging

Bagging

- É uma sigla para **B**ootstrap **agg**gregat**ing**
- Combina o resultados das classificações de conjuntos de treinamento gerados aleatoriamente
- Melhora a estabilidade e a acurácia dos algoritmos, além de reduzir a variância e evitar o sobreajuste

Bagging

- Bootstrap é uma técnica de reamostragem com reposição utilizada para estimar algum parâmetro de uma população
- Assuma que dispomos de uma amostra $X_1, X_2, \cdots, X_n$ e queremos alguma informação sobre o parâmetro θ da variável aleatória X
- São tomadas B reamostras $X_1^*, X_2^*, \cdots, X_n^*$, com reposição, de tamanho n
- Calculamos a estatística de interesse $\widehat{\theta}^*$ para cada reamostra
- Assim, conseguimos construir a distribuição empírica do estimador $\widehat{\theta}$

Bagging

- Gere B subamostras com reposição a partir do conjunto de treinamento
- Treine um modelo CART em cada nova amostra
- Classificação: a classe é definida pela maioria dos votos
- Regressão: média dos valores preditos
- A estabilidade e a acurácia são melhoradas, além de reduzir a variância e evitar o sobreajuste

Algoritmo

Algoritmo

- Random forest é uma coleção de várias árvores de decisão decorrelacionadas
- Algoritmos de decorrelação são técnicas usadas para reduzir autocorrelação
- Random forest (floresta aleatória) possui este nome porque é definido através do uso de várias árvores de classificação e regressão

Algoritmo

- Suponha que temos uma matriz S composta de n amostras de treinamento, com 3 variáveis preditoras (X, Y e Z)

$$S = \begin{bmatrix} X_1 & Y_1 & Z_1 & C_1 \\ X_2 & Y_2 & Z_2 & C_2 \\ \vdots & \vdots & \vdots & \vdots \\ X_n & Y_n & Z_n & C_n \end{bmatrix}$$

Algoritmo

- A ideia é criar B subamostras aleatórias $S_1, S_2, S_3, \cdots, S_B$ da matriz S, todas de tamanho n, com reposição

$$S_1 = \begin{bmatrix} X_5 & Y_5 & Z_5 & C_5 \\ X_8 & Y_8 & Z_8 & C_8 \\ \vdots & \vdots & \vdots & \vdots \\ X_{33} & Y_{33} & Z_{33} & C_{33} \end{bmatrix}$$

$$S_2 = \begin{bmatrix} X_3 & Y_3 & Z_3 & C_3 \\ X_{20} & Y_{20} & Z_{20} & C_{20} \\ \vdots & \vdots & \vdots & \vdots \\ X_6 & Y_6 & Z_6 & C_6 \end{bmatrix}$$

Algoritmo

$$S_3 = \begin{bmatrix} X_9 & C_9 \\ X_{38} & C_8 \\ \vdots & \vdots \\ X_{45} & C_{45} \end{bmatrix}$$

$$\vdots$$

$$S_B = \begin{bmatrix} Y_1 & Z_1 & C_1 \\ Y_{12} & Z_{12} & C_{12} \\ \vdots & \vdots & \vdots \\ Y_{97} & Z_{97} & C_{97} \end{bmatrix}$$

Algoritmo

- Ajustamos, a partir de cada uma das subamostras criadas, um modelo CART diferente
- Cada um desses modelos terá um número aleatório de variáveis preditoras
- Ou seja, S_1 terá um modelo próprio com k_1 variáveis preditoras, S_2 terá outro modelo próprio com k_2 variáveis preditoras e assim por diante
- Os $k_i, i = 1, \cdots, B$ não serão necessariamente iguais

Algoritmo

- Ao fim, teremos B CARTs diferentes
- Portanto, teremos uma floresta com B árvores
- A partir destes resultados, calculamos a média das árvores estimadas no caso de regressão ou contamos a maioria de votos, no caso de classificação

Algoritmo

- De maneira mais formal, temos o seguinte algoritmo:
  1. Tome B subconjuntos de seu conjunto de dados originais, com reposição
  2. Ajuste uma CART $\widehat{f}_i$ a cada um destes subconjuntos, com um número aleatório de variáveis preditoras
  3. Encontre uma estimativa para a random forest fazendo

$$\widehat{f} = \frac{1}{B} \sum_{i=1}^{B} \widehat{f}_i$$

Algoritmo

- É possível encontrar a importância de cada variável ao rodarmos uma random forest
- Durante o processo de ajuste do modelo, o erro de ajuste para cada nó é medido e registrado
- Para medir a importância da j-ésima variável, basta permutar os seus valores dentro de cada iteração
- Assim, temos os valores dos erros de ajuste dos conjuntos de dados normais e perturbados
- Desta forma medimos a importância de cada variável

Algoritmo

- Cada vez uma divisão ocorre para a variável j, o nível de impureza para os dois nós descendentes é menor do que o do nó original
- Somando os índices de Gini para cada variável sobre todas as árvores, obtemos uma medida da importância da variável que é consistente com o do teste de permutação descrito anteriormente, só que mais rápido

Importância de Gini

- O índice de pureza Gini é definido como

$$G = \sum_{i=1}^{n_c} p_i(1 - p_i)$$

em que n_c é o número de classes na variável j e p_i é a proporção desta classe (note que este G é calculado para cada árvore na floresta)
- A partir disto, a importância é calculada como

$$I = G_{\text{pai}} - G_{\text{filho 1}} - G_{\text{filho 2}}$$

- Por fim, é calculada a média de todos os nós para todas as árvores

Exemplo

Exemplo

- O conjunto de dados `penguins` faz parte do pacote `palmerpenguins`
- Ele possui 344 observações para 8 variáveis
- Nosso objetivo é classificar as espécies de pinguins baseando-nos nas outras variáveis do conjunto de dados

Exemplo

```
> # pacotes carregados
>
> library(tidymodels)
> theme_set(theme_bw())
> library(onehot)
> library(palmerpenguins)
> library(GGally)
> library(ggfortify)
> library(vip)
```

Exemplo

```
> # checagem dos dados
>
> glimpse(penguins)

## Rows: 344
## Columns: 8
## $ species           <fct> Adelie, Adelie, Adelie, Adelie, ~
## $ island            <fct> Torgersen, Torgersen, Torgersen,~
## $ bill_length_mm    <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3~
## $ bill_depth_mm     <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6~
## $ flipper_length_mm <int> 181, 186, 195, NA, 193, 190, 181~
## $ body_mass_g       <int> 3750, 3800, 3250, NA, 3450, 3650~
## $ sex               <fct> male, female, female, NA, female~
## $ year              <int> 2007, 2007, 2007, 2007, 2007, 20~
```

Exemplo

```
> # criacao de variaveis dummy
>
> pp <-
+   penguins %>%
+   select(!where(is.numeric)) %>%
+   select(-species) %>%
+   onehot() %>%
+   predict(penguins) %>%
+   as.data.frame() %>%
+   select(i_Biscoe     = `island=Biscoe`,
+          i_Dream      = `island=Dream`,
+          i_Torgersen  = `island=Torgersen`,
+          s_fem        = `sex=female`,
+          s_male       = `sex=male`)
```

Exemplo

```
> pp <-
+   penguins %>%
+   select(where(is.numeric), species, -year) %>%
+   bind_cols(pp) %>%
+   relocate(species) %>%
+   na.omit()
```

Exemplo

```
> # treino/teste
>
> penguins %>%
+   group_by(species) %>%
+   count()

## # A tibble: 3 x 2
## # Groups:   species [3]
##   species       n
##   <fct>     <int>
## 1 Adelie      152
## 2 Chinstrap    68
## 3 Gentoo      124
```

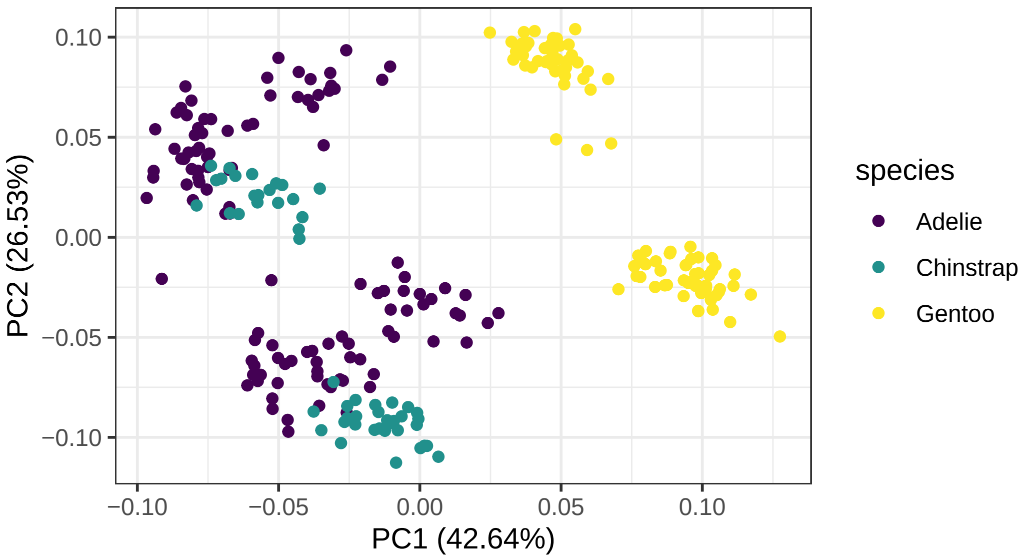
Exemplo

```
> # 75% dos dados como treino
>
> set.seed(1232)
>
> pp_split <- initial_split(pp, prop = .75, strata = species)
>
> # criar os conjuntos de dados de treino e teste
>
> pp_treino <- training(pp_split)
> pp_teste  <- testing(pp_split)
```

Exemplo

```
> # eda
>
> autoplot(prcomp(pp_treino %>% select(-species),
+                 center = TRUE, scale. = TRUE),
+          data = pp_treino,
+          colour = "species") +
+   scale_colour_viridis_d()
```

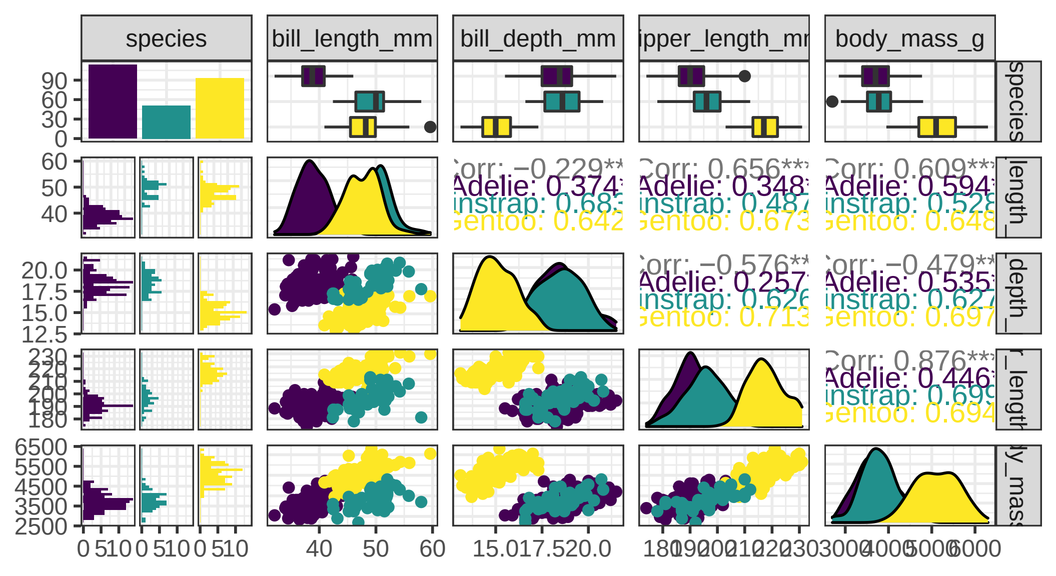
Exemplo


0.10
0.05
0.00
−0.05
−0.10
PC2 (26.53%)
−0.10
−0.05
0.00
0.05
0.10
PC1 (42.64%)
species
Adelie
Chinstrap
Gentoo

Exemplo

```
> ggpairs(pp_treino %>% select(species,
+                              bill_length_mm,
+                              bill_depth_mm,
+                              flipper_length_mm,
+                              body_mass_g),
+         aes(colour = species)) +
+   scale_colour_viridis_d() +
+   scale_fill_viridis_d()
```

Exemplo



species
bill_length_mm
bill_depth_mm
ipper_length_mr
body_mass_g
Corr: −0.229***
Adelie: 0.374**
instrap: 0.683
Gentoo: 0.642
Corr: 0.656***
Adelie: 0.348*
instrap: 0.487
Gentoo: 0.673
Corr: 0.609***
Adelie: 0.594*
instrap: 0.528
Gentoo: 0.648
Corr: −0.576***
Adelie: 0.257
instrap: 0.626
Gentoo: 0.713
Corr: −0.479***
Adelie: 0.535*
instrap: 0.627
Gentoo: 0.697
Corr: 0.876***
Adelie: 0.446*
instrap: 0.699
Gentoo: 0.694

Exemplo

```
> # pre-processamento
>
> pp_rec <-
+   recipe(species ~ .,
+          data = pp_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
>
> pp_treino_t <- juice(pp_rec)
>
> # preparar o conjunto de teste
>
> pp_teste_t <- bake(pp_rec,
+                    new_data = pp_teste)
```

Exemplo

```
> ######################
> # definicao do tuning
>
> pp_rf_tune <-
+   rand_forest(
+     mtry = tune(),
+     trees = 1000,
+     min_n = tune()
+   ) %>%
+   set_mode("classification") %>%
+   set_engine("ranger", importance = "impurity")
```

Exemplo

```
> # grid de procura
>
> pp_rf_grid <- grid_regular(mtry(range(1, 9)),
+                            min_n(range(10, 50)),
+                            levels = c(9, 5))
```

Exemplo

```
> # workflow
>
> pp_rf_tune_wflow <-
+   workflow() %>%
+   add_model(pp_rf_tune) %>%
+   add_formula(species ~ .)
```

Exemplo

```
> # definicao da validacao cruzada
>
> set.seed(2389)
>
> pp_treino_cv <- vfold_cv(pp_treino_t, v = 7)
```

Exemplo

```
> # avaliacao do modelo
>
> pp_rf_fit_tune <-
+   pp_rf_tune_wflow %>%
+   tune_grid(
+     resamples = pp_treino_cv,
+     grid = pp_rf_grid
+   )
```

Exemplo

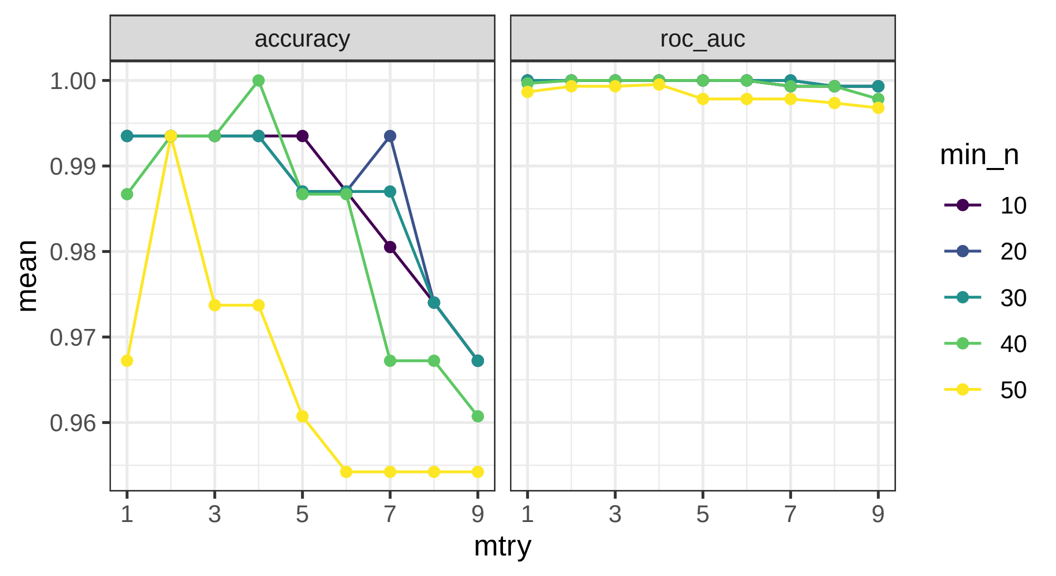
```
> # resultados
>
> collect_metrics(pp_rf_fit_tune)

## # A tibble: 90 x 8
##     mtry min_n .metric  .estimator  mean     n std_err
##    <int> <int> <chr>    <chr>      <dbl> <int>   <dbl>
##  1     1    10 accuracy multiclass 0.994     7 0.00649
##  2     1    10 roc_auc  hand_till  1         7 0
##  3     2    10 accuracy multiclass 0.994     7 0.00649
##  4     2    10 roc_auc  hand_till  1         7 0
##  5     3    10 accuracy multiclass 0.994     7 0.00649
##  6     3    10 roc_auc  hand_till  1         7 0
##  7     4    10 accuracy multiclass 0.994     7 0.00649
##  8     4    10 roc_auc  hand_till  1         7 0
```

Exemplo

```
> pp_rf_fit_tune %>%
+   collect_metrics() %>%
+   mutate(min_n = factor(min_n)) %>%
+   ggplot(., aes(x = mtry, y = mean, colour = min_n, group = min_n)) +
+   geom_line() +
+   geom_point() +
+   facet_grid(~ .metric) +
+   scale_x_continuous(breaks = seq(1, 9, 2)) +
+   scale_colour_viridis_d()
```

Exemplo



accuracy
roc_auc
1.00
0.99
0.98
0.97
0.96
mean
1
3
5
7
9
1
3
5
7
9
mtry
min_n
10
20
30
40
50

Exemplo

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

## # A tibble: 5 x 8
##    mtry min_n .metric .estimator  mean     n std_err .config
##   <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1     1    10 roc_auc hand_till      1     7       0 Prepro~
## 2     2    10 roc_auc hand_till      1     7       0 Prepro~
## 3     3    10 roc_auc hand_till      1     7       0 Prepro~
## 4     4    10 roc_auc hand_till      1     7       0 Prepro~
## 5     5    10 roc_auc hand_till      1     7       0 Prepro~
```

Exemplo

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

## # A tibble: 5 x 8
##    mtry min_n .metric .estimator  mean     n std_err .config
##   <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1     4    40 accura~ multiclass 1         7 0       Prepro~
## 2     1    10 accura~ multiclass 0.994     7 0.00649 Prepro~
## 3     2    10 accura~ multiclass 0.994     7 0.00649 Prepro~
## 4     3    10 accura~ multiclass 0.994     7 0.00649 Prepro~
## 5     4    10 accura~ multiclass 0.994     7 0.00649 Prepro~
```

Exemplo

```
> # melhor modelo
>
> pp_rf_best <-
+   pp_rf_fit_tune %>%
+   select_best("accuracy")
>
> pp_rf_final <-
+   pp_rf_tune_wflow %>%
+   finalize_workflow(pp_rf_best)
>
> pp_rf_final <- fit(pp_rf_final,
+                    pp_treino_t)
```

Exemplo

```
> # resultados no conjunto de teste
>
> resultado_rf <-
+   pp_teste_t %>%
+   bind_cols(predict(pp_rf_final, pp_teste_t) %>%
+               rename(predicao_rf = .pred_class))
```

Exemplo

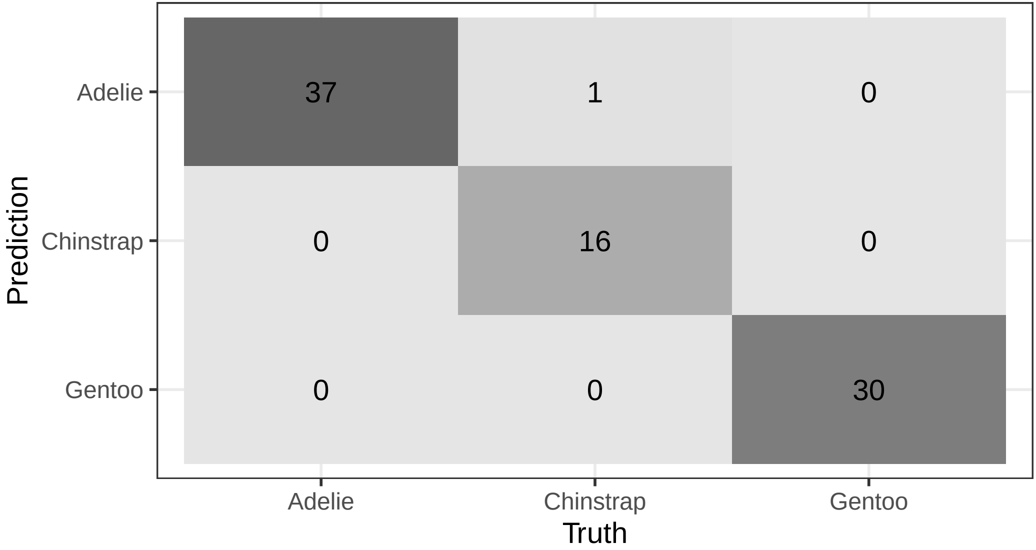
```
> metrics(resultado_rf,
+         truth = species,
+         estimate = predicao_rf,
+         options = "roc")

## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.988
## 2 kap      multiclass     0.981
```

Exemplo

```
> conf_mat(resultado_rf,
+          truth = species,
+          estimate = predicao_rf) %>%
+   autoplot(type = "heatmap")
```

Exemplo



Prediction
Adelie
Chinstrap
Gentoo
37
1
0
0
16
0
0
0
30
Adelie
Chinstrap
Gentoo
Truth

Exemplo

```
> # sensitividade
>
> sens(resultado_rf,
+      truth = species,
+      estimate = predicao_rf)

## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 sens    macro          0.980
```

Exemplo

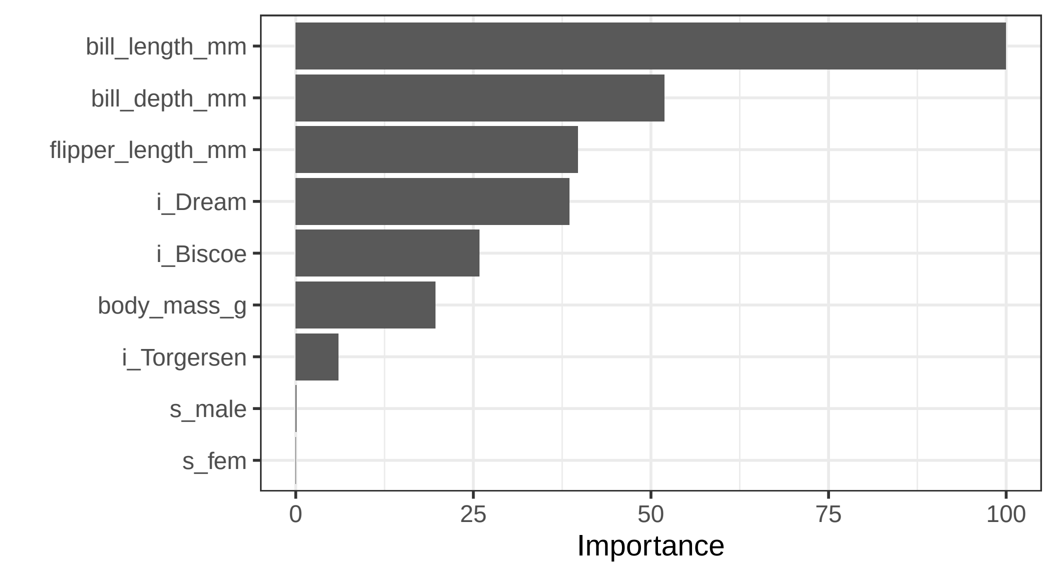
```
> # especificidade
>
> spec(resultado_rf,
+      truth = species,
+      estimate = predicao_rf)

## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 spec    macro          0.993
```

Exemplo

```
> # importancia das variaveis
>
> pp_rf_final %>%
+   pull_workflow_fit() %>%
+   vip(scale = TRUE)
```

Exemplo



bill_length_mm
bill_depth_mm
flipper_length_mm
i_Dream
i_Biscoe
body_mass_g
i_Torgersen
s_male
s_fem
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 conjuntos de treinamento e teste.

Exercícios

2. Utilize o método random forest para criar um modelo para o diagnóstico de diabetes neste conjunto de dados.
3. Encontre as variáveis mais importantes do modelo ajustado.
4. Avalie se o modelo final é bom o suficiente na sua opinião. Justifique sua resposta.
5. Compare o resultado das classificações utilizando Random Forest e CART, feito na última aula. Qual é a sua conclusão?

Random Forest

EST0133 - Introdução à Modelagem de Big Data

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

Universidade Federal do Rio Grande do Norte