APRENDENDO COM OS DADOS

UMA ABORDAGEM DE CIÊNCIA DE DADOS E APRENDIZADO DE MÁQUINA UTILIZANDO R

(PARTE 2)

Prof. Rafael G. Mantovani

31/10/2019

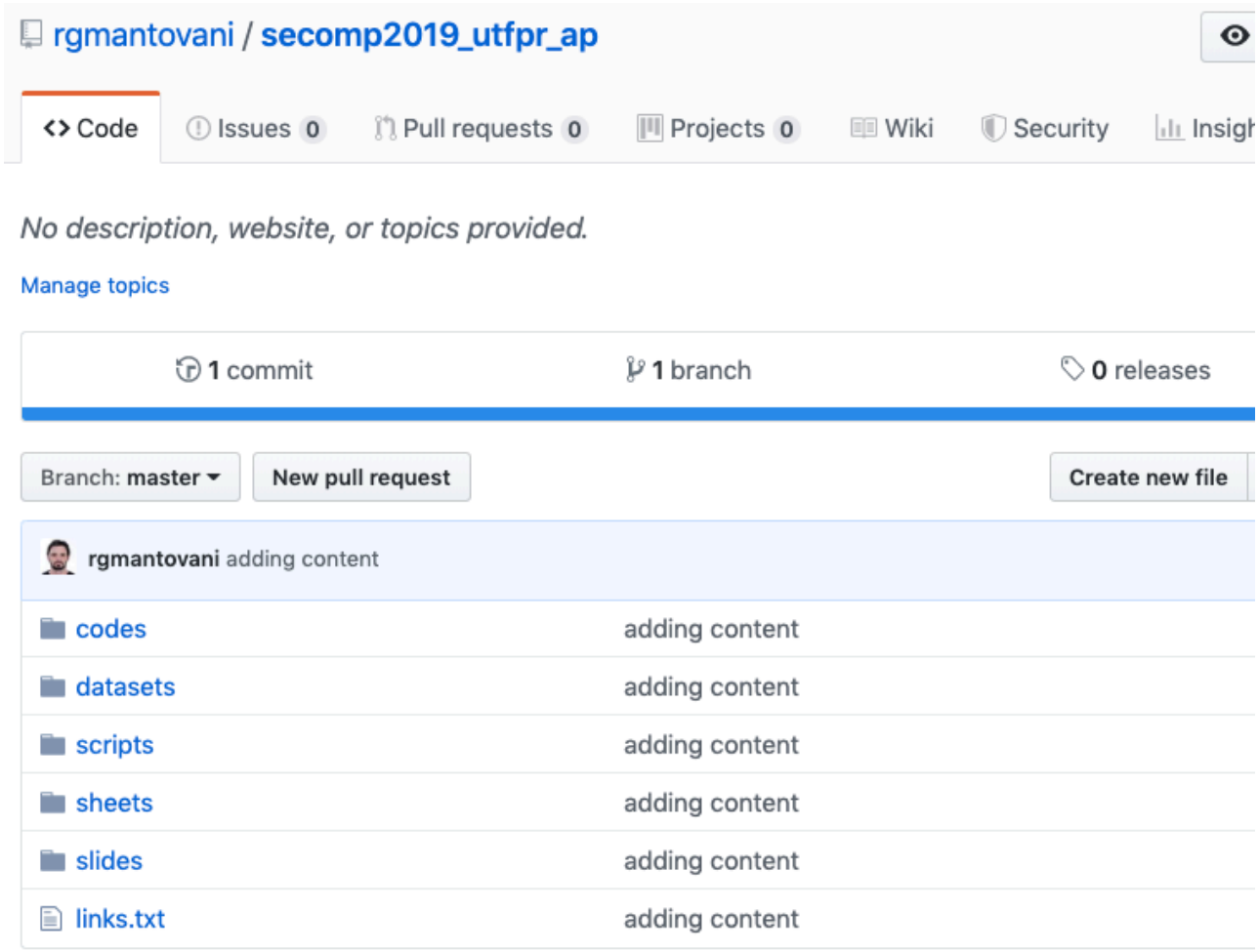
UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Apucarana - PR, Brasil
Outubro, 2019

Universidade Tecnológica Federal do Paraná (UTFPR)
I Semana de Engenharia de Computação

Material

Link: https://github.com/rgmantovani/secomp2019_utfpr_ap



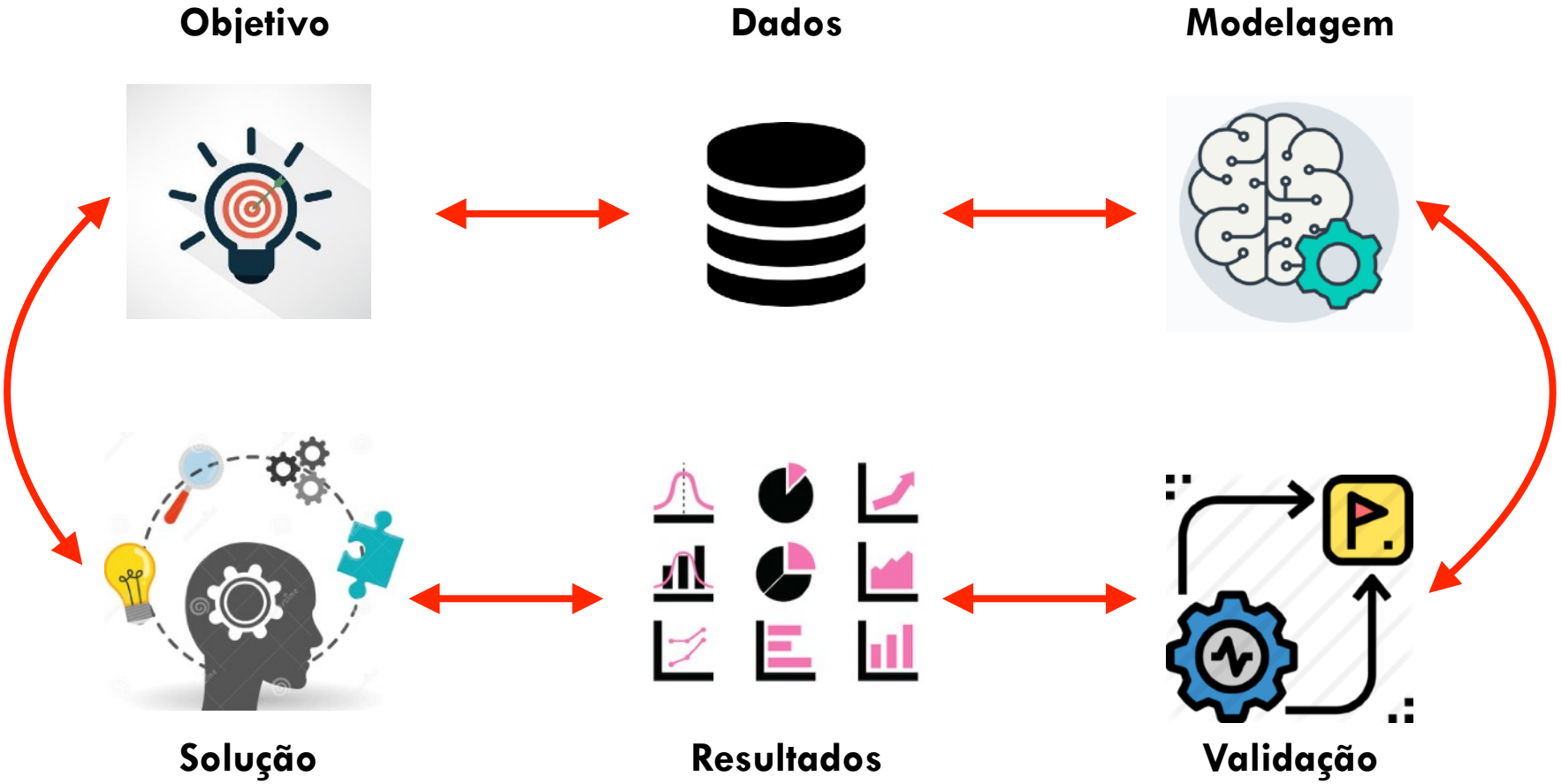
Roteiro

1. Revisão
2. Visualização dos dados (ggplot2)
3. Limpeza dos dados
4. Classificação holdout / CV
5. Benchmark
6. Titanic
7. Referências

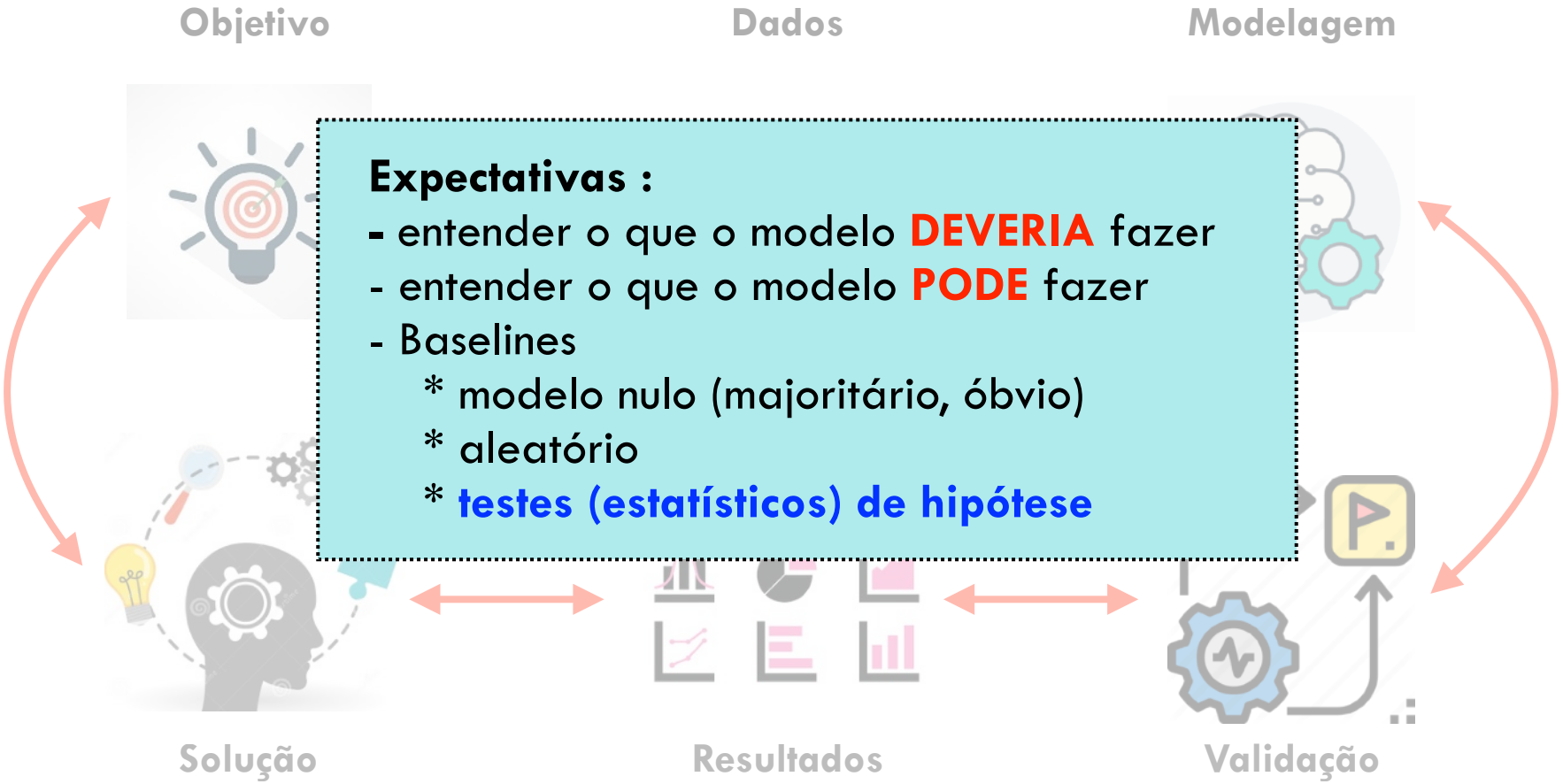
Roteiro

1. Revisão
2. Visualização dos dados (ggplot2)
3. Limpeza dos dados
4. Classificação holdout / CV
5. Benchmark
6. Titanic
7. Referências

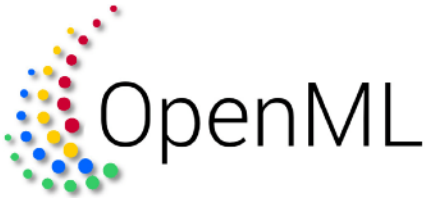
Fluxo de Ciência de Dados



Fluxo de Ciência de Dados



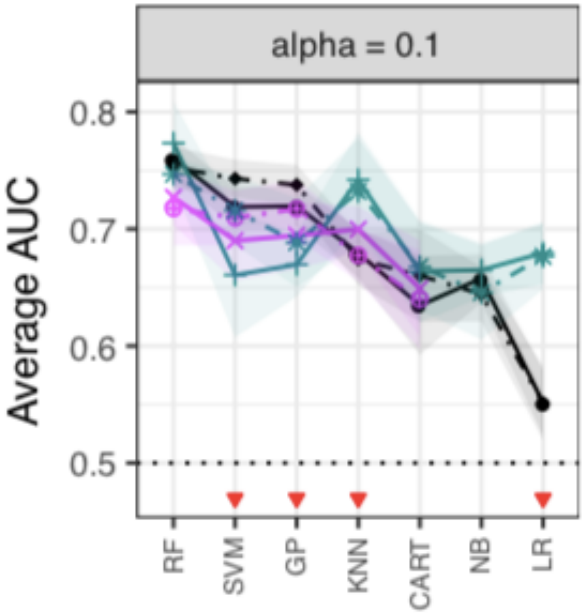
Ferramentas



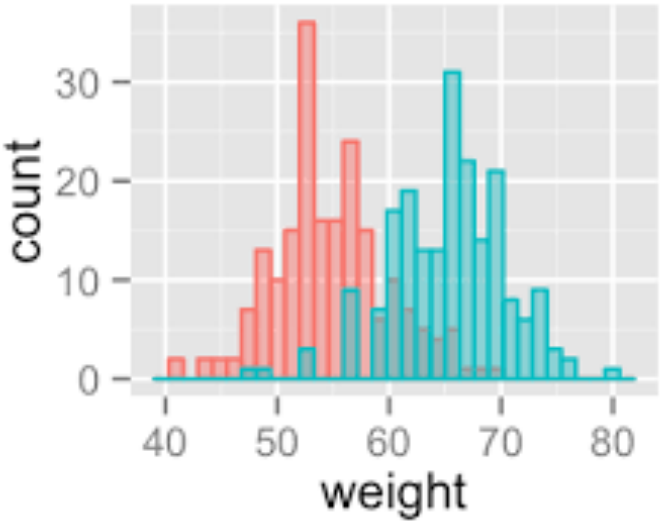
Roteiro

1. Revisão
2. Visualização dos dados (ggplot2)
3. Limpeza dos dados
4. Classificação holdout / CV
5. Benchmark
6. Titanic
7. Referências

Visualização de dados

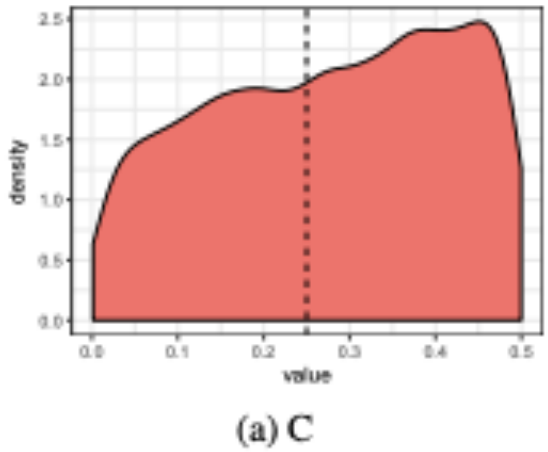


geom_line

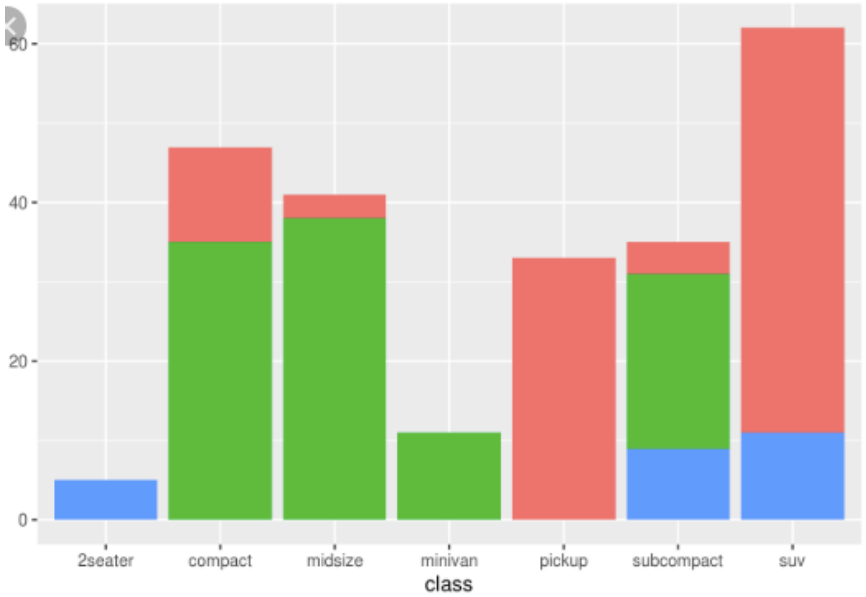


geom_histogram

Visualização de dados

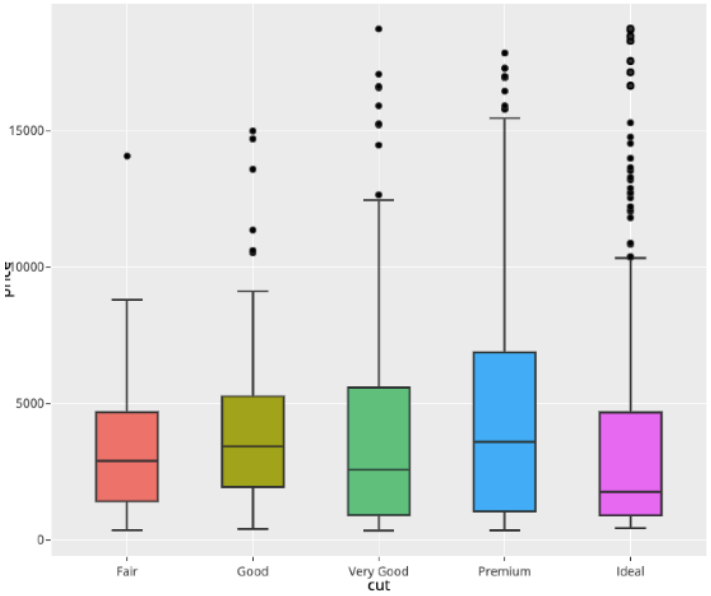


geom_density

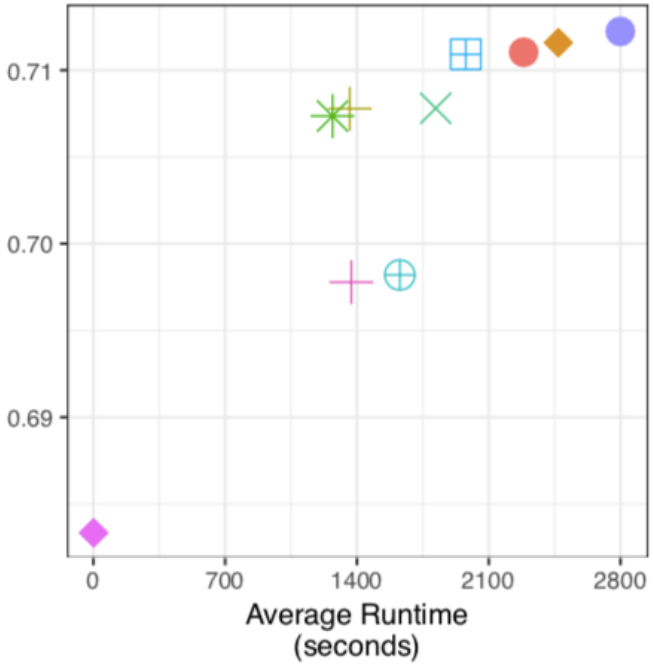


geom_bar

Visualização de dados

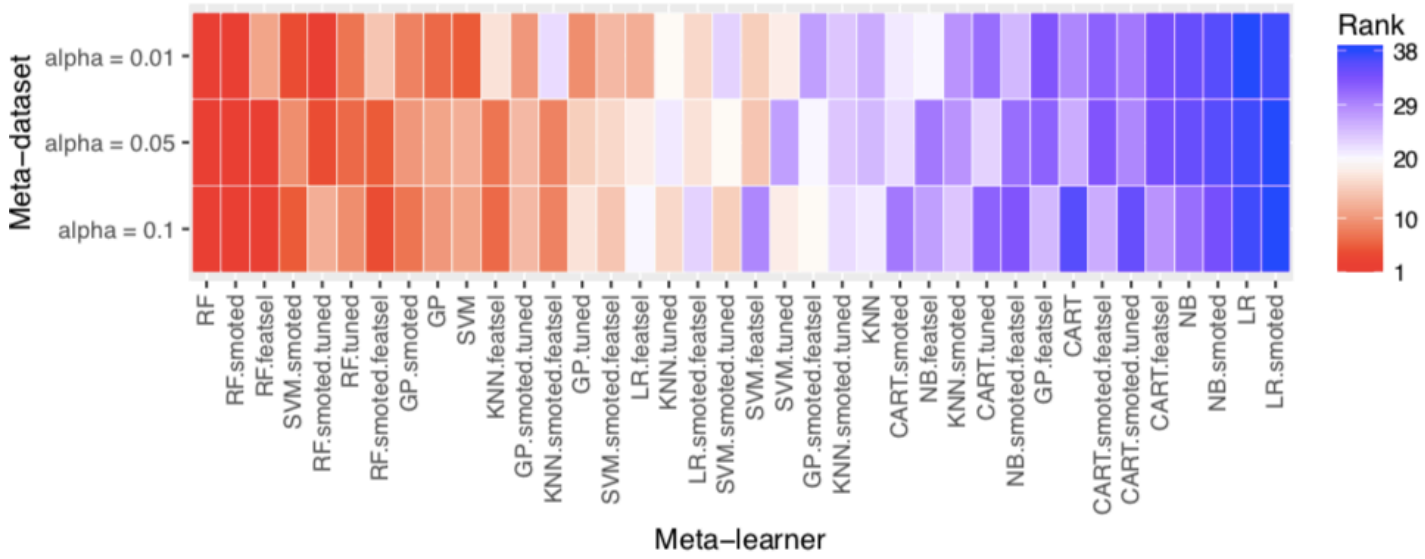


geom_boxplot



geom_point

Visualização de dados



heat maps (geom_tile)

plotting.R

```
1
2  library(ggplot2)
3
4  mpg
5
6  # plotar grafico com x = mpg$dipl e y = mph$hwy
7  ggplot(data = mpg) + geom_point(mapping = aes(x = displ, y = hwy))
8
9  # adicionando a cor das classes
10 ggplot(data = mpg) +
11   geom_point(mapping = aes(x = displ, y = hwy, colour = class))
12
13 # tamanho dos pontos
14 ggplot(data = mpg) +
15   geom_point(mapping = aes(x = displ, y = hwy, size = class))
16
17 # tamanho dos pontos + coores
18 ggplot(data = mpg) +
19   geom_point(mapping = aes(x = displ, y = hwy, colour = class, size = class))
20
21 # shape -> forma dos pontos
22 # alpha -> transparencia
```

Roteiro

1. Revisão
2. Visualização dos dados (ggplot2)
3. Limpeza dos dados
4. Classificação holdout / CV
5. Benchmark
6. Titanic
7. Referências

cleaning.R

```r
# carregar os pacotes
library(OpenML)      # para acessar e obter o dataset
library(mlr)         # para manipularmos os dados
library(ggplot2)     # para visualizar o dataset

# loading dataset usando o pacote OpenML
data = getOMLDataSet(data.id = 510)

# tambem poderia ser
# data = OpenML::getOMLDataSet(data.id = 510)

# o objeto "data" retorna muita informacao, vamos pegar apenas o dataset no slot $data
dataset = data$data
View(dataset)

# Verificar os problemas que podemos ter com os dados
# 1 - atributos (features) desnecessarios -> ids
teste = apply(dataset, 2, unique)
lapply(teste, length)

# 'species' e um atributo do tipo identificador,
# ele nao adiciona nada na resolucao do problema
ggplot(data = dataset) +
  geom_point(mapping = aes(x = species, y = overall_danger_index)) +
```

cleaning.R

- Comandos úteis:

OpenML::getDataSet ( )	Pegar dataset do site do OpenML
mlr::impute ( )	Imputar dados ausentes
caret::findCorrelation( )	Encontrar atributos correlacionados
mlr::normalizeFeatures ( )	Normaliza as features

Roteiro

1. Revisão
2. Visualização dos dados (ggplot2)
3. Limpeza dos dados
4. **Classificação holdout / CV**
5. Benchmark
6. Titanic
7. Referências

ex01_holdout.R

```r
# Carregando os pacotes necessarios para o script funcionar
library(OpenML)
library(mlr)
library(ggplot2)
library(reshape2)

# Pegar o dataset iris do OpenML (id = 61)
dataObj = getOMLDataSet(data.id = 61)
dataset = dataObj$data

# Caracteristicas gerais do dataset
summary(dataset)
colnames(dataset)[ncol(dataset)] = "Species"

# Plotar a distribuicao de classes
ggplot(data = dataset) + geom_bar(aes(x = Species,
  colour = Species, fill = Species))

# criar uma taerfa de classificacao
task = makeClassifTask(data = dataset, target = "Species")
print(task)
```

ex01_holdout.R

- Comandos úteis:

OpenML::getDataSet ( )	Pegar dataset do site do OpenML
mlr::makeClassifTask( )	Define uma tarefa preditiva
mlr::makeLearner ( )	Define um learner / algoritmo
mlr::makeResampleDesc ( )	Define método de avaliação
mlr::resample ( )	Rodamos nosso experimento
mlr::getRRPredictions ( )	Acessar o valor das predições

Exercício 01

- Replicar o script para outro dataset do OpenML
  - ler os dados via código / baixar o arquivo
  - definir algum algoritmo de classificação
    - ver aqui: https://mlr.mlr-org.com/articles/tutorial/integrated_learners.html#classification-82
  - plotar (predito x valor real)

ex02_CV.R

```r
# Dividir os dados do dataset em treino e teste, e rodar varias permutacoes
rdesc = makeResampleDesc(method = "CV", iters = 10, stratify = TRUE)

# Medidas de desempenho para avaliar os resultados
measures = list(acc, bac)

# Rodar o algoritmo na tarefa e coletar os resultados
result = resample(learner = algo, task = task, resampling = rdesc,
                  measures = measures, show.info = TRUE)
result

# mostrando o resultado
print(result$aggr)

# Checar as predicoes obtidas pelo algoritmo
result$pred
pred = getRRPredictions(res = result)
print(pred)

# tabela predicoes
table(pred$data[,2:3])

```

ex02_CV.R

- Comandos úteis:

OpenML::getDataSet ( )	Pegar dataset do site do OpenML
mlr::makeClassifTask( )	Define uma tarefa preditiva
mlr::makeLearner ( )	Define um learner / algoritmo
mlr::makeResampleDesc ( )	Define método de avaliação
mlr::resample ( )	Rodamos nosso experimento
mlr::getRRPredictions ( )	Acessar o valor das predições

Exercício 02

- Replicar o script e comparar os resultados com o exercício anterior
  - usar validação cruzada (10 partições /folds)
  - plotar (predito x valor real)

Roteiro

1. Revisão
2. Visualização dos dados (ggplot2)
3. Limpeza dos dados
4. Classificação holdout / CV
5. Benchmark
6. Titanic
7. Referências

ex03_benchmarkA.R

```r
# carregando os pacotes necessarios
library(OpenML)
library(mlr)
library(ggplot2)

# Obter o dataset do OpenML
data_obj = getOMLDataSet(data.id=61)
dataset = data_obj$data

# Criar uma tarefa de classificacao
task = makeClassifTask(data = dataset, target = "class")

# Criar uma lista de algoritmos (learners)
lrns = list(
  makeLearner("classif.lda", id = "lda"),                  # LDA - algoritmo linear
  makeLearner("classif.svm", id = "svm"),                  # SVM
  makeLearner("classif.rpart", id = "rpart"),              # DT  - arvore de decisao
  makeLearner("classif.randomForest", id = "randomForest")   # RF  - random Forest
)

```

ex03_benchmarkB.R

```
19
20  # Criar tarefas de classificacao, uma para cada dataset
21  task1 = makeClassifTask(id =  data1$desc$name, data = dataset1, target = "class")
22  task2 = makeClassifTask(id =  data2$desc$name, data = dataset2, target = "Class")
23  task3 = makeClassifTask(id =  data3$desc$name, data = dataset3, target = "Class")
24  task4 = makeClassifTask(id =  data4$desc$name, data = dataset4, target = "Class")
25  tasks = list(task1, task2, task3, task4)
26
27  # Criando uma lista de algoritmos (learners)
28  lrns = list(
29    makeLearner("classif.lda", id = "lda"),
30    makeLearner("classif.svm", id = "svm"),
31    makeLearner("classif.rpart", id = "rpart"),
32    makeLearner("classif.randomForest", id = "randomForest")
33  )
34
35  # Criando uma estrategia de validacao
36  # rdesc = makeResampleDesc("RepCV", folds = 10, reps = 10, stratify = TRUE)
37  rdesc = makeResampleDesc("CV", iters = 10, stratify = TRUE)
38
```

ex03_benchmark(A | B).R

- Comandos úteis:

OpenML::getDataSet ( )	Pegar dataset do site do OpenML
mlr::makeClassifTask( )	Define uma tarefa preditiva
mlr::makeLearner ( )	Define um learner / algoritmo
mlr::makeResampleDesc ( )	Define método de avaliação
mlr::benchmark ( )	Rodamos nosso experimento com vários algoritmos e tarefas
mlr::plotBMR… ( )	Plota os resultados do benchmark

Exercício 03

- Replicar o script:
  - Selecionar vários datasets
  - Definir vários algoritmos
  - usar validação cruzada (10 partições /folds)
  - plotar os resultados

Roteiro

1. Revisão
2. Visualização dos dados (ggplot2)
3. Limpeza dos dados
4. Classificação holdout / CV
5. Benchmark
6. Titanic
7. Referências

Desafio :)

Desafio :)

Desafio :)

- https://www.kaggle.com/c/titanic
- Titanic (OpenML data id = 40945)

- **Tarefa:**
  - dados ausentes, ruídos, desbalanceamento, etc …
  - classificação binária
    - quem sobreviveu e quem morreu?
    - quais perfis de pessoas tiveram uma chance maior de sobrevivência?

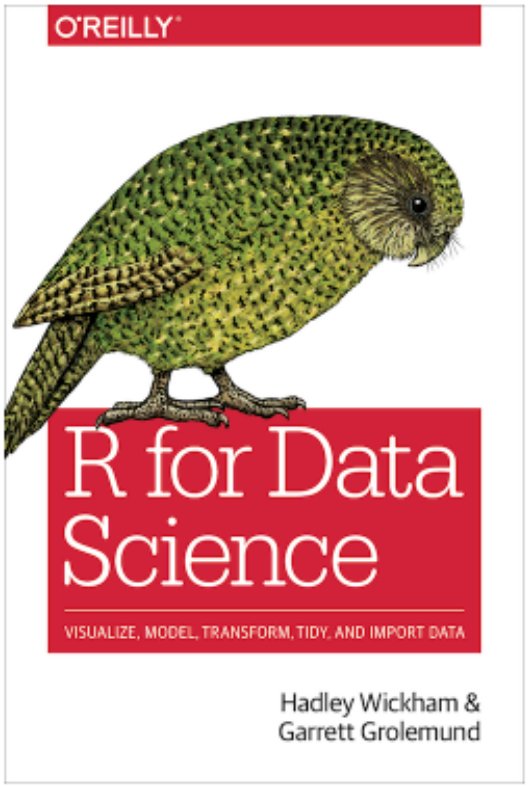
Desafio :)

- Passos:
  - 01: ler os dados
  - 02: limpar e entender os dados
  - 03: definir um algoritmo
  - 04: definir um processo de validação
  - 05: avaliar o modelo (métricas)
  - 06: mostrar os resultados e entender as predições

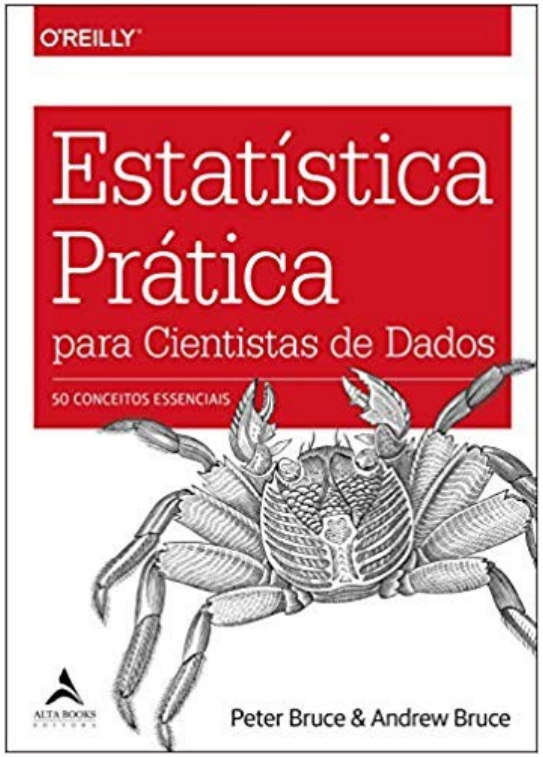
Roteiro

1. Revisão
2. Visualização dos dados (ggplot2)
3. Limpeza dos dados
4. Classificação holdout / CV
5. Benchmark
6. Titanic
7. Referências

Referências

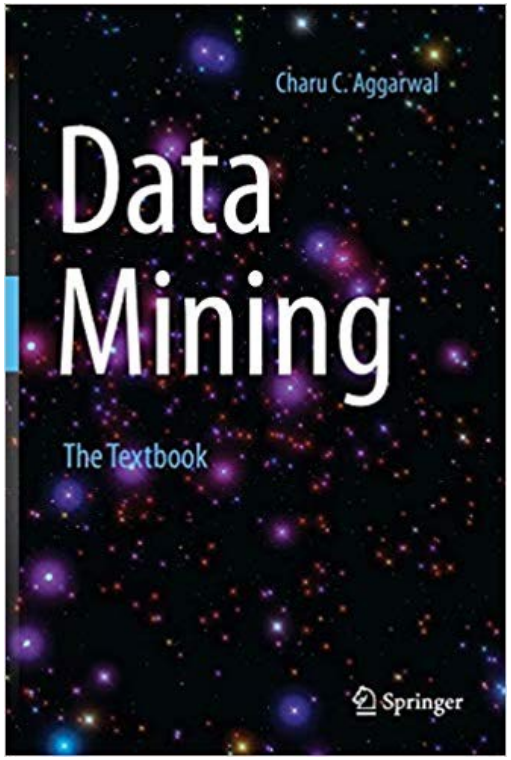


[Wickham & Grolemund, 2018]

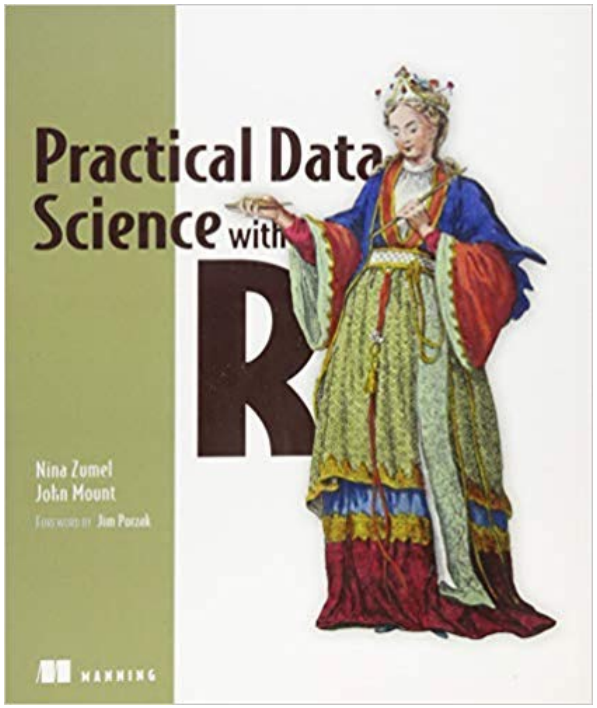


[Bruce & Bruce, 2019]

Referências



[Aggarwal, 2015]



[Zumel and Mount, 2014]

Perguntas?

Prof. Rafael G. **Mantovani**

rafaelmantovani@utfpr.edu.br