Máquinas de Vetor Suporte

EST0133 - Introdução à Modelagem de Big Data

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

Universidade Federal do Rio Grande do Norte

Introdução

Introdução

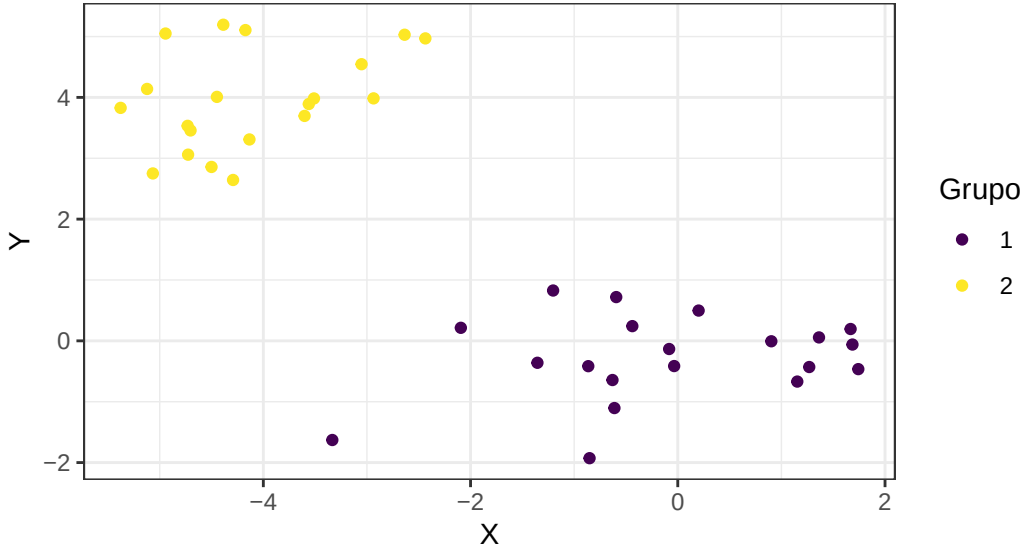
- Também conhecida por Support Vector Machines (SVM)
- É outra técnica utilizada para classificação de dados, que também pode ser usada para regressão
- Trata-se do aprimoramento de um antigo algoritmo chamado perceptron
- Os dados não necessitam ser linearmente separáveis

SVM

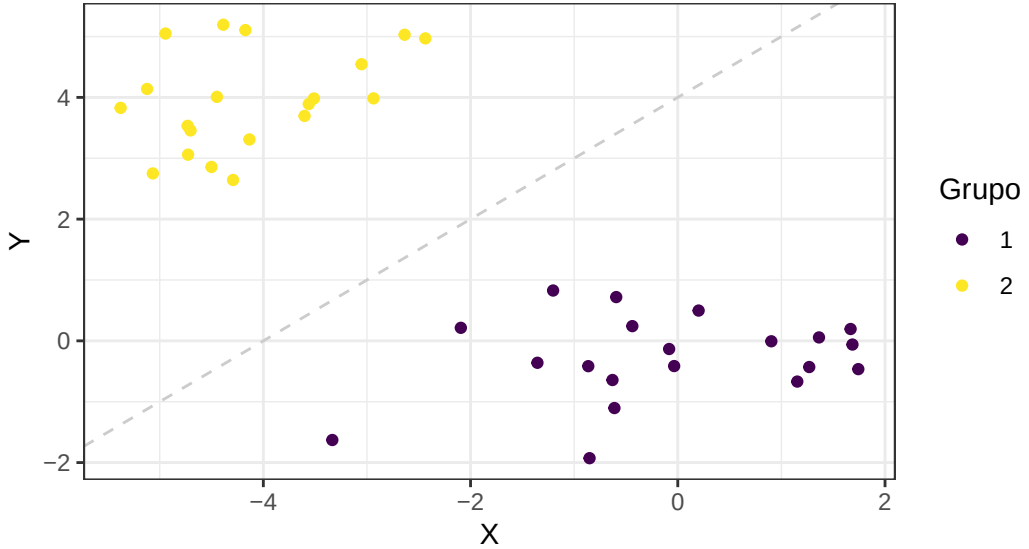
SVM

- Considere uma tarefa de classificação binária em $\mathbb{R}^2$
- Se pudermos criar uma reta que separe as duas classes, então dizemos que o conjunto é linearmente separável

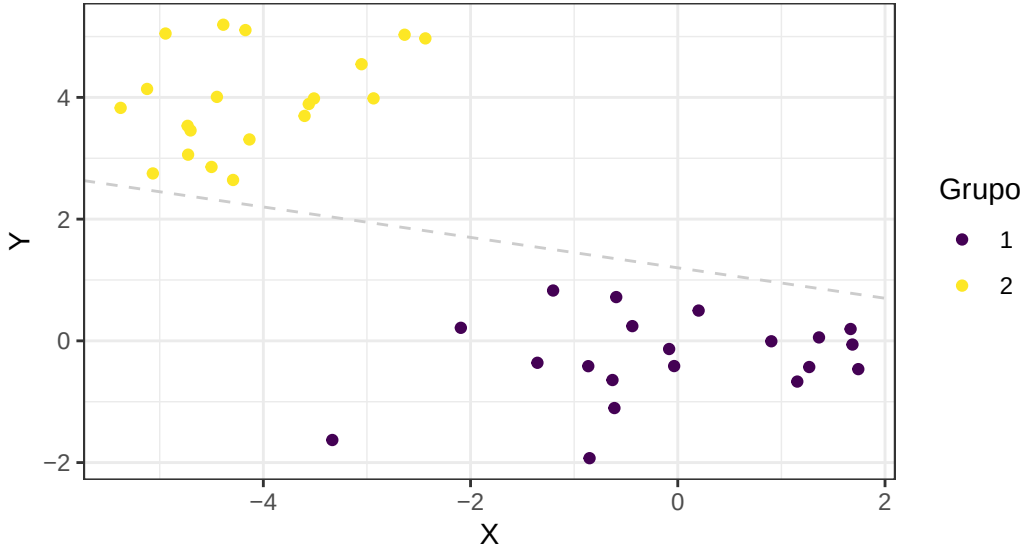
SVM



SVM



SVM



SVM

- Em um espaço de dimensão k, precisamos encontrar o subespaço de dimensão $k-1$ que separa as duas classes
- Ou seja, a ideia é utilizar hiperplanos para separar o conjunto de dados
- Um hiperplano é um conjunto L tal que

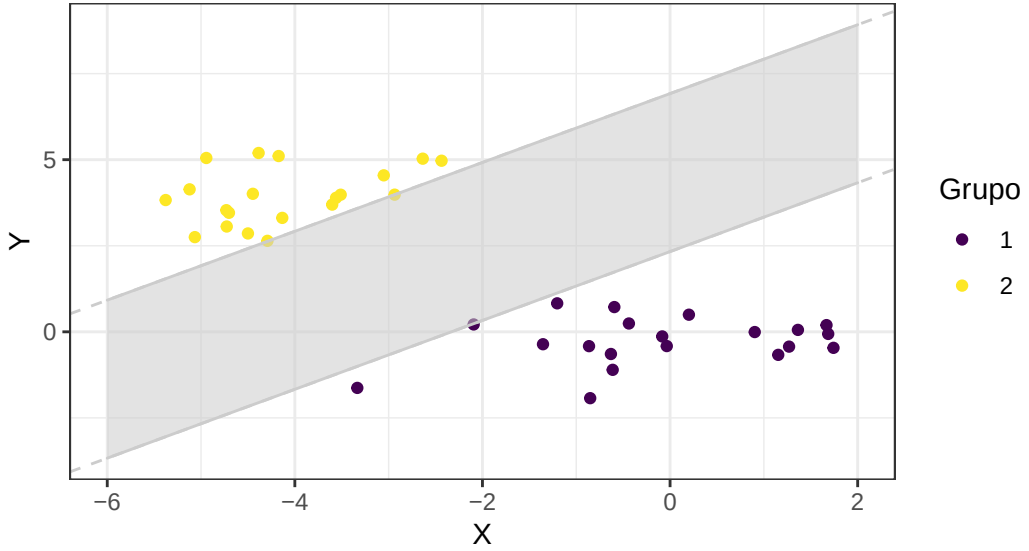
$$L = \{X : f(\boldsymbol{x}) = \boldsymbol{W}'\boldsymbol{x} - b = 0\}$$

- Se $k = 2$ (um plano), o hiperplano é uma reta ($k = 1$)
- O hiperplano que separa duas classes pode não ser único

SVM

- Encontre dois hiperplanos que separem as duas classes
- Maximize a separação mínima entre os dois hiperplanos
- A distância entre os hiperplanos é chamada de margem
- Maximizar a margem minimiza o risco de erro de classificação

SVM



SVM

- Como um hiperplano L pode ser expresso como o produto interno entre um vetor de coeficientes $\boldsymbol{w}$ e um vetor de características $\boldsymbol{x}$, temos

$$\boldsymbol{w}'\boldsymbol{x} = 0$$

- Assim, a distância deste hiperplano a um vetor $\boldsymbol{x}$ é dada por

$$\frac{\boldsymbol{w}'\boldsymbol{x}}{\|\boldsymbol{w}\|}$$

em que $\|\boldsymbol{w}\|$ é o comprimento do vetor $\boldsymbol{w}$

SVM

- Vamos assumir que possuímos um conjunto de treinamento $(\boldsymbol{x}_i, y_i)$ com n observações
- A classificação é determinada pelo resultado de $\boldsymbol{w}'\boldsymbol{x} + b$
- Se $\boldsymbol{w}'\boldsymbol{x} + b > 0$, então $\boldsymbol{x}$ pertence à classe 1
- Caso contrário, $\boldsymbol{x}$ pertence à classe 2

SVM

- Se um ponto é classificado corretamente pelo hiperplano, então $y_i(\boldsymbol{w}'\boldsymbol{x} + b) > 0$
- A distância de um ponto classificado corretamente para o hiperplano é dada por

$$\frac{y_i(\boldsymbol{w}'\boldsymbol{x} + b)}{\|\boldsymbol{w}\|}$$

SVM

- Precisamos encontrar o vetor $\boldsymbol{w}$ e o coeficiente b tais que a margem é maximizada para todos os pontos do conjunto de treinamento, ou seja,

$$\arg\max_{w,b} \left\{ \frac{1}{\|\boldsymbol{w}\|} \min_i [y_i(\boldsymbol{w}'\boldsymbol{x} + b)] \right\}$$

- Este é um problema de otimização difícil, pois envolve maximizar uma minimização
- Felizmente, é possível simplificá-lo

SVM

- Aplique uma transformação $\phi(\cdot)$ nos dados tal que

$$y_i(\mathbf{w}'\phi(\mathbf{x}_i) + b) = 1$$

- Todos os pontos classificados corretamente vão satisfazer

$$y_i(\mathbf{w}'\phi(\mathbf{x}_i) + b) \geq 1$$

- Esta é a chamada representação canônica do hiperplano de decisão

SVM

- Agora as margens se tornam

$$\frac{y_i(\boldsymbol{w}'\phi(\boldsymbol{x}_i) + b)}{\|\boldsymbol{w}\|} = \frac{1}{\|\boldsymbol{w}\|}$$

- O objetivo agora se torna encontrar $\boldsymbol{w}$ e b tais que a margem é maximizada sujeita ao conjunto de treinamento utilizado no ajuste do modelo
- Assim, maximizar a margem é equivalente a minimizar a norma de $\boldsymbol{w}$

SVM

- Encontre $\boldsymbol{w}$ tal que

$$\min \frac{1}{2}\|\boldsymbol{w}\|^2$$

sujeito a

$$y_i(\boldsymbol{w}'\phi(\boldsymbol{x}_i) + b) \geq 1$$

SVM

- Se duas classes são linearmente separáveis, o algoritmo converge para a solução em um número finito de passos
- Entretanto, existem alguns problemas com este algoritmo:
  - Quando os dados são linearmente separáveis, há muitas soluções, que são dependentes das condições iniciais
  - O número de passos pode ser muito grande
  - Quando os dados não são linearmente separáveis, o algoritmo pode não convergir

SVM

- Se duas classes são linearmente separáveis, então
  - O hiperplano de separação ótimo separa os dados em duas classes e maximiza a distância ao ponto mais próximo em cada classe
  - Os vetores suporte são os pontos nas margens do hiperplano
  - A solução é única e funciona muito bem na prática

Kernel

Kernel

- Kernel é a função utilizada para separar dados que não sejam linearmente separáveis
- O kernel aplica uma transformação nos dados originais, levando um espaço euclideano para um espaço não-linear

Kernel

- Sejam $\boldsymbol{x}_1$ e $\boldsymbol{x}_2$ vetores no espaço $\mathbb{R}^k$ original e sejam α_i e b escalares
- Queremos encontrar a função

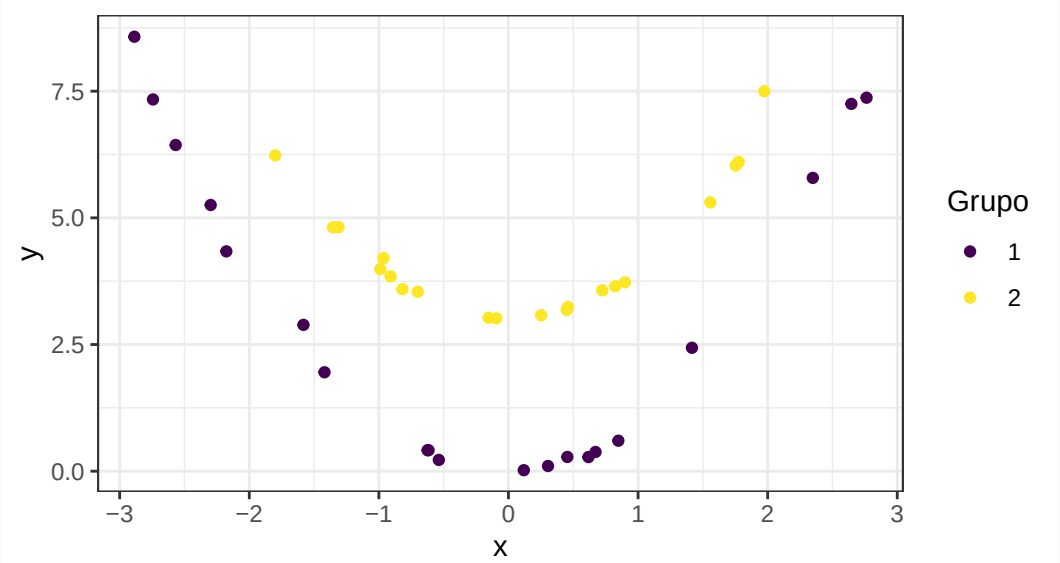
$$f(\boldsymbol{x}) = b + \sum_{i=1}^{n} \alpha_i K(\boldsymbol{x}_1, \boldsymbol{x}_2)$$

- A função $K(\cdot, \cdot)$ é chamada de kernel

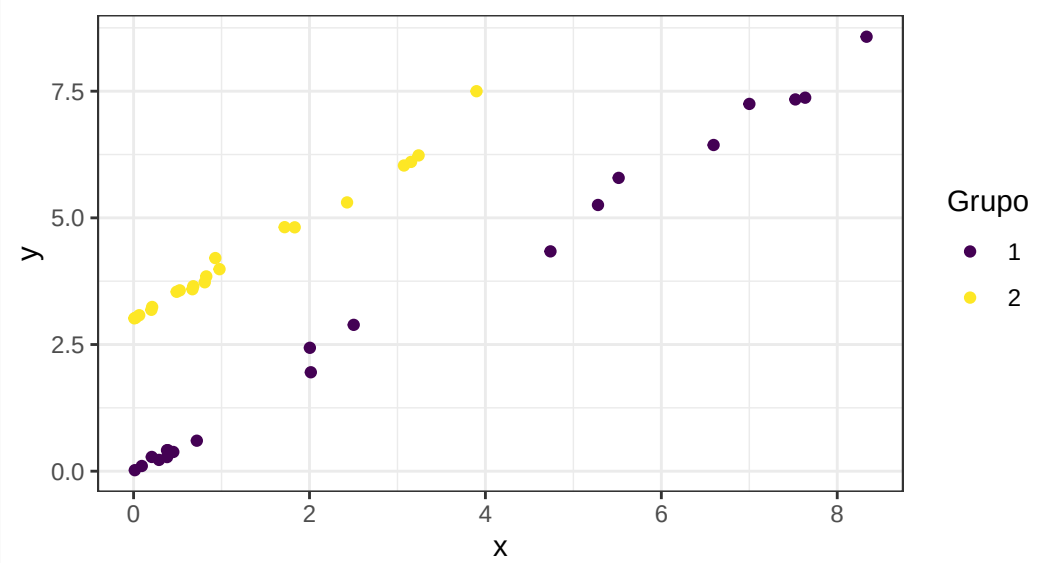
Kernel

- O pacote `tidymodels` consegue trabalhar com kernels de outros pacotes, como o `kernlab`, a fim de realizar estas transformações
- Sejam x_1 e x_2 vetores de $\mathbb{R}^k$, γ e σ parâmetros de escala, c uma constante e d um inteiro positivo. Assim, os kernels mais comuns são
  - Linear: $x_1' \cdot x_2$
  - Polinomial: $(x_1' \cdot x_2 + c)^d$
  - Radial: $\exp(-\sigma \|x_1 - x_2\|^2)$
  - Sigmoide: $\tanh(\gamma x_1' \cdot x_2 + c)$, em que $\tanh(x) = \dfrac{e^x - e^{-x}}{e^x + e^{-x}}$

Kernel



Kernel



Kernel

- Não estamos limitados a apenas estes tipos de kernel
- Entretanto, os kernels linear e radial são os mais utilizados na prática

Exemplo

Exemplo

- Vamos analisar o conjunto de dados `Ionosphere`, do pacote `mlbench`
- São dados provenientes de medidas realizadas com radares, a fim de detectar estruturas na ionosfera

Exemplo

```
> # pacotes
>
> library(tidymodels)
> theme_set(theme_bw())
> library(ggfortify)
> library(onehot)
> library(janitor)
>
> # dados
>
> data("Ionosphere", package = "mlbench")
```

Exemplo

```
> # criacao de variaveis dummy
>
> ionosphere <-
+   Ionosphere %>%
+   select(!where(is.numeric)) %>%
+   select(-V2, -Class) %>%
+   onehot() %>%
+   predict(Ionosphere) %>%
+   as.data.frame() %>%
+   select(V1_0 = `V1=0`, V1_1 = `V1=1`)
```

Exemplo

```
> ionosphere <-
+   Ionosphere %>%
+   select(-V1, -V2) %>%
+   bind_cols(ionosphere) %>%
+   relocate(Class)
```

Exemplo

```
> # treino/teste
>
> ionosphere %>%
+   group_by(Class) %>%
+   count()

## # A tibble: 2 x 2
## # Groups:   Class [2]
##   Class     n
##   <fct> <int>
## 1 bad     126
## 2 good    225
```

Exemplo

```
> # semente aleatoria
>
> set.seed(5659)
>
> # 70% dos dados como treino
>
> ionosphere_split <- initial_split(ionosphere,
+         prop = .70,
+         strata = Class)
```

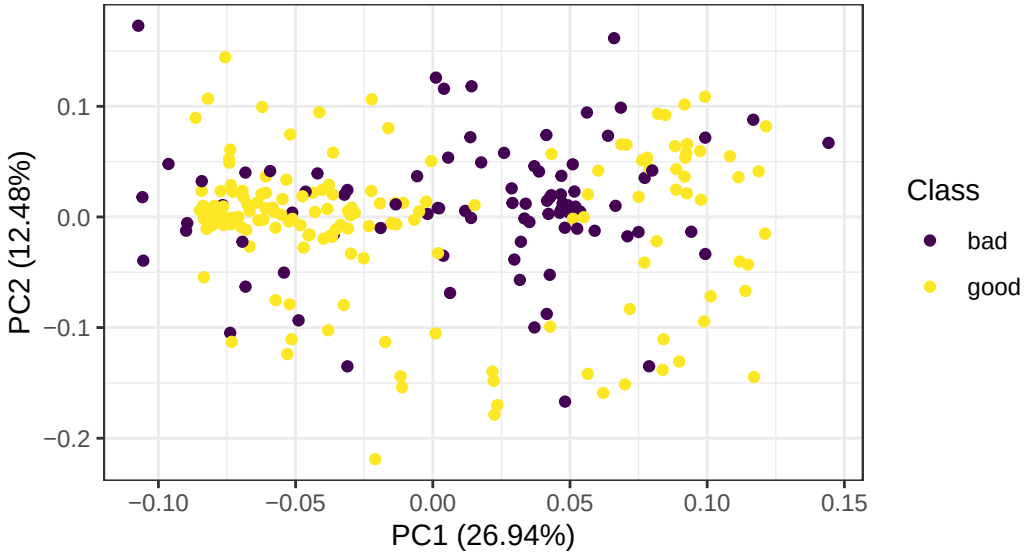
Exemplo

```
> # criar os conjuntos de dados de treino e teste
>
> ionosphere_treino <- training(ionosphere_split)
> nrow(ionosphere_treino)/nrow(ionosphere)

## [1] 0.7037037

> ionosphere_teste  <- testing(ionosphere_split)
> nrow(ionosphere_teste)/nrow(ionosphere)

## [1] 0.2962963
```

Exemplo

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

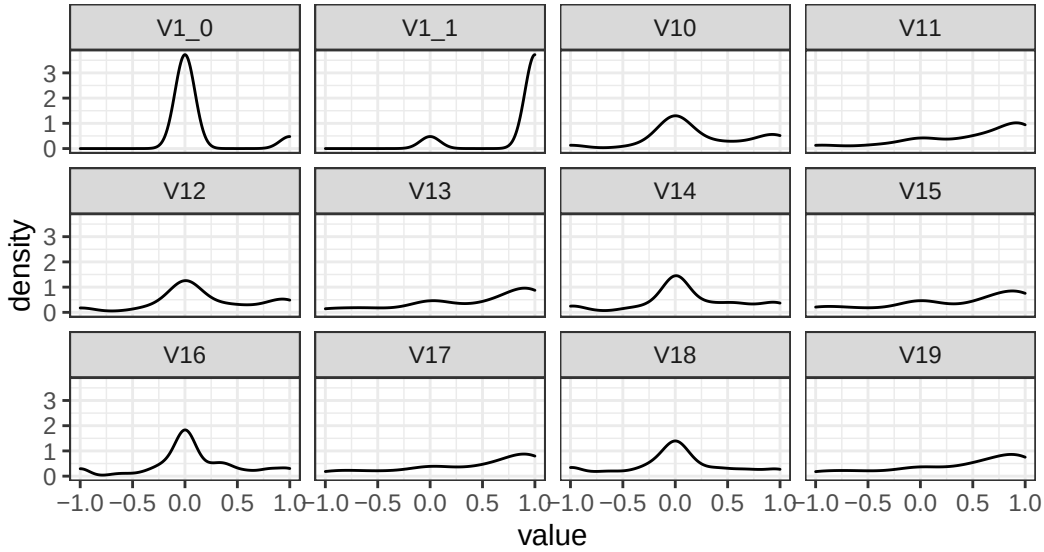
Exemplo


0.1
0.0
−0.1
−0.2
PC2 (12.48%)
−0.10
−0.05
0.00
0.05
0.10
0.15
PC1 (26.94%)
Class
bad
good

Exemplo

```
> ionosphere_treino %>%
+   pivot_longer(-Class) %>%
+   ggplot(aes(x = value)) +
+   geom_density() +
+   facet_wrap(~ name)
```

Exemplo


V1_0
V1_1
V10
V11
V12
V13
V14
V15
V16
V17
V18
V19
density
value
0
1
2
3
−1.0
−0.5
0.0
0.5
1.0

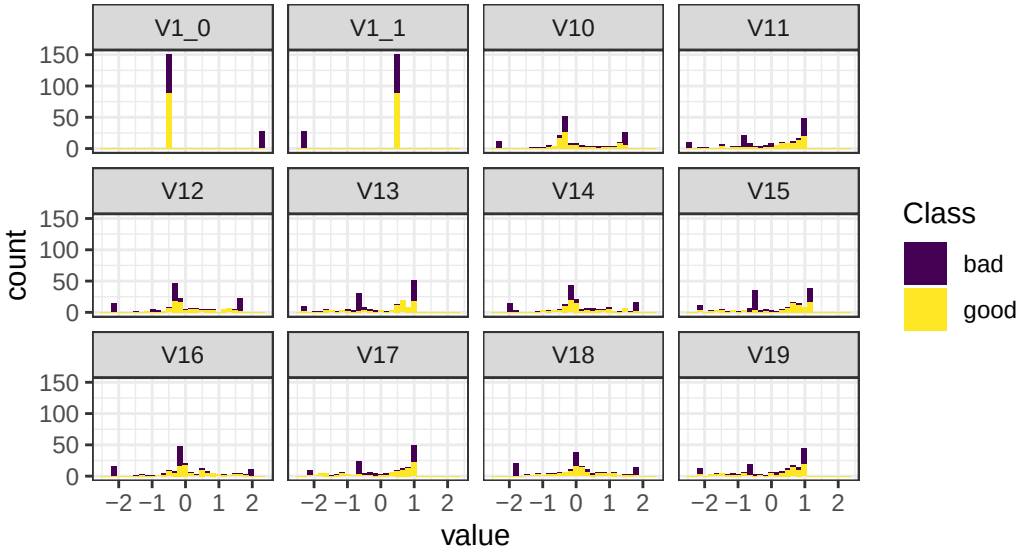
Exemplo

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

Exemplo

```
> # aplicar a transformacao aos dados
>
> ionosphere_treino_t <- juice(ionosphere_rec)
>
> # verificar o resultado do processamento de dados
>
> ionosphere_treino_t %>%
+   pivot_longer(-Class) %>%
+   filter(str_detect(name, "V1")) %>%
+   ggplot(., aes(fill = Class)) +
+   geom_histogram(aes(value)) +
+   facet_wrap(~ name) +
+   scale_fill_viridis_d()
```

Exemplo


V1_0
V1_1
V10
V11
V12
V13
V14
V15
V16
V17
V18
V19
count
150
100
50
0
−2 −1 0 1 2
value
Class
bad
good

Exemplo

```
> # preparar o conjunto de teste
>
> ionosphere_teste_t <- bake(ionosphere_rec,
+                            new_data = ionosphere_teste)
```

Exemplo

```
> ########################################
> # definicao do tuning para svm polinomial
>
> ionosphere_svm_poly_tune <-
+   svm_poly(
+     cost = tune(),
+     degree = tune()) %>%
+   set_engine("kernlab") %>%
+   set_mode("classification")
```

Exemplo

```
> # grid de procura
>
> ionosphere_svm_poly_grid <- grid_regular(cost(range(-10, 5)),
+                                          degree(),
+                                          levels = c(10, 2))
```

Exemplo

```
> # workflow
>
> ionosphere_svm_poly_tune_wflow <-
+   workflow() %>%
+   add_model(ionosphere_svm_poly_tune) %>%
+   add_formula(Class ~ .)
```

Exemplo

```
> # definicao da validacao cruzada
>
> set.seed(6472)
>
> ionosphere_treino_cv <- vfold_cv(ionosphere_treino_t, v = 10)
```

Exemplo

```
> # avaliacao do modelo
>
> ionosphere_svm_poly_fit_tune <-
+   ionosphere_svm_poly_tune_wflow %>%
+   tune_grid(
+     resamples = ionosphere_treino_cv,
+     grid = ionosphere_svm_poly_grid
+   )
```

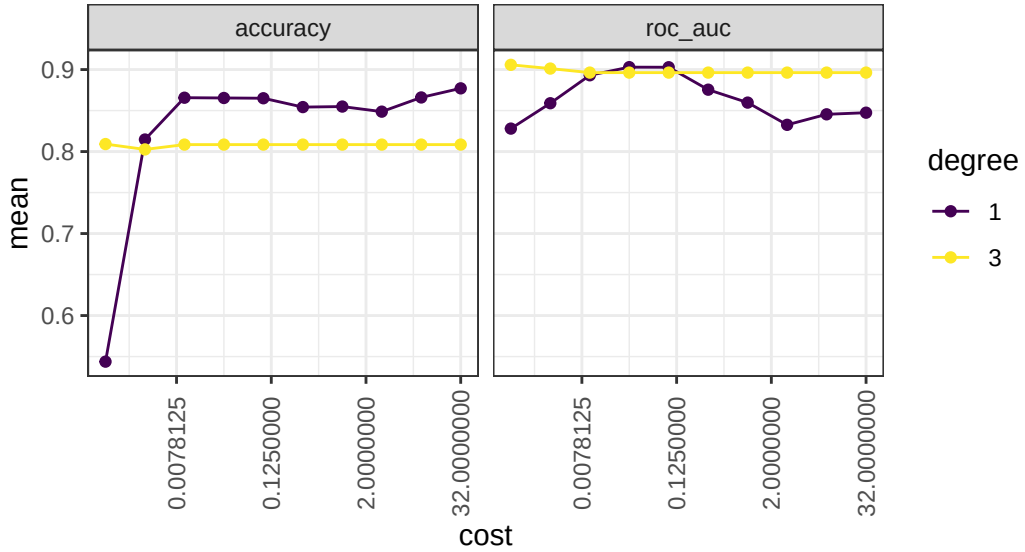
Exemplo

```
> # resultados
>
> collect_metrics(ionosphere_svm_poly_fit_tune)

## # A tibble: 40 x 8
##        cost degree .metric  .estimator  mean     n std_err
##       <dbl>  <dbl> <chr>    <chr>      <dbl> <int>   <dbl>
##  1 0.000977      1 accuracy binary     0.544    10  0.0506
##  2 0.000977      1 roc_auc  binary     0.828    10  0.0368
##  3 0.00310       1 accuracy binary     0.815    10  0.0187
##  4 0.00310       1 roc_auc  binary     0.859    10  0.0385
##  5 0.00984       1 accuracy binary     0.866    10  0.0167
##  6 0.00984       1 roc_auc  binary     0.893    10  0.0359
##  7 0.0312        1 accuracy binary     0.865    10  0.0170
##  8 0.0312        1 roc_auc  binary     0.903    10  0.0319
```

Exemplo

```
> ionosphere_svm_poly_fit_tune %>%
+   collect_metrics() %>%
+   mutate(degree = factor(degree)) %>%
+   ggplot(., aes(x = cost, y = mean, colour = degree)) +
+   geom_line() +
+   geom_point() +
+   facet_grid(~ .metric) +
+   scale_x_continuous(trans = "log2") +
+   theme(axis.text.x = element_text(angle = 90, vjust = 0.5)) +
+   scale_colour_viridis_d()
```

Exemplo


accuracy
roc_auc
mean
0.9
0.8
0.7
0.6
0.0078125
0.1250000
2.0000000
32.0000000
cost
degree
1
3

Exemplo

```
> # melhores modelos
>
> ionosphere_svm_poly_fit_tune %>%
+   show_best("accuracy")

## # A tibble: 5 x 8
##       cost degree .metric  .estimator  mean     n std_err
##      <dbl>  <dbl> <chr>    <chr>      <dbl> <int>   <dbl>
## 1 32            1 accuracy binary     0.877    10  0.0229
## 2 10.1          1 accuracy binary     0.866    10  0.0235
## 3  0.00984      1 accuracy binary     0.866    10  0.0167
## 4  0.0312       1 accuracy binary     0.865    10  0.0170
## 5  0.0992       1 accuracy binary     0.865    10  0.0125
## # ... with 1 more variable: .config <chr>
```

Exemplo

```
> ionosphere_svm_poly_fit_tune %>%
+   show_best("roc_auc")

## # A tibble: 5 x 8
##       cost degree .metric .estimator  mean     n std_err
##      <dbl>  <dbl> <chr>   <chr>      <dbl> <int>   <dbl>
## 1 0.000977      3 roc_auc binary     0.906    10  0.0155
## 2 0.0312        1 roc_auc binary     0.903    10  0.0319
## 3 0.0992        1 roc_auc binary     0.903    10  0.0307
## 4 0.00310       3 roc_auc binary     0.901    10  0.0232
## 5 0.00984       3 roc_auc binary     0.896    10  0.0267
## # ... with 1 more variable: .config <chr>
```

Exemplo

```
> # melhor modelo
>
> ionosphere_svm_poly_best <-
+   ionosphere_svm_poly_fit_tune %>%
+   select_best("accuracy")
>
> ionosphere_svm_poly_final <-
+   ionosphere_svm_poly_tune_wflow %>%
+   finalize_workflow(ionosphere_svm_poly_best)
>
> ionosphere_svm_poly_final <- fit(ionosphere_svm_poly_final,
+                                  ionosphere_treino_t)
```

Exemplo

```
> ionosphere_svm_poly_final

## == Workflow [trained] ==========================================
## Preprocessor: Formula
## Model: svm_poly()
##
## -- Preprocessor ----------------------------------------------
## Class ~ .
##
## -- Model -----------------------------------------------------
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 32
##
```

Exemplo

```
> # resultados no conjunto de teste
>
> resultado_poly <-
+   ionosphere_teste_t %>%
+   bind_cols(predict(ionosphere_svm_poly_final, ionosphere_teste_t) %>%
+               rename(predicao_svm_poly = .pred_class))
```

Exemplo

```
> metrics(resultado_poly,
+         truth = Class,
+         estimate = predicao_svm_poly,
+         options = "roc")

## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.798
## 2 kap      binary         0.551
```

Exemplo

```
> ######################################
> # definicao do tuning para svm radial
>
> ionosphere_svm_rbf_tune <-
+   svm_rbf(
+     cost = tune(),
+     rbf_sigma = tune()) %>%
+   set_engine("kernlab") %>%
+   set_mode("classification")
```

Exemplo

```
> # grid de procura
>
> ionosphere_svm_rbf_grid <- grid_regular(cost(range(-10, 5)),
+                                         rbf_sigma(),
+                                         levels = c(10, 5))
```

Exemplo

```
> # workflow
>
> ionosphere_svm_rbf_tune_wflow <-
+   workflow() %>%
+   add_model(ionosphere_svm_rbf_tune) %>%
+   add_formula(Class ~ .)
```

Exemplo

```
> # avaliacao do modelo
>
> ionosphere_svm_rbf_fit_tune <-
+   ionosphere_svm_rbf_tune_wflow %>%
+   tune_grid(
+     resamples = ionosphere_treino_cv,
+     grid = ionosphere_svm_rbf_grid
+   )
```

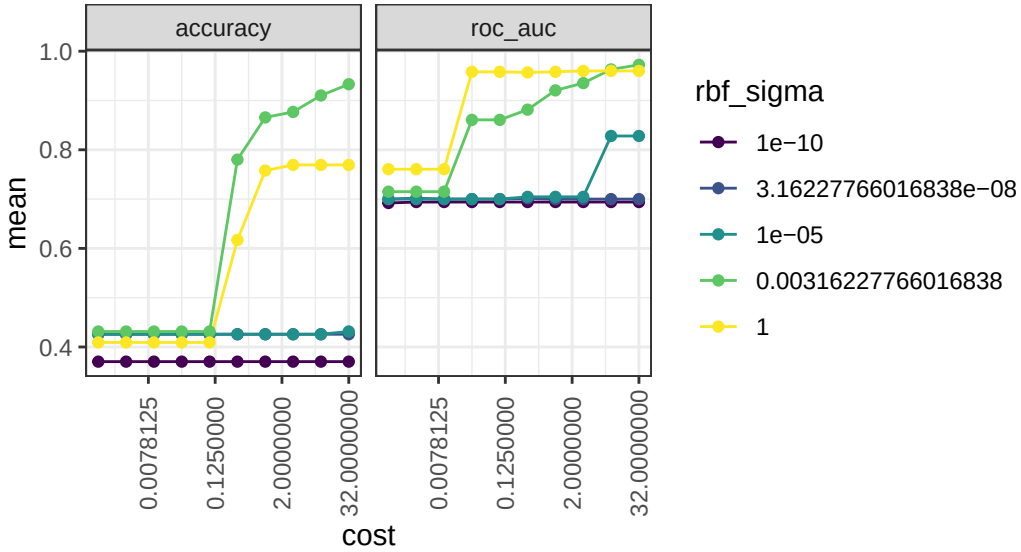
Exemplo

```
> # resultados
>
> collect_metrics(ionosphere_svm_rbf_fit_tune)

## # A tibble: 100 x 8
##        cost rbf_sigma .metric .estimator  mean     n std_err
##       <dbl>     <dbl> <chr>   <chr>      <dbl> <int>   <dbl>
##  1 0.000977  1.00e-10 accura~ binary     0.370    10  0.0316
##  2 0.000977  1.00e-10 roc_auc binary     0.692    10  0.0916
##  3 0.00310   1.00e-10 accura~ binary     0.370    10  0.0316
##  4 0.00310   1.00e-10 roc_auc binary     0.694    10  0.0917
##  5 0.00984   1.00e-10 accura~ binary     0.370    10  0.0316
##  6 0.00984   1.00e-10 roc_auc binary     0.694    10  0.0918
##  7 0.0312    1.00e-10 accura~ binary     0.370    10  0.0316
##  8 0.0312    1.00e-10 roc_auc binary     0.694    10  0.0918
```

Exemplo

```
> ionosphere_svm_rbf_fit_tune %>%
+   collect_metrics() %>%
+   mutate(rbf_sigma = factor(rbf_sigma)) %>%
+   ggplot(., aes(x = cost, y = mean, colour = rbf_sigma)) +
+   geom_line() +
+   geom_point() +
+   facet_grid(~ .metric) +
+   scale_x_continuous(trans = "log2") +
+   theme(axis.text.x = element_text(angle = 90, vjust = 0.5)) +
+   scale_colour_viridis_d()
```

Exemplo


accuracy
roc_auc
mean
1.0
0.8
0.6
0.4
0.0078125
0.1250000
2.0000000
32.0000000
cost
rbf_sigma
1e−10
3.16227766016838e−08
1e−05
0.00316227766016838
1

Exemplo

```
> # melhores modelos
>
> ionosphere_svm_rbf_fit_tune %>%
+   show_best("accuracy")

## # A tibble: 5 x 8
##     cost rbf_sigma .metric  .estimator  mean     n std_err
##    <dbl>     <dbl> <chr>    <chr>      <dbl> <int>   <dbl>
## 1 32       0.00316 accuracy binary     0.933    10  0.0216
## 2 10.1     0.00316 accuracy binary     0.910    10  0.0189
## 3  3.17    0.00316 accuracy binary     0.877    10  0.0160
## 4  1       0.00316 accuracy binary     0.866    10  0.0187
## 5  0.315   0.00316 accuracy binary     0.780    10  0.0295
## # ... with 1 more variable: .config <chr>
```

Exemplo

```
> ionosphere_svm_rbf_fit_tune %>%
+   show_best("roc_auc")

## # A tibble: 5 x 8
##    cost rbf_sigma .metric .estimator  mean     n std_err
##   <dbl>     <dbl> <chr>   <chr>      <dbl> <int>   <dbl>
## 1 32      0.00316 roc_auc binary     0.972    10  0.0109
## 2 10.1    0.00316 roc_auc binary     0.963    10  0.0125
## 3  3.17   1       roc_auc binary     0.960    10  0.0208
## 4 10.1    1       roc_auc binary     0.960    10  0.0208
## 5 32      1       roc_auc binary     0.960    10  0.0208
## # ... with 1 more variable: .config <chr>
```

Exemplo

```
> # melhor modelo
>
> ionosphere_svm_rbf_best <-
+   ionosphere_svm_rbf_fit_tune %>%
+   select_best("accuracy")
>
> ionosphere_svm_rbf_final <-
+   ionosphere_svm_rbf_tune_wflow %>%
+   finalize_workflow(ionosphere_svm_rbf_best)
>
> ionosphere_svm_rbf_final <- fit(ionosphere_svm_rbf_final,
+                                 ionosphere_treino_t)
```

Exemplo

```
> ionosphere_svm_rbf_final

## == Workflow [trained] ==========================================
## Preprocessor: Formula
## Model: svm_rbf()
##
## -- Preprocessor ----------------------------------------------
## Class ~ .
##
## -- Model -----------------------------------------------------
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 32
##
```

Exemplo

```
> # resultados no conjunto de teste
>
> resultado_rbf <-
+   ionosphere_teste_t %>%
+   bind_cols(predict(ionosphere_svm_rbf_final, ionosphere_teste_t) %>%
+               rename(predicao_svm_rbf = .pred_class))
```

Exemplo

```
> metrics(resultado_rbf,
+         truth = Class,
+         estimate = predicao_svm_rbf,
+         options = "roc")

## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.904
## 2 kap      binary         0.785
```

Exemplo

```
> # comparacao polinomial e radial
>
> metrics(resultado_poly,
+         truth = Class,
+         estimate = predicao_svm_poly,
+         options = "roc")

## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.798
## 2 kap      binary         0.551
```

Exemplo

```
> metrics(resultado_rbf,
+         truth = Class,
+         estimate = predicao_svm_rbf,
+         options = "roc")

## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.904
## 2 kap      binary         0.785
```

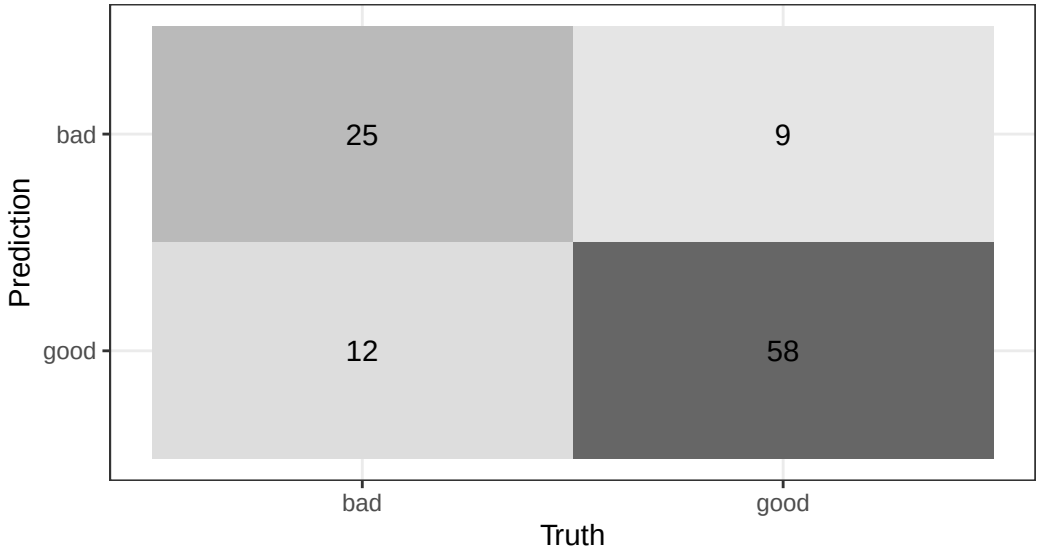
Exemplo

```
> conf_mat(resultado_poly,
+          truth = Class,
+          estimate = predicao_svm_poly)

##            Truth
## Prediction bad good
##       bad   25    9
##       good  12   58
```

Exemplo

```
> conf_mat(resultado_poly,
+          truth = Class,
+          estimate = predicao_svm_poly) %>%
+   autoplot(type = "heatmap")
```

Exemplo



Exemplo

```
> sens(resultado_poly,
+      truth = Class,
+      estimate = predicao_svm_poly)

## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 sens    binary         0.676
```

Exemplo

```
> spec(resultado_poly,
+      truth = Class,
+      estimate = predicao_svm_poly)

## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 spec    binary         0.866
```

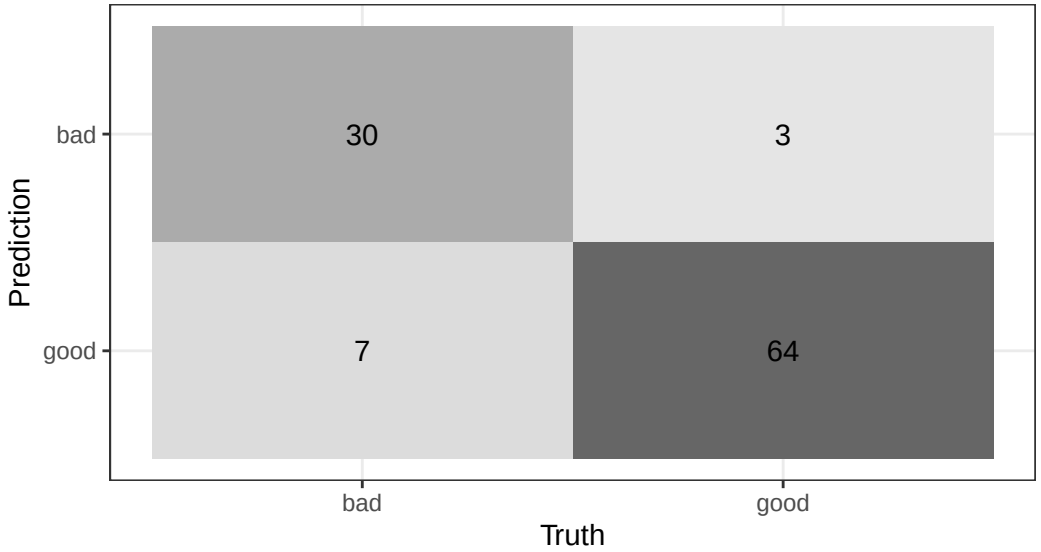
Exemplo

```
> conf_mat(resultado_rbf,
+          truth = Class,
+          estimate = predicao_svm_rbf)

##           Truth
## Prediction bad good
##       bad   30    3
##       good   7   64
```

Exemplo

```
> conf_mat(resultado_rbf,
+          truth = Class,
+          estimate = predicao_svm_rbf) %>%
+   autoplot(type = "heatmap")
```

Exemplo



Exemplo

```
> sens(resultado_rbf,
+      truth = Class,
+      estimate = predicao_svm_rbf)

## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 sens    binary         0.811
```

Exemplo

```
> spec(resultado_rbf,
+      truth = Class,
+      estimate = predicao_svm_rbf)

## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 spec    binary         0.955
```

Exercícios

Exercícios

1. O arquivo `credito.csv` contém registros de transações com cartões de crédito classificadas como legítimas ou fraudes. São 31 variáveis:
   - `Time`: tempo (em segundos) decorrido entre a transação e o início da coleta dos dados
   - `V1-V28`: registros anonimizados sobre as transações
   - `Amount`: valor da transação
   - `Class`: classificação da transação como legítima (`true`) ou fraude (`fraud`)

   Estamos interessados em identificar uma transação como legítima ou como fraude. Para isso, importe esse conjunto de dados no R.

Exercícios

2. Faça uma análise exploratória nos dados, identificando possíveis variáveis problemáticas e tratando-as corretamente.
3. Encontre o modelo de máquina de vetor suporte que melhor classifica a variável resposta, utilizando pelo menos dois kernels diferentes.

Exercícios

4. O conjunto de dados `Sacramento`, dentro do pacote `caret`, possui dados a respeito de 932 residências da região metropolitana de Sacramento, capital da Califórnia. São 8 variáveis preditoras:
   - `city`: cidade da região metropolitana
   - `zip`: CEP da residência
   - `beds`: número de quartos
   - `baths`: número de banheiros
   - `sqft`: área da casa, em pés quadrados
   - `type`: tipo da residência, podendo ser Condo (casa em um condomínio), Multi_Family (apartamento em um prédio ou condomínio) ou Residential (casa)
   - `latitude`: latitude da residência
   - `longitude`: longitude da residência

Exercícios

4. (Cont.) Carregue este conjunto de dados na memória do R através do comando `data(Sacramento, package = "caret")`. Crie um novo objeto chamado `residencias`, apenas com as colunas `price`, `type`, `sqft`, `baths` e `beds`. Considere `price` como variável resposta.
5. Crie boxplots comparando o preço dos imóveis de acordo com o tipo de residência. Parece que há alguma influência do tipo de imóvel em seu preço?
6. Utilize a função `GGally::ggpairs` para verificar as correlações entre as variáveis quantitativas deste conjunto de dados. Argumente se há correlação entre as variáveis preditoras.

Exercícios

7. Com o mesmo gráfico da questão anterior, avalie a relação de `price` versus cada uma das variáveis quantitativas. Que tipo(s) de relação(ões) podemos perceber?
8. Queremos prever a variável `price` destes imóveis. Para isto, ajuste dois modelos SVM a esses dados, utilizando os kernels linear e radial.
9. Encontre o modelo melhor ajustado. Justifique a sua escolha.

Máquinas de Vetor Suporte

EST0133 - Introdução à Modelagem de Big Data

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

Universidade Federal do Rio Grande do Norte