

# Suport Vector Regression Machine - SVRM

Comparando com outros modelos

Prof. Dr. Pedro Rafael D. Marinho

2023-07-20

## Carregando bibliotecas necessárias

```
library(tidymodels)
```

```
-- Attaching packages ----- tidymodels 1.1.0 --
```

v broom	1.0.5	v recipes	1.0.6
v dials	1.2.0	v rsample	1.1.1
v dplyr	1.1.2	v tibble	3.2.1
v ggplot2	3.4.2	v tidyr	1.3.0
v infer	1.0.4	v tune	1.1.1
v modeldata	1.1.0	v workflows	1.1.3
v parsnip	1.1.0	v workflowsets	1.0.1
v purrr	1.0.1	v yardstick	1.2.0

```
-- Conflicts ----- tidymodels_conflicts() --
```

```
x purrr::discard() masks scales::discard()
x dplyr::filter() masks stats::filter()
x dplyr::lag() masks stats::lag()
x recipes::step() masks stats::step()
* Use suppressPackageStartupMessages() to eliminate package startup messages
```

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ----- tidyverse 2.0.0 --
v forcats 1.0.0      v readr 2.1.4
v lubridate 1.9.2    v stringr 1.5.0
```

```
-- Conflicts ----- tidyverse_conflicts() --
```

```
x readr::col_factor() masks scales::col_factor()
```

```
x purrr::discard()    masks scales::discard()
```

```
x dplyr::filter()     masks stats::filter()
```

```
x stringr::fixed()    masks recipes::fixed()
```

```
x dplyr::lag()        masks stats::lag()
```

```
x readr::spec()       masks yardstick::spec()
```

```
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become
```

```
library(GGally)
```

Registered S3 method overwritten by 'GGally':

```
method from
```

```
+.gg    ggplot2
```

```
library(skimr)
```

```
# Resolvendo possíveis conflitos entre o tidymodels e outras bibliotecas
```

```
tidymodels::tidymodels_prefer()
```

## Importando a base de dados

Utilizando os dados de vinho vermelho , disponíveis [aqui](#), faça uma pequena análise exploratória dos dados. No [link](#) do Kaggle você consegue uma explicação sobre o que significa cada uma das variáveis.

Os dados, no meu caso, estão no diretório `"../dados/winequality-red.csv"`.

Você deverá alterar o *path* para o diretório encontra-se a base que deverá ser obtida no link acima.

```
dados <- readr::read_csv(file = "../dados/winequality-red.csv")
```

```

Rows: 1599 Columns: 12
-- Column specification -----
Delimiter: ","
dbl (12): fixed acidity, volatile acidity, citric acid, residual sugar, chlo...

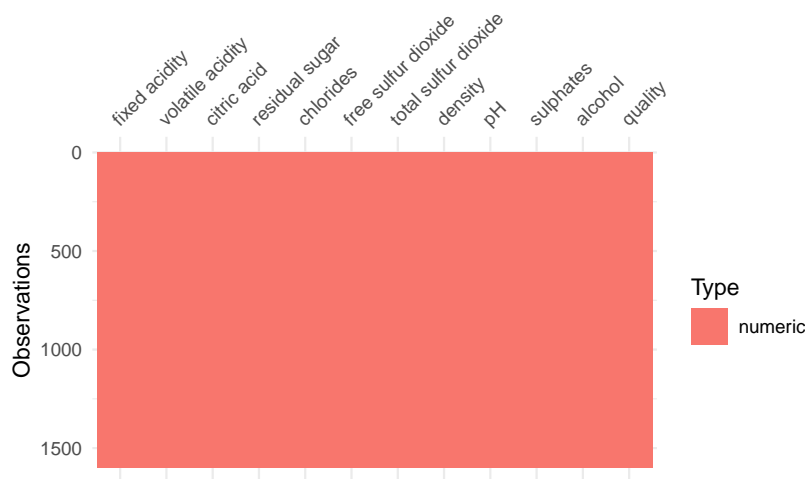
i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.

```

## Uma exploração rápida dos dados

É sempre importante olhar os dados antes de tentar modelar. Uma análise exploratória sempre será útil para identificarmos possíveis inconsistências.

```
visdat::vis_dat(dados)
```



O gráfico acima mostra que temos uma base de dados sem informações faltantes e todas as *features* presentes na base são numéricas. É uma situação confortável, haja vista que, aqui, não precisaremos nos preocupar com imputação de dados faltantes.

Um resumo dos dados poderá ser obtido utilizando a função `glimpse` do pacote **dplyr** que é carregado com a biblioteca **tidyverse** de R.

```
dados |>
  dplyr::glimpse()
```

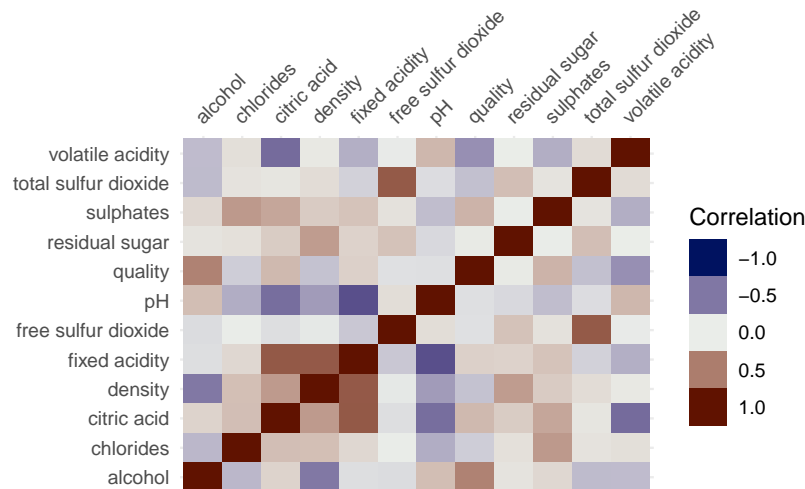
```
Rows: 1,599
Columns: 12
$ `fixed acidity`      <dbl> 7.4, 7.8, 7.8, 11.2, 7.4, 7.4, 7.9, 7.3, 7.8, 7~
$ `volatile acidity`  <dbl> 0.700, 0.880, 0.760, 0.280, 0.700, 0.660, 0.600~
$ `citric acid`       <dbl> 0.00, 0.00, 0.04, 0.56, 0.00, 0.00, 0.06, 0.00,~
$ `residual sugar`    <dbl> 1.9, 2.6, 2.3, 1.9, 1.9, 1.8, 1.6, 1.2, 2.0, 6.~
$ chlorides           <dbl> 0.076, 0.098, 0.092, 0.075, 0.076, 0.075, 0.069~
$ `free sulfur dioxide` <dbl> 11, 25, 15, 17, 11, 13, 15, 15, 9, 17, 15, 17, ~
$ `total sulfur dioxide` <dbl> 34, 67, 54, 60, 34, 40, 59, 21, 18, 102, 65, 10~
$ density             <dbl> 0.9978, 0.9968, 0.9970, 0.9980, 0.9978, 0.9978,~
$ pH                  <dbl> 3.51, 3.20, 3.26, 3.16, 3.51, 3.51, 3.30, 3.39,~
$ sulphates           <dbl> 0.56, 0.68, 0.65, 0.58, 0.56, 0.56, 0.46, 0.47,~
$ alcohol             <dbl> 9.4, 9.8, 9.8, 9.8, 9.4, 9.4, 9.4, 10.0, 9.5, 1~
$ quality             <dbl> 5, 5, 5, 6, 5, 5, 5, 7, 7, 5, 5, 5, 5, 5, ~
```

É possível todas as correlações entre todas as variáveis da base, com a função `data_vis_cor`. Um gráfico útil com as correlações poderá ser obtido usando a função `vis_cor`, conforme abaixo:

```
visdat::data_vis_cor(dados)
```

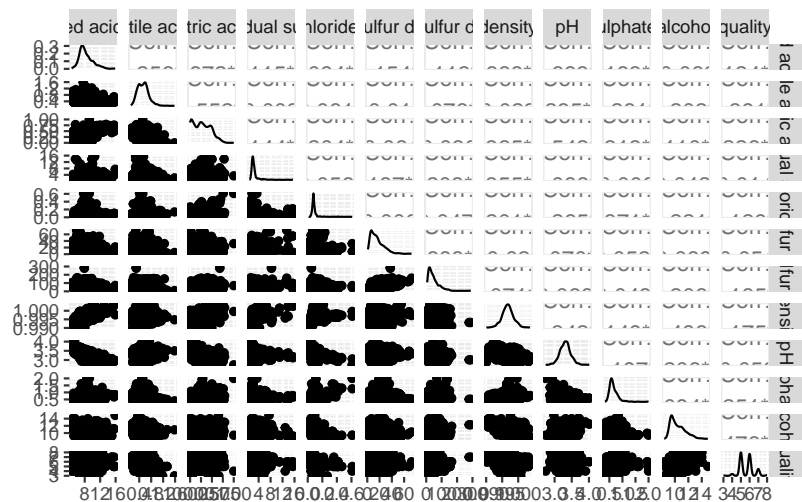
```
# A tibble: 144 x 3
  row_1      row_2      value
  <chr>      <chr>      <dbl>
1 fixed acidity fixed acidity      1
2 fixed acidity volatile acidity -0.256
3 fixed acidity citric acid      0.672
4 fixed acidity residual sugar    0.115
5 fixed acidity chlorides         0.0937
6 fixed acidity free sulfur dioxide -0.154
7 fixed acidity total sulfur dioxide -0.113
8 fixed acidity density          0.668
9 fixed acidity pH               -0.683
10 fixed acidity sulphates       0.183
# i 134 more rows
```

```
visdat::vis_cor(dados)
```



As bibliotecas **GGally** e **skimr** também possuem funções úteis que podem nos auxiliar no processo de exploração dos dados.

```
dados |>
  GGally::ggpairs()
```



```
dados |>
  skimr::skim()
```

Tabela 1: Data summary

Name	dados
Number of rows	1599
Number of columns	12
Column type frequency:	
numeric	12
Group variables	None

### Variable type: numeric

skim_variable	n_missing	n_nonmissing	mean	sd	p0	p25	p50	p75	p100	hist
fixed	0	1	8.32	1.74	4.60	7.10	7.90	9.20	15.90	
acidity										
volatile	0	1	0.53	0.18	0.12	0.39	0.52	0.64	1.58	
acidity										
citric	0	1	0.27	0.19	0.00	0.09	0.26	0.42	1.00	
acid										
residual	0	1	2.54	1.41	0.90	1.90	2.20	2.60	15.50	
sugar										
chlorides	0	1	0.09	0.05	0.01	0.07	0.08	0.09	0.61	
free	0	1	15.87	10.46	1.00	7.00	14.00	21.00	72.00	
sulfur										
dioxide										
total	0	1	46.47	32.90	6.00	22.00	38.00	62.00	89.00	
sulfur										
dioxide										
density	0	1	1.00	0.00	0.99	1.00	1.00	1.00	1.00	
pH	0	1	3.31	0.15	2.74	3.21	3.31	3.40	4.01	
sulphates	0	1	0.66	0.17	0.33	0.55	0.62	0.73	2.00	
alcohol	0	1	10.42	1.07	8.40	9.50	10.20	11.10	14.90	
quality	0	1	5.64	0.81	3.00	5.00	6.00	6.00	8.00	

## Partição dos dados

Como sabemos, precisamos dividir nossa base de dados em conjunto de treinamento, em que nesse conjunto iremos ser realizado o procedimento de validação cruzada e uma base de dados de teste para a avaliação final dos nossos modelos. Deixaremos 80% para o treinamento do modelo e 20% para teste. Iremos estratificar nossa amostra usando a variável *quality*, variável que queremos estimar (*label*/rótulo).

```
divisao <- rsample::initial_split(dados, prop = 0.8, strata = "quality")
treinamento <- rsample::training(divisao)
teste <- rsample::testing(divisao)
```