Regressao Carseats

Renato Camargo

26/05/2020

Descrição do Problema

Trabalharemos com a base Carseats do pacote ISRL, uma base de dados que simula as vendas de assentos de carro infantil para 400 lojas. Nosso objetivo será criar um modelo de regressão para predizer a variável 'Advertsiment', o orçamento para publicidade dedicado a cada local. Portanto, queremos responder qual o investimento habitual em publicidade de uma dada localidade baseado em suas características e estratégia de preço.

Importação do banco de dados

```
# Base de dados
library(ISLR)

# Análise exploratória
library(skimr)
library(GGally)

#Métodos de regressão
library(tidyverse)
library(tidymodels)
library(doParallel)
library(vip)
library(kernlab)
```

Análise descritiva

```
skim(Carseats)
```

Data summary

Name	Carseats
Number of rows	400
Number of columns	11

_____

Column type frequency:

factor	3
numeric	8

Group variables None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
ShelveLoc	0	1	FALSE	3	Med: 219, Bad: 96, Goo: 85
Urban	0	1	FALSE	2	Yes: 282, No: 118
US	0	1	FALSE	2	Yes: 258, No: 142

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
Sales	0	1	7.50	2.82	0	5.39	7.49	9.32	16.27	▁▆▃▁
CompPrice	0	1	124.97	15.33	77	115.00	125.00	135.00	175.00	▁▃▇▂▁
Income	0	1	68.66	27.99	21	42.75	69.00	91.00	120.00	▇▇▇▆
Advertising	0	1	6.63	6.65	0	0.00	5.00	12.00	29.00	▇▂▁▁
Population	0	1	264.84	147.38	10	139.00	272.00	398.50	509.00	▇▇▇▇
Price	0	1	115.80	23.68	24	100.00	117.00	131.00	191.00	▁▃▇▁
Age	0	1	53.32	16.20	25	39.75	54.50	66.00	80.00	▆▆▇▇
Education	0	1	13.90	2.62	10	12.00	14.00	16.00	18.00	▇▁▅▇

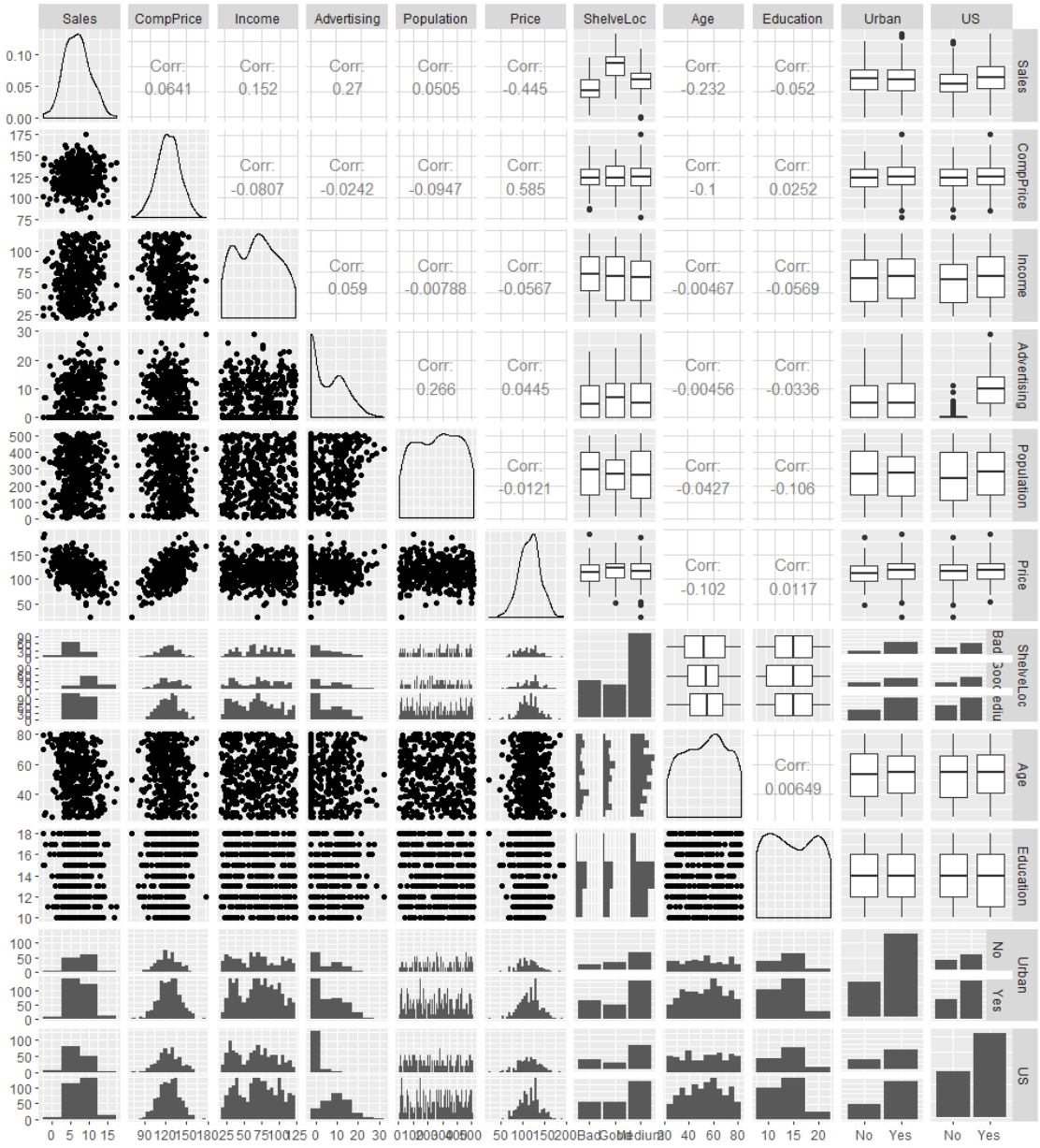
O banco de dados tem 400 observações e 11 variáveis (3 fatores e 11 numéricos) e nenhum dado faltante. Também vemos que as variáveis fatores possuem distribuição significativa entre todos os seus valores, de maneira que não será necessário fazer agrupamentos para a modelagem.

As onze variáveis são:

- Sales: Vendas por Unidades (em milhares) de cada localização
- CompPrice: preço praticado por competidor em cada localidade
- Income: Nível de renda da comunidade (milhares de dólares)
- Advertising: Orçamento local para publicidade (em milhares de dólares)
- Population: tamanho da população local (em milhares)
- Price: preço praticado por assento infantil
- Age: idade média da população local
- Education: nível educacional da localização
- ShelveLoc: fator (ruim, mediano, bom) que indica a qualidade da localização da prateleira do assento de carro infantil na prateleira.
- Urban: fator (sim, não) que indica se uma loja está em um local urbano ou não (ou seja, rural)
- US:fator (sim, não) que indica se a loja é nos EUA ou não.

Agora vamos fazer a análise relacionando as variáveis, usando o pacote GGalary:

```
GGally::ggpairs( Carseats, progress = FALSE,
                 upper = list(alignPercent=1),
                 lower = list(combo = wrap("facethist", binwidth = 5)))
```



- Da primeira tabela, vemos que ao menos 1 quarto das lojas não investem em publicidade. Com o gráfico de dispersão do GGPairs notamos que além desse pico em investimento zero, também há um segundo pico de concentração mais próximo ao centro. Possivelmente temos dois grupos que investem ou não em publicidade com comportamentos distintos.

- Não vemos correlações elevadas que devam interferir em nosso modelos. A correlação mais elevada é 'preço' vs 'preço dos concorrentes' com 0,58 e depois nenhuma correlação superior a 0,3. No entanto podemos fazer algumas inferências iniciais:

  - Investimento em publicidade tem uma maior correlação com Vendas e Tamanho Populacional. Talvez localidades em maiores mercados tendem

a investir mais em publicidade e lojas com grandes vendas fazem mais investimento em publicidade. Vale ressaltar que Vendas não tem uma alta correlação com tamanho da população.

- Investimento em publicidade tem correlação negativa com preço de concorrentes e Idade Populacional, mesmo sendo correlações baixas. Assim, podemos também inferir de que lojas em mercados com preços mais elevados sentem menor necessidade de investimento.
- Podemos ver um comportamento muito distinto no investimento em publicidade para lojas dentro e fora dos EUA

Com isso em mente, começamos a construção do modelo.

Definição da base de treino e teste

quebraremos em 80% treino (tr) e 20% teste (ts)

```
set.seed(1)
split <- initial_split(Carseats, prop = 0.8)

tr <- training(split) #treino
ts <- testing(split) #test

head(tr)
```

```
##   Sales CompPrice Income Advertising Population Price ShelveLoc Age
Education
## 1  9.50       138     73          11        276   120       Bad  42
17
## 2 11.22       111     48          16        260    83      Good  65
10
## 3 10.06       113     35          10        269    80    Medium  59
12
## 4  7.40       117    100           4        466    97    Medium  55
14
## 5  4.15       141     64           3        340   128       Bad  38
13
## 6 10.81       124    113          13        501    72       Bad  78
16
##   Urban  US
## 1   Yes Yes
## 2   Yes Yes
## 3   Yes Yes
## 4   Yes Yes
## 5   Yes  No
## 6    No Yes
```

Processamento

Para o processamento, usaremos o Tidymodels, normalizando as variáveis numéricas e preparando 10 grupos para fazer validações cruzadas

```
receita <- recipe(Advertising ~ ., tr) %>%
            step_normalize(all_numeric()) %>%
            step_dummy(all_nominal())


receita_prep <- prep(receita)

tr_process <- juice(receita_prep)
ts_process <- bake(receita_prep, new_data = ts)

cv_split <- vfold_cv(tr, v = 10)
```

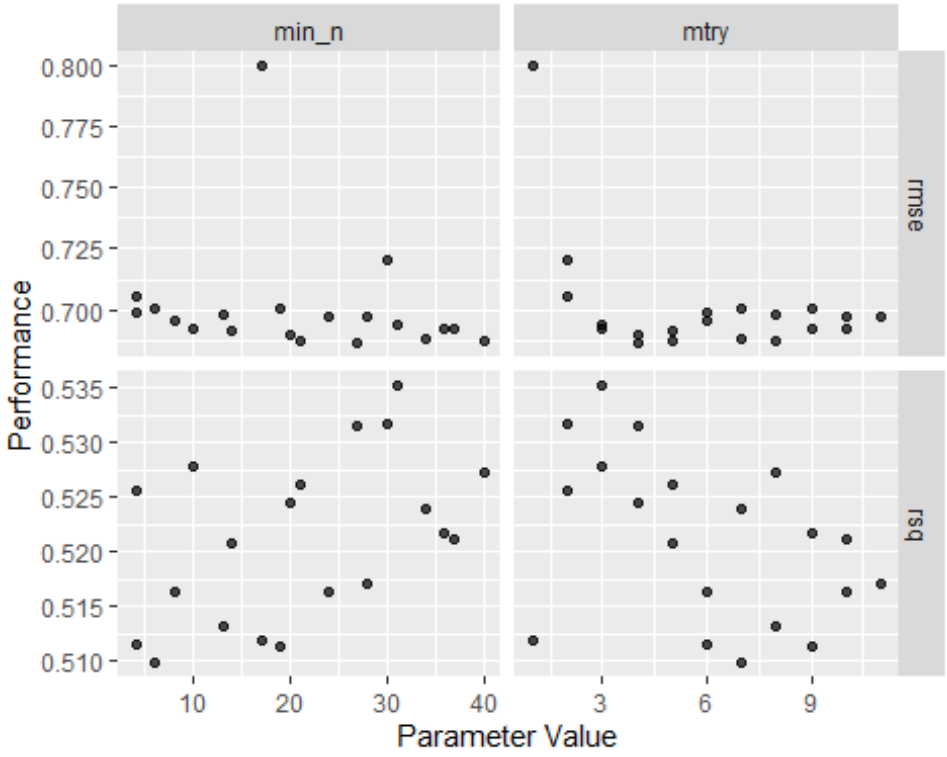
Modelos

Para predizer a variável 'Advertsiment', usaremos a Random Forest e SVM

Random Forest

Usaremos o pacote Ranger através do Tidymoldes, selecionando o número de preditores e observações mínimas por validação cruzada usando o Erro quadrático médio como métrica.

```
set.seed(1)
#definição do modelo
rf <- rand_forest(mtry = tune(), min_n = tune()) %>%
  set_mode("regression") %>%
  set_engine("ranger",importance = "permutation")

doParallel::registerDoParallel()

##computando diferentes combinacoes de parametros mtry e min_n
grid_rf <- tune_grid(rf,
                     receita,
                     resamples = cv_split,
                     grid = 20)

## i Creating pre-processing data to finalize unknown parameter: mtry

#avisao grafica do resultado
autoplot(grid_rf)
```




```
grid_rf %>%
  collect_metrics()
```

```
## # A tibble: 40 x 7
##     mtry min_n .metric .estimator  mean     n std_err
##    <int> <int> <chr>   <chr>      <dbl> <int>   <dbl>
##  1     1    17 rmse    standard   0.800    10  0.0421
##  2     1    17 rsq     standard   0.512    10  0.0313
##  3     2     4 rmse    standard   0.705    10  0.0395
##  4     2     4 rsq     standard   0.526    10  0.0353
##  5     2    30 rmse    standard   0.721    10  0.0414
##  6     2    30 rsq     standard   0.532    10  0.0346
##  7     3    10 rmse    standard   0.692    10  0.0402
##  8     3    10 rsq     standard   0.528    10  0.0359
##  9     3    31 rmse    standard   0.694    10  0.0412
## 10     3    31 rsq     standard   0.535    10  0.0323
## # ... with 30 more rows
```

```
#selecao do melhor hiperparametro
best_rf <- grid_rf %>%
  select_best(metric = "rmse")
best_rf
```

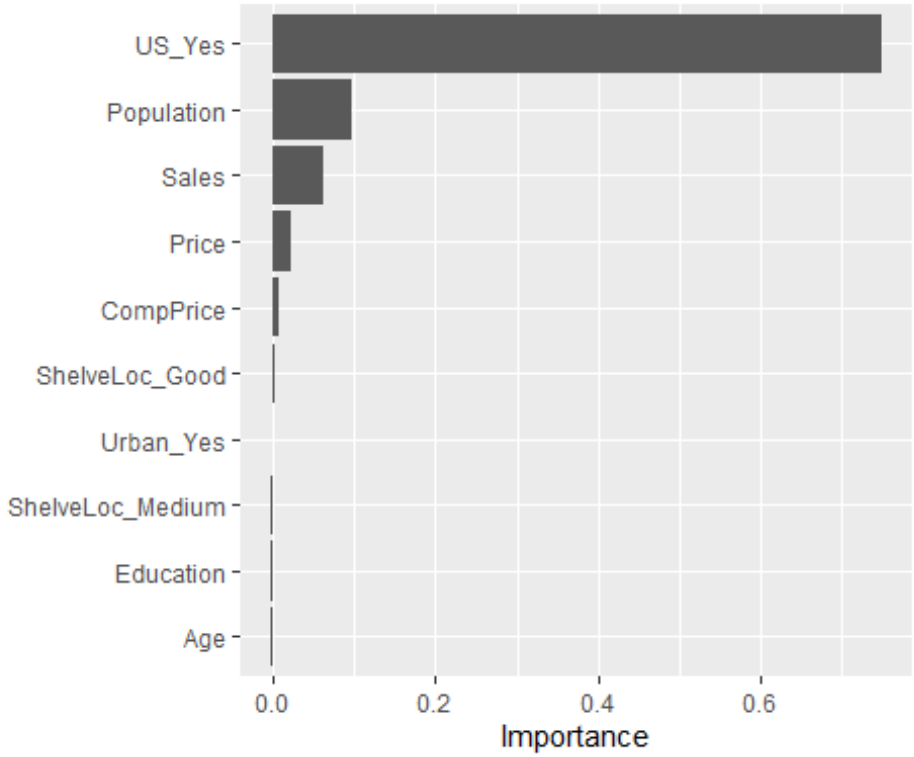
```
## # A tibble: 1 x 2
##    mtry min_n
##   <int> <int>
## 1     4    27
```

assim definimos a quantidade ótima de variáveis e observações mínimas para atingir o melhor erro quadrático médio.

```
#finalizando o modelo
rf_fit <- finalize_model(rf, parameters = best_rf)

#ajustando o modelo
rf_fit <- fit(rf_fit,
              Advertising ~ .,
              data = tr_process)

fitted <- rf_fit %>%
            predict(new_data = ts_process) %>%
              mutate(observado = ts_process$Advertising,
                modelo = "random forest")

rf_fit$fit
```

```
## Ranger result
##
## Call:
##  ranger::ranger(formula = formula, data = data, mtry = ~4L,
min.node.size = ~27L,      importance = ~"permutation", num.threads = 1,
verbose = FALSE,      seed = sample.int(10^5, 1))
##
## Type:                             Regression
## Number of trees:                  500
## Sample size:                      321
## Number of independent variables:  11
## Mtry:                             4
## Target node size:                 27
## Variable importance mode:         permutation
## Splitrule:                        variance
## OOB prediction error (MSE):       0.4826937
## R squared (OOB):                  0.5173063
```

```
# importância
vip(rf_fit)
```



Assim, dentro do nosso modelo de Random Forest a variável mais importante é Ser uma loja nos EUA (~40%), seguido por Vendas e Tamanho da População (~5% cada).

Aplicando nosso modelo na amostra teste:

```
x = rf_fit %>%
            predict(new_data = ts_process) %>%
              mutate(observado = ts_process$Advertising)
rmse(x, truth = observado, estimate = .pred)
```

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

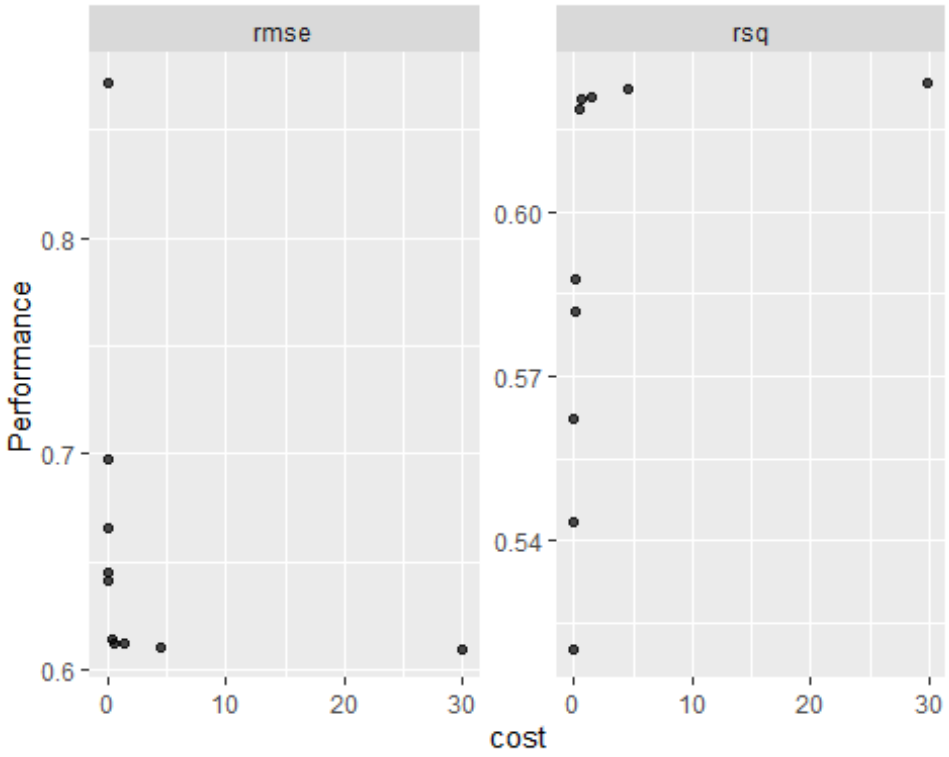
Temos um erro quadrático médio elevado.

SVM

Par ao SVM vamos usar o modelo Polinomial através do Tidymodel, ajustando o 'custo' para atingir o melhor erro quadrático médio.

```
#definição do modelo
svm <- svm_poly( cost = tune()) %>%
        set_args(kernel = "vanilladot") %>%
        set_mode("regression") %>%
        set_engine("kernlab")
```

```
##computando diferentes parametros do custo
grid_svm <- tune_grid(svm,
                      receita,
                      resamples = cv_split,
                      grid = 10)
#visao grafica do resultado
autoplot(grid_svm)
```




```
grid_svm%>%
  collect_metrics()
```

```
## # A tibble: 20 x 6
##        cost .metric .estimator  mean     n std_err
##       <dbl> <chr>   <chr>      <dbl> <int>   <dbl>
##  1  0.00123 rmse    standard   0.872    10  0.0436
##  2  0.00123 rsq     standard   0.520    10  0.0265
##  3  0.00623 rmse    standard   0.697    10  0.0409
##  4  0.00623 rsq     standard   0.543    10  0.0267
##  5  0.0153  rmse    standard   0.666    10  0.0383
##  6  0.0153  rsq     standard   0.562    10  0.0253
##  7  0.0485  rmse    standard   0.645    10  0.0348
##  8  0.0485  rsq     standard   0.582    10  0.0240
##  9  0.0630  rmse    standard   0.641    10  0.0341
## 10  0.0630  rsq     standard   0.588    10  0.0233
## 11  0.381   rmse    standard   0.614    10  0.0293
```

```
## 12  0.381   rsq     standard   0.618    10  0.0205
## 13  0.564   rmse    standard   0.612    10  0.0291
## 14  0.564   rsq     standard   0.621    10  0.0201
## 15  1.42    rmse    standard   0.612    10  0.0287
## 16  1.42    rsq     standard   0.621    10  0.0200
## 17  4.53    rmse    standard   0.611    10  0.0285
## 18  4.53    rsq     standard   0.622    10  0.0197
## 19 29.9     rmse    standard   0.610    10  0.0285
## 20 29.9     rsq     standard   0.624    10  0.0201

#selecao do melhor hiperparametro
best <- grid_svm %>%
          select_best(metric = "rmse")
best

## # A tibble: 1 x 1
##    cost
##   <dbl>
## 1  29.9
```

achamos o custo para minimizar o erro quadratico médio.

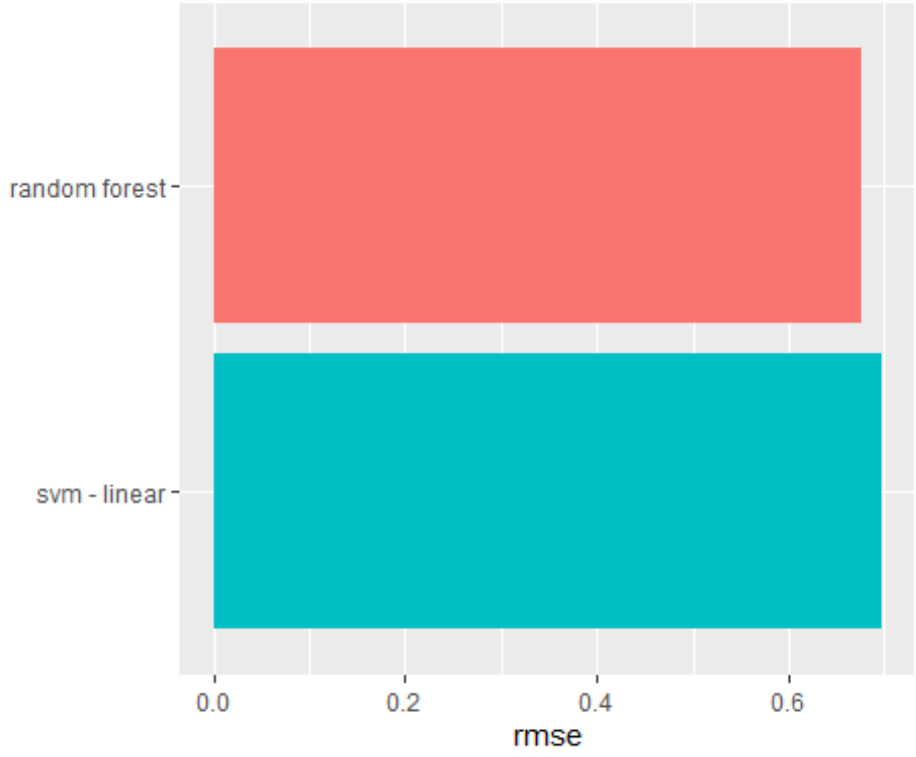
```
#finalizando o modelo
svm_fit <- finalize_model(svm, parameters = best)

#ajustando o modelo
svm_fit <- fit(svm_fit,
               Advertising ~ .,
               data = tr_process)

##  Setting default kernel parameters

#juntando os dois modelos para avaliação
fitted <- bind_rows(fitted,
                    svm_fit %>%
                     predict(new_data = ts_process) %>%
                     mutate(observado = ts_process$Advertising,
                            modelo = "svm - linear"))



x = svm_fit %>%
            predict(new_data = ts_process) %>%
              mutate(observado = ts_process$Advertising)
rmse(x, truth = observado, estimate = .pred)

## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       0.697
```

com o SVM vemos um resultado muito similar da Random Forest com um erro quadrático médio também elevado.

Avaliação

comparação do desemepenho dos dois modelos: Random Forest e SVM

```
fitted %>%
  group_by(modelo) %>%
  rmse(truth = observado, estimate = .pred) %>%
  ggplot(aes(reorder(modelo, -.estimate), .estimate, fill = modelo )) +
    geom_col(show.legend = FALSE) +
    labs(x = "", y = "rmse") +
    coord_flip()
```



Na comparação vemos que os dois modelos apresentam resultados muito semelhante, sendo praticamente permutáveis na prediçao da variável Advertisment.