# Lista 1 Aprenziado de Máquina

Lyncoln Sousa

2020-07-12

```
library(caret)
library(dplyr)
```

**1) Carregue o banco de dados Sacramento, nativo do pacote caret. Ele contém diversas informações sobre imóveis na região de Sacramento-CA. Vamos tentar predizer o preço do imóvel em função das suas características.**

```
data("Sacramento")
str(Sacramento)
```

```
## 'data.frame':    932 obs. of  9 variables:
##  $ city     : Factor w/ 37 levels "ANTELOPE","AUBURN",..: 34 34 34 34 34 34 34 34 29 31 ...
##  $ zip      : Factor w/ 68 levels "z95603","z95608",..: 64 52 44 44 53 65 66 49 24 25 ...
##  $ beds     : int  2 3 2 2 2 3 3 3 2 3 ...
##  $ baths    : num  1 1 1 1 1 1 2 1 2 2 ...
##  $ sqft     : int  836 1167 796 852 797 1122 1104 1177 941 1146 ...
##  $ type     : Factor w/ 3 levels "Condo","Multi_Family",..: 3 3 3 3 3 1 3 3 1 3 ...
##  $ price    : int  59222 68212 68880 69307 81900 89921 90895 91002 94905 98937 ...
##  $ latitude : num  38.6 38.5 38.6 38.6 38.5 ...
##  $ longitude: num  -121 -121 -121 -121 -121 ...
```

**2) Filtre apenas os imóveis localizado na cidade de SACRAMENTO.**

```
data = Sacramento %>%
  filter(city == "SACRAMENTO")
```

**3) Altere a variável price para escala logarítmica na base 10.**

```
data = data %>%
  mutate(price = log10(price))
```

**4) Fixe a semente no valor 100. Crie amostras treino (70%) e teste (30%).**

```r
set.seed(100)
treino_index = createDataPartition(data$price, p=0.7,list=F)
treino = data[treino_index,]
teste = data[-treino_index,]
```

**5) Treine um regressor pelo método svmLinear (Support Vector Machine com kernel linear, aprenderemos com detalhes em breve), utilizando apenas as variáveis explicativas beds, baths, sqft, type. Obtenha RMSE, R2 e MAE da amostra teste. Dica: no train(), utilize price beds+baths+sqft+type**

```r
#Treino
modeloFit_svmLinear = train(price ~ beds+baths+sqft+type,
                  data = treino,
                  method = "svmLinear")
modeloFit_svmLinear
```

```
## Support Vector Machines with Linear Kernel
##
## 308 samples
##   4 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 308, 308, 308, 308, 308, 308, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.1700181  0.4185146  0.1316613
##
## Tuning parameter 'C' was held constant at a value of 1
```

```r
#Teste
predicao_svmLinear = predict(modeloFit_svmLinear, teste)
postResample(predicao_svmLinear, teste$price)
```

```
##      RMSE  Rsquared       MAE
## 0.1689180 0.3752767 0.1335637
```

As estimativas dos erros out of sample e in sample foram parecidas. A qualidade de ajuste do modelo está um pouco melhor para a amostra treino.

**6) Realize os mesmos passos acima, mas treinando o regressor pelo método lm (regressão linear). Compare os resultados com o svmLinear. O que é possível observar?**

```r
#Treino
modeloFit_lm = train(price ~ beds+baths+sqft+type,
                  data = treino,
                  method = "lm")
modeloFit_lm
```

```
## Linear Regression
##
## 308 samples
##   4 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 308, 308, 308, 308, 308, 308, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.1638328  0.4321061  0.1279004
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
#Teste
predicao_lm = predict(modeloFit_lm, teste)
postResample(predicao_lm, teste$price)
```

```
##      RMSE  Rsquared       MAE
## 0.1691847 0.3764531 0.1342744
```

As estimativas dos erros out of sample foram um pouco maiores do que o in sample, enquanto a qualidade do ajuste do modelo foi menor.

Comparando as duas modelagens, observa-se que os erros in-sample e out-of-sample foram bastante parecidos. As qualidades de ajustes ($R^2$) também foram equivalentes, porém baixas, isto indica que esses modelos explicam pouco a variação dos dados. Com base nos valores dessas estatísticas obtidas, conclui-se que os modelos criados não são adequados para representar os dados.